Developing Adaptive Information Systems: Learning and Evolving through Human Feedback

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Zhendong Chu

May 2024

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Zhendong Chu

This Dissertation has been read and approved by the examing committee:

Advisor:	Hongning Wang
Advisor:	
Committee Member:	Aidong Zhang
Committee Member:	Yangfeng Ji
Committee Member:	Sheng Li
Committee Member:	Yen-Ling Kuo
Committee Member:	
Committee Member:	

Accepted for the School of Engineering and Applied Science:

[illegible]

Jennifer L. West, School of Engineering and Applied Science

May 2024

Abstract

Modern information systems, such as recommender systems, are typically characterized by their human-centric designs and adaptiveness, where the development of Human-Feedback-driven Learning (HFL) mechanisms is the central focus. However, most existing works treat human feedback as readily available ground-truth data, yet numerous challenges await dedicated resolutions: 1) From a system's perspective, real-world human feedback, typically collected from ordinary users, often lacks rigorous quality control. This results in feedback data that is overly noisy and requires sophisticated treatments, such as data cleansing and augmentation before it can be utilized to develop robust systems. 2) From users' perspective, disappointment arises when systems misinterpret their feedback or fail to react to their needs, leading to lower quality future feedback and hindering system improvement and user satisfaction. In this dissertation, we focus on modeling human feedback from both perspectives. On the one hand, we propose novel frameworks for learning from noisy and sparse human feedback. On the other hand, we devise algorithms that efficiently and effectively learn personalized policies, enabling systems to interpret and elicit users' interests and intentions through their feedback. We evaluate the effectiveness of proposed methods in various scenarios, including crowdsourcing, recommender systems, and nature language generation.

Acknowledgements

Completing this thesis has been one of the most challenging yet rewarding endeavors of my academic journey, made all the more challenging by the unprecedented times of the pandemic. It would not have been possible without the support from a lot of individuals. I am profoundly grateful to everyone who contributed to this journey.

First and foremost, I extend my deepest thanks to my advisor, Prof. Hongning Wang, for his invaluable guidance, patience, and unwavering support during my PhD journey. Prof. Wang has been more than just a mentor; he is a remarkable role model in academia and a genuine friend outside of it. His hardworking character, enthusiasm for tackling interesting problems, and his insightful suggestions and keen comments have greatly inspired me. I sincerely appreciate his devotion and the personal and professional growth I have experienced under his mentorship. His example has deeply influenced my approach to research and will undoubtedly continue to inspire me in my future endeavors.

I also wish to express my genuine gratitude to my thesis committee members, Prof. Aidong Zhang, Prof. Yangfeng Ji, Prof. Sheng Li, and Prof. Yen-Ling Kuo for their constructive feedback, thoughtful advice, and the time they invested in in reviewing my work. I have benefited a lot from discussions with them and their perspectives have greatly enriched this dissertation.

Moreover, I would like to thank my mentors Dr. Tong Sun, Dr. Ruiyi Zhang, Dr. Tong Yu, and Dr. Zichao Wang at Adobe Research, along with Dr. Lingfei Wu at JD.com Silicon Valley Research Center, for their invaluable guidance and tremendous support in industrial research. My gratitude also goes to my collaborators, Renqin Cai, Nan Wang, Jing Ma and Xuchuang Wang. They have provided a lot of insightful suggestions to my research. Additionally, I am grateful to all the members of the HCDM lab for fostering a hardworking and supportive environment, crucial for both research and life. Their collective effort has truly made a difference.

Personally, I wish to express my heartfelt appreciation to my family. They are my biggest supporters and my strongest source of resilience during this challenging journey. To my friends, the moments we shared with each other always heal me in every difficult time, especially during the pandemic. Your companion makes this journey more colorful. Last but not least, I owe many thanks to my lovely kitten, Fuji. Her gentle breathing while sleeping always makes me peaceful while working.

To my family and my beloved kitten, Fuji

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern information systems are typically characterized by their human-centric designs and adaptiveness, such as online shopping platforms, recommender systems, etc. Evolving in tandem with human users is crucial for their success. Within this human-system ecosystem, users are not just passive recipients of information and services; they are active contributors. Their feedback plays a crucial role in the continuous learning and evolution of systems. Based on this background, a new learning paradigm has emerged, known as Human-Feedback-driven Learning (HFL). In this dissertation, we focus on developing advanced HFL techniques from both human and system perspectives. Figure 3.11 depicts the human feedback loop between human users and systems: Human users contribute feedback, which serves as a foundational learning source for the systems (the upper arc). In response, systems engage users in an interaction loop, prompting them for feedback to further refine and personalize their experiences. HFL is extensively employed to train systems, both by aggregating consensus from a wide range of users, such as in crowdsourcing systems [148, 172, 202], and by delivering personalized experiences, as seen in recommender systems [120].

1.1 Challenges

However, despite the promising prospects of HFL, several oversights and challenges remain to be addressed. The prevalent approach to HFL has been to treat human feedback as readily available ground-truth data [31, 156]. Yet, this perspective is overly simplistic and fails to consider the complexities inherent in human feedback. My proposed research focus on two interrelated challenges: (1) *How do systems sieve through noisy and non-expert feedback to extract meaningful insights?* Real-world human feedback is typically

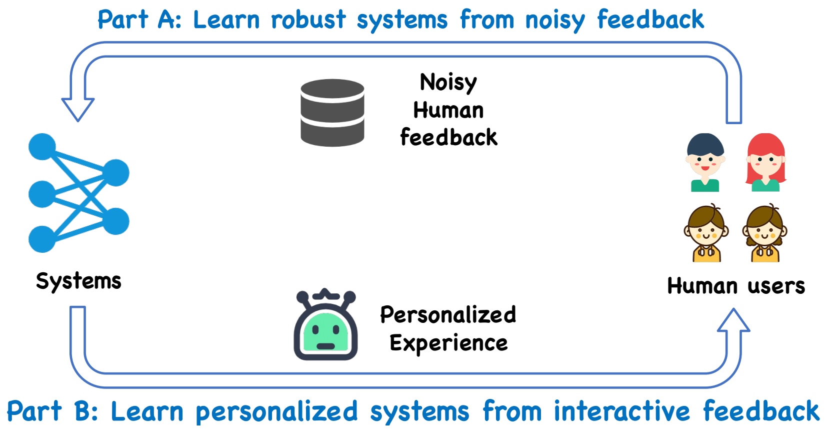

Part A: Learn robust systems from noisy feedback
Noisy Human feedback
Systems
Human users
Personalized Experience
Part B: Learn personalized systems from interactive feedback

Figure 1.1: An illustration of the human feedback loop between human users and systems. This dissertation focuses on two key areas: (1) improving the system's ability to learn from real-world human feedback, particularly when confronted with noise, shown as the upper arc; (2) refining how systems interpret and elicit human feedback to provide a personalized experience, shown as the loop.

collected from ordinary users and lacks the necessary quality control. Given the powerful memorization capabilities of increasingly larger neural networks, this lack of quality can lead to a fundamental conflict of knowledge and potentially ruin the learning process due to the noisy feedback [153]. Thus, we are motivated to design more robust learning algorithms capable of deriving effective models from real-world human feedback. (2) *How to facilitate sustainable human-system interaction?* For human users, if systems are perceived as inadequate or unresponsive for their feedback, users may become disillusioned, leading to feedback fatigue or even disengagement [58, 111, 143], calls for more effective and efficient feedback interpretation and elicitation approaches. Each challenge requires specialized treatment, which will be discussed in detail in the following paragraphs.

Learning from noisy human feedback. Standard supervised learning systems are data hungry, especially for labeled data, which unfortunately is expensive to acquire at scale. Crowdsourcing provides a label collection schema from human users that is both cost- and time-efficient [15], enabling the training of large-scale systems at a feasible expense. However, several practical challenges may impede training systems using such human feedback: (1) Due to varying and unknown expertise of human annotators, crowdsourced annotations are usually noisy. It is reported that ImageNet [45], a highly influential dataset created through crowdsourcing, contains 5.83% label errors[1]. Properly modeling the generation of noise in crowdsourcing is the first step to learn an accurate model from crowdsourced annotations. Thus, we propose a novel noise model accounting for the common noise based

[1]https://l7.curtisnorthcutt.com/tag/noisy-labels/

on empirical analysis on real-world crowdsourcing datasets. (2) Ideally, the "wisdom of crowds" should mitigate the noise in annotations through redundancy, i.e., by obtaining multiple labels from different annotators for each instance. However, in practice, to minimize annotation cost, the instances in crowdsourced data are typically labeled by a small number of annotators; and each annotator will only be assigned to a few instances. This introduces serious sparsity in crowdsourced data, potentially compromising the effectiveness of noise reduction. To address the issue of data sparsity, we have devised a new framework that generates authentic-like and informative pseudo-annotations, serving to augment the data obtained from crowdsourcing. (3) Recently, it has been shown that human-annotated data is crucial for fine-tuning Large Language Models (LLMs) for downstream tasks [127]. However, LLMs fine-tuned with data from annotators of varying fidelities exhibit differing performance levels. To mitigate the noise of low-fidelity data, we introduce a novel fine-tuning framework designed to improve low-fidelity LLMs with the guidance from high-fidelity LLMs, all without directly accessing the high-fidelity LLMs' training data.

Learning from interactive human feedback. Human-system interactions are no longer limited to one-way communication. It is essential to promptly and precisely comprehend and respond to human feedback in order to foster sustainable human-system interactions. The following challenges are needed to be resolved to provide a more personalized experience from both *efficacy* and *efficiency* perspectives: (1) How to assign credit accurately to systems for actions that contribute to sustainable interactions? To facilitate sustainable human-system interactions, the system must not only capture user preferences but also ensure their satisfaction during the conversation. Appropriately rewarding the system is crucial for it to strategically act in alignment with user interests and intentions. Thus, we are motivated to learn the optimal reward function to effectively and efficiently elicit user feedback in the conversation. (2) How to efficiently create personalized systems with limited human feedback? The tolerance of users about a prolonged interactions or unmet goals is limited [58, 111, 143]. Hence, it is impractical to rely on extensive interactions to tailor a system to individual preferences. We then propose to efficiently learn personalized policies by adapting from a meta policy. (3) Also, the recent emerged LLM-based systems [110, 119] pose new challenges in capturing user preferences in human-system interactions, where we find LLMs are ineffective in interpreting and utilizing temporal information in sequential user interactions. Consequently, we design a principled prompting framework to improve the temporal awareness of LLMs.

1.2 Thesis Organization

The rest of this dissertation is structured as follows:

In Chapter 2, we investigate advanced approaches for learning from noisy human feedback. First, we introduce a novel noise model for human feedback that distinguishes between common noise shared across annotators and the individual confusions specific to each annotator. Building on this, we develop a Common Noise Adaptation Layer (CoNAL) designed to capture and address common noise within existing end-to-end training frameworks. Second, to address the challenge of sparse human feedback, we employ data augmentation through generative models to enrich the dataset with missing annotations. We introduce a generative adversarial framework, named CrowdInG, specifically designed to produce informative annotations. Third, LLMs show significant potential in generating human-like data, offering a pathway to reduce reliance on human feedback and simplify the data cleansing process. We propose a novel Guided Fine-Tuning (GFT) framework to improve low-fidelity LLMs with the guidance from high-fidelity LLMs, all without directly accessing the high-fidelity LLMs' training data.

In Chapter 3, we study the problem of learning from interactive human feedback. We specifically focus on the application in recommendation systems, where interactive human feedback is ubiquitous. First, we introduce a multi-objective optimization framework to learn the optimal rewards for conversational recommender systems (CRS). This framework learns reward functions that foster both effective and efficient conversational policies in interactions with human users. Second, we address the problem of CRS policy learning for cold-start users via a novel meta-reinforcement-learning framework (MetaCRS). Third, it is empirically found that LLMs fall short in recognizing and utilizing temporal information, rendering poor performance in tasks that require an understanding of sequential data, such as sequential recommendation. We design a principled prompting framework Tempura, inspired by the human cognitive process, to interpret complex user interests and intentions. This is achieved by leveraging the zero-shot reasoning capabilities of LLMs.

In Chapter 4, we summarize this dissertation and discuss future research directions.

Chapter 2

Learning from Noisy Human Feedback

The first objective of my research is to enhance the system's ability to learn from real-world human feedback, with the goal of significantly improving its learning performance. Due to the varying and unknown expertise of human annotators, crowdsourced annotations are often noisy, which poses challenges for training high-quality systems. Moreover, collecting human annotations is both costly and time-consuming, necessitating the development of efficient and effective learning algorithms. In this chapter, we discuss solutions to learn from noisy human feedback.

2.1 Learning from crowds by modeling common confusions

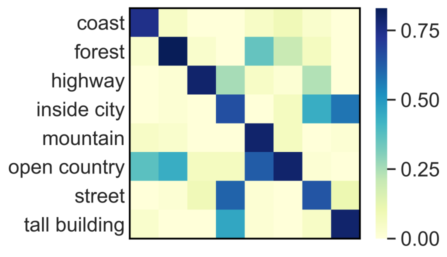
2.1.1 Introduction

The availability of large amounts of labeled data is often a prerequisite for applying supervised learning solutions in practice. Crowdsourcing makes it possible to collect massive labeled data in both time- and cost-efficient manner [15]. However, because of varying and unknown expertise of annotators, crowdsourced labels are usually noisy, which naturally lead to an important research problem: *how to train an accurate learning model with only crowdsourced annotations?*

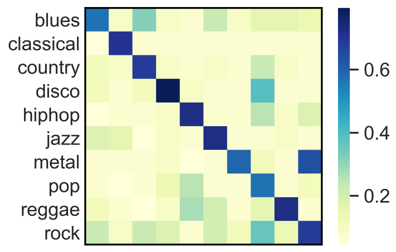
The first step to estimate an accurate learning model from crowdsourced annotations is to properly model the generation of such data. In this work, we focus on the crowdsourced classification problem. The seminal work from Dawid and Skene [43] (known as the DS model) assumes that each annotator has his/her own class-dependent confusion when providing annotations to instances. This is modeled by an annotator-specific confusion matrix, whose entries are the probability of flipping one class into another. The DS model has become the cornerstone of most learning from crowds solutions; and mainstream solutions perform label aggregation prior to classifier training: their key difference lies on different label aggregation methods based on the DS model [173, 185, 209]. Recent developments focus more on unified solutions, where variants of the Expectation-Maximization (EM) algorithm are proposed to integrate label aggregation and classifier training [2, 20, 133]. Typically, such solutions treat the classifier's predictions as latent variables, which are then mapped to the observed crowdsourced labels using individual confusion matrices of annotators. Rodrigues and Pereira [138] further fuse label inference and classifier training in an end-to-end approach using neural networks, where the gradient from label aggregation is directly propagated to estimate the annotators' confusion matrices. Tanno et al. [163] propose a similar solution but encourage the annotator confusion matrix to be close to an identity matrix by trace regularization.

All existing DS-model-based solutions assume noise in crowdsourced labels is only caused by individual annotators' expertise. However, it is not uncommon that different annotators would share common confusion about the labels. For example, when a *bird* in an image is too small, every annotator has a chance to confuse it with an *airplane* because of the background sky. We hypothesize that on an instance the annotator is confident about, he/she is more likely to use his/her expertise to provide a label (i.e., introducing individualized noise), while he/she would use common sense to label those unconfident ones. We empirically evaluate this hypothesis on two public crowdsourcing datasets, one for image labeling and one for music genre classification (more details of the datasets can be found in the Experiment Section), and visualize the results in Figure 2.1. On both datasets, there are quite some commonly made mistakes across annotators. For example, on the image labeling dataset LabelMe, 61.0% annotators mistakenly labeled *street* as *inside city* and 44.1% of them mislabeled *open country* as *forest*; on the music classification dataset, 63.6% annotators mislabeled *metal* as *rock* and 38.6% of them mislabeled *disco* as *pop*. The existence of such shared confusions across annotators directly affects label aggregation: the majority of annotators are not necessarily correct, as their mistakes are no longer independent (e.g., those large off-diagonal entries in Figure 2.1). This is against the fundamental assumption in the DS model, and strongly urges new noise modeling to better handle real-world crowdsourced data.

Moving beyond the independent noise assumption in the family of DS models [43, 138], we decompose annotation noise into two sources, *common noise* and *individual noise*, and differentiate the source of noise based on both annotators and instances. We refer to the annotation confusions shared across annotators as common noise, and model it by a global



(a) LabelMe



(b) Music

Figure 2.1: Analysis of commonly made mistakes across annotators on two real-world crowdsourcing datasets. The value of each entry in the heatmap denotes the percentage of annotators with this confusion pair (e.g., mistakenly label *street* as *inside city* on LabelMe dataset).

confusion matrix shared by all annotators. In the meanwhile, we also maintain annotator-specific confusion matrices for individual noise modeling. We still treat ground-truth labels of instances as latent variables, but map them to noisy annotations by two parallel confusion matrices, to capture these different sources of noise. We determine the choice of confusion matrices on a per-instance-annotator basis, by explicitly modeling of annotator expertise and instance difficulty [185, 198]. To leverage the power of representation learning to model annotator expertise and instance difficulty, we realize all our model components using neural networks. In particular, we model the two types of confusion matrices as two parallel noise adaptation layers [60]. For each annotator-instance pair, the classifier first maps the instance to a latent class label, then an auxiliary network decides which noise adaptation layer to map the latent class label to the observed annotation. Cross-entropy loss is counted on the predicted annotations for end-to-end training of these components. We name this approach *CoNAL* - learning from crowds with **co**mmon **n**oise **a**daptation **l**ayers. Extensive experiments show considerable improvement of our new noise modeling approach against a rich set of baselines on two synthesized datasets, including a fully synthesized dataset and one based on CIFAR-10 dataset with various settings of noise generation, as well as two real-world datasets, e.g., LabelMe for image classification, and Music for music genre classification.

2.1.2 Related works

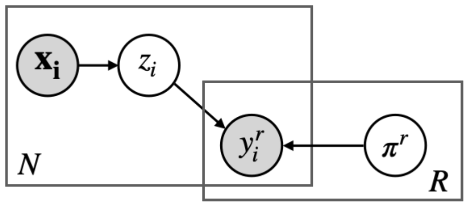
Several existing studies focused on modeling the different roles of instance and annotator in crowdsourced data. Whitehill et al. [185] model the accuracy of each annotation, which depends on instance difficulty and annotator expertise, to weigh each instance in final majority vote. Welinder et al. [184] model each annotator as a multi-dimensional classifier and consider instance difficulty as single dimension latent variable. Zhou et al. [214] propose a minimax entropy principle on a probability distribution over annotators, instances and annotations, in which by minimizing entropy instance confusability and annotator expertise are naturally inferred. Khetan and Oh [94] and Shah, Balakrishnan, and Wainwright [147] consider generalized DS models which model the instance difficulty. Instead of simply using a single scalar to model instance difficulty and annotator expertise as in previous works, we model them by learning their corresponding representations via an auxiliary network, which can better capture the shared statistical pattern across observed annotations.

Our method is closely related to several existing DS-based models considering relations among annotators; but it is also clearly distinct from them. Kamar, Kapoor, and Horvitz [90] use a global confusion matrix to capture the identical mistakes by all annotators, and it is designed to replace the individual matrix when observations of an annotator are rare. Moreover, the choice of confusion matrix in this solution only depends on the number of annotations an annotator provided. This unnecessarily reflects the annotator expertise, as the task assignment is typically out of their control in crowdsourcing. Venanzi et al. [173] and Imamura, Sato, and Sugiyama [86] cluster annotators to generate their own confusion matrices from a shared community-wide confusion matrix. However, the above approaches still assume a single underlying noise source, and thus they do not consider the difference between global (or community-level) and individual confusions. Li, Rubinstein, and Cohn [114] explore the correlation of annotation across annotators by classifying them into auxiliary subtypes under different ground-truth classes. However, the characteristics of each annotator are missing since they are only represented by a specific subtype. In our work, we still characterize individual annotators by modeling their own confusions.

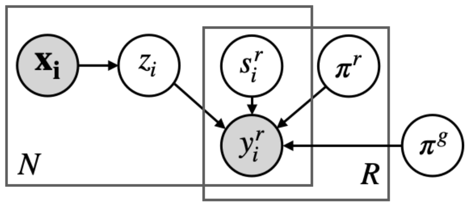
2.1.3 Common confusion modeling in crowdsourced data

Assume we have N instances labeled by R annotators out of C possible classes. We define $\boldsymbol{x}_i$ as the feature vector of the i-th instance and y_i^r as its label provided by the r-th annotator. Denote z_i as the unobservable ground-truth label for the i-th instance, which is considered as a latent variable sampled from a multinomial distribution parameterized by $\{p(z_i = c|\boldsymbol{x}_i)\}_{c=1}^{C}$. For simplicity, we collectively define $X = \{\boldsymbol{x}_i\}_{i=1}^{N}$, $Y = \{y_i^r\}_{i=1,r=1}^{N,R}$ and $Z = \{z_i\}_{i=1}^{N}$. The final goal of learning from crowds is to obtain the classifier $P(Z|X)$ only with crowdsourced annotations Y.

Similar to the DS-based models (see Figure 2.2a for reference), the confusion of the r-th annotator is measured by an annotator-specific confusion matrix π^r, in which the



(a) DS model.



(b) Common noise model.

Figure 2.2: Graphical model presentations of the DS model and our common noise model.

(z, z')-element $\pi^r_{z,z'}$ denotes the probability that annotator r will label the true label z as z'. Aside from individual confusion, the key assumption of our solution is that annotation mistakes can also be introduced by common confusion, which is modeled by a globally shared confusion matrix π^g across all annotators. We define the confusion matrices set as $\Pi = \{\pi^{1:R}, \pi^g\}$. We associate a Bernoulli random variable $s_i^r \sim B(\omega_i^r)$ with each annotation y_i^r to differentiate the source of noise on it: s_i^r=1 if the confusion is caused by the common noise, where w_i^r is the probability of the global confusion matrix being chosen by annotator r on instance i (see Figure 2.2b). Denote the set of parameters governing the generation of s_i^r across all annotations as Ω.

Suggested by the successful practice in modeling crowdsourced data, we also impose the following two commonly made assumptions: 1) each annotator provides their annotations independently [43]; and 2) each annotation is independent from the instance's features given the ground-truth labels [138, 194]. We should note the first assumption is *not* contradicting to our common confusion modeling: as the annotators can independently choose the shared common noise model to generate their annotations, the resulting observed annotations are no longer independent across annotators. As a result, the complete data likelihood of observed annotations under our model can be defined as,

$$\begin{aligned} p\left(Y, Z|X, \Pi, \Omega\right) &= \prod_{i=1}^{N}\prod_{r=1}^{R}\sum_{z=1}^{C} p\left(y_i^r|z_i; \Pi, \omega_i^r\right)p(z_i|\boldsymbol{x}_i)\,, \\ p\left(y_i^r|z_i; \Pi, \omega_i^r\right) &= \omega_i^r p\left(y_i^r|z_i, \pi^g\right) + \left(1-\omega_i^r\right) p\left(y_i^r|z_i, \pi^r\right)\,. \end{aligned} \tag{2.1}$$

Based on the above imposed problem structure, we derive an information-theoretical lower bound about the resulting noise modeling quality. Let $\hat{Z}$ be the estimated true labels of all instances. Noise modeling quality is measured by the error rate given by $\mathcal{L}(\hat{Z}, Z) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\hat{z}_i \neq z_i\right)$, where $\mathbb{I}(\cdot)$ is an indicator function. Given the ground-truth instance-specific class distribution $\rho_i = \{\rho_{ic}\}_{c=1}^{C}$ and confusion matrices Π, we have the following theorem about the lower bound of minimax error rate of our model.

Theorem 1. *The minimax error rate of our model is lower bounded by*

$$\inf_{\hat{Z}} \sup_{Z\in[C]^N} \mathbb{E}\left[\mathcal{L}(\hat{Z}, Z)\right] \geq \frac{1}{N^2 \log C} \sum_{i=1}^{N} F(\rho_i, \Pi, \Omega) - \frac{\log 2}{N^2 \log C},$$

$$\begin{aligned} F(\rho_i, \Pi, \Omega) =& H(\rho_i) - \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{c'=1}^{C} \rho_{ic}\rho_{ic'} \Big(\omega_i^r \,\mathrm{KL}(\pi_{c*}^g \parallel \pi_{c'*}^g) \\ &+ (1-\omega_i^r)\,\mathrm{KL}\left(\pi_{c*}^r \parallel \pi_{c'*}^r\right)\Big). \end{aligned}$$

where $H(\rho_i) = -\sum_{c=1}^{C} \rho_{ic} \log \rho_{ic}$ is the entropy of ground-truth class distribution and π_{c*} is the c-th row in confusion matrix π.

Proof. In our setting, the ground-truth class distribution ρ_i depends on the instance features. Then the minimax error rate of the crowdsourcing problem can be lower bounded by the following,

$$\inf_{\hat{Z}} \sup_{Z\in[C]^N} \mathbb{E}\left[\mathcal{L}(\hat{Z}, Z)\right] \geq \frac{1}{N^2 \log C} \sum_{i=1}^{N} R(\rho_i, \Pi') - \frac{\log 2}{N^2 \log C} \tag{2.2}$$

where

$$R(\rho_i, \Pi') = H(\rho_i) - \sum_{r=1}^{R} \sum_{c=1}^{C} \sum_{c'=1}^{C} \rho_{ic}\rho_{ic'} \mathrm{KL}(\pi_{c*}^{\prime r} \parallel \pi_{c'*}^{\prime r}) \tag{2.3}$$

and $\Pi' = \{\pi'^r\}_{r=1}^R$ denotes the set of annotator-level confusion matrices. We use π' to differentiate with our defined individual confusion matrix in the main paper. The proof of Eq (2.2) is similar to [86]. Based on our new noise generation assumption, the annotation noise can be decomposed by common noise and individual noise. Thus we can further bound the minimax error rate under this noise assumption.

Under our new noise assumption, we can evaluate the confusion matrix on a per-instance-annotator basis. Specifically, in each annotation, the effective confusion matrix is a weighted combination of the global and individual confusion matrices, where the weight is w_i^r. In a mixture model, the Kullback–Leibler divergence can be decomposed accordingly by,

$$\begin{aligned} \mathrm{KL}(\pi_{c*}^{\prime r} \parallel \pi_{c'*}^{\prime r}) &= \mathrm{KL}(\omega_i^r \pi_{c*}^g + (1-\omega_i^r)\pi_{c*}^r \parallel \omega_i^r \pi_{c'*}^g + (1-\omega_i^r)\pi_{c'*}^r) \\ &\leq \mathrm{KL}(\boldsymbol{\omega}_i^r \parallel \boldsymbol{\omega}_i^r) + \omega_i^r\,\mathrm{KL}(\pi_{c*}^g \parallel \pi_{c'*}^g) + (1-\omega_i^r)\,\mathrm{KL}(\pi_{c*}^r \parallel \pi_{c'*}^r) \quad (2.4) \\ &= \omega_i^r\,\mathrm{KL}(\pi_{c*}^g \parallel \pi_{c'*}^g) + (1-\omega_i^r)\,\mathrm{KL}(\pi_{c*}^r \parallel \pi_{c'*}^r) \quad (2.5) \end{aligned}$$

where $\boldsymbol{\omega}_i^r = (\omega_i^r, 1-\omega_i^r)$. The inequality can be derived by the log-sum inequality. Substitute Eq (2.5) back to Eq (2.3), we can get the new term $F(\rho, \Pi, \Omega)$ in Theorem 1. Plug it back into Eq (2.2), we can get the refined result in Theorem 1. □

Remarks. This result extends the known lower bound result of DS models [86]. Lower bound on the error rate measures the difficulty of a crowdsourcing problem. Theorem 1 suggests the proposed decomposition has the potential to further reduce the lower bound, i.e., to obtain better inferred true labels. To understand this result, we should first note that the lower bound mainly depends on the KL distance between the class distributions conditioned on different ground-truth classes, as defined in $F(\rho_i, \Pi, \Omega)$, i.e., how two different classes will be confused with other classes. The more different they are (i.e., a larger KL distance), the easier one can differentiate the two from the observed noisy labels. For example, consider a crowdsourced dataset where an annotator labels a set of instances as *airplane*; but among them, 50% cases should be *bird*, and the other 50% should be *spacecraft*. Intuitively, without any additional knowledge, it is hard to determine the true label when he/she labels an instance as *airplane*. And this is asserted by Theorem 1: If we only used a single confusion matrix for this annotator, the conditional class distributions for *bird* and *spacecraft* will be pushed closer, because their entries on *airplane* are close. This causes a smaller KL term in $F(\rho_i, \Pi, \Omega)$ between *bird* and *spacecraft* (e.g., setting ω_i^r=0 for all instances in annotator r). But if we knew that the confusion between *bird* and *airplane* is caused by common noise, and the confusion between *spacecraft* and *airplane* is caused by individual noise, these mistakes could be attributed to two confusion matrices separately, which eliminates the misleading similarity between the conditional probabilities for *bird* and *spacecraft* caused by *airplane*.

2.1.4 End-to-end learning framework

To apply our noise modeling in crowdsourced data, we need to estimate the confusion matrices Π together with the classifier. Instead of building a vanilla tabular model for them, we realize them using neural models, to take advantage of the power of representation learning. In particular, we map the output of the classifier to noisy annotations by two types of confusion layers, which we refer to as noise adaptation layers [60]. We also introduce an auxiliary network that takes both annotator and instance as input to predict the choice of these two noise adaptation layers. Since we treat the ground-truth label of an instance as a latent variable, the Expectation Maximization (EM) algorithm becomes a natural choice for model learning, as typically done in literature [2, 12, 138]. However, the EM-based algorithm has several clear drawbacks in our solution: 1) In crowdsourced data, because the annotators typically only label a small proportion of instances, EM-based algorithm becomes very sensitive to the initialization of model parameters. It can easily cause instability issues in training a neural network model. 2) In every EM iteration, we need to retrain the neural network, which causes a huge overhead when handling large networks. Instead, we take an end-to-end approach to jointly perform latent variable inference and model parameter estimation. We define cross-entropy loss on the observed annotations and use error back-propagation to update the classifier's output and the network parameters simultaneously.

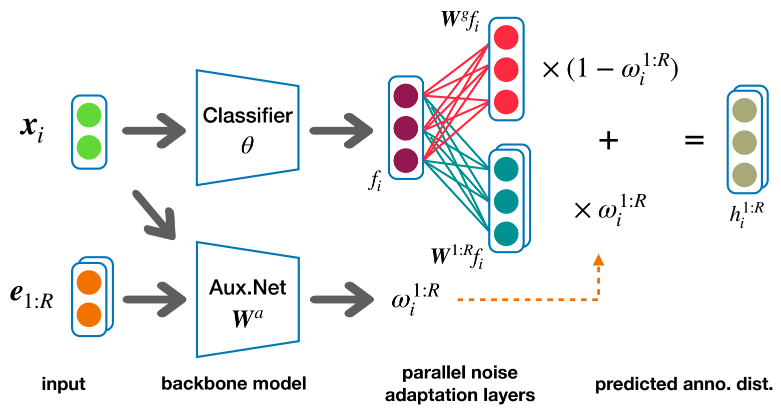


Figure 2.3: Overview of our framework for classification with 3 classes and R annotators.

We construct a neural network classifier with non-linear intermediate layers and a softmax output layer. The probability distribution of the predicted true label z_i given the instance feature vector $\boldsymbol{x}_i$ is thus specified as $p_{\boldsymbol{\theta}}(z_i|\boldsymbol{x}_i)$, where $\boldsymbol{\theta}$ is the network parameter set including the softmax layer. We denote the immediate output of the classifier as $f_i = f(\boldsymbol{x}_i) \in \mathbb{R}^C$. We then use noise adaptation layers to map the classifier's output into noisy annotations, which are implemented by introducing additional softmax output layers on top of the output layer of the classifier (see overview in Figure 3.7). The weight matrices of the noise adaptation layers resemble confusion matrices Π in a probabilistic sense. The output of the noise adaptation layer is thus the probability distribution of predicted annotation $p_{\boldsymbol{W}}(\hat{y}_i^r|f(\boldsymbol{x}_i))$, where $\boldsymbol{W}$ is the parameter set of the noise adaptation layer.

We consider two types of noise adaptation layers: one individual noise adaptation layer for every annotator parameterized by $\boldsymbol{W}^r$, and a common noise adaptation layer shared across all annotators parameterized by $\boldsymbol{W}^g$. The final probability distribution of annotations is obtained as,

$$p(\hat{y}_i^r|\boldsymbol{x}_i) = \omega_i^r\, p_{\boldsymbol{W}^g}(\hat{y}_i^r|f(\boldsymbol{x}_i)) + (1 - \omega_i^r)\, p_{\boldsymbol{W}^r}(\hat{y}_i^r|f(\boldsymbol{x}_i)).$$

where ω_i^r governs the distribution that the mistake of annotator r on instance i is caused by common confusion π^g, denoted by the noise source indicator s_i^r.

As s_i^r is unobservable, we introduce an auxiliary network to model $s_i^r \sim B(\omega_i^r)$ by parameterizing it over annotator expertise and instance difficulty, both of which are modeled via learnt representations by the auxiliary network. Specifically, as in our problem setup, every instance is associated with raw features, the auxiliary network takes instance feature $\boldsymbol{x}_i$ as input for learning instance i's embedding $\boldsymbol{v}_i$. The same can be applied to annotator r, if any raw feature $\boldsymbol{e}^r$ is available about the annotator, otherwise we use its one-hot encoding as input for learning annotator embedding $\boldsymbol{u}_r$. Then ω_i^r can be obtained as follows,

$$\begin{gathered} \boldsymbol{v}_i = \boldsymbol{W}_v \boldsymbol{x}_i + b_v, \boldsymbol{u}_r = \boldsymbol{W}_u \boldsymbol{e}_r + b_u, \\ \omega_i^r = \sigma(\boldsymbol{u}_r^\top \boldsymbol{v}_i). \end{gathered} \tag{2.6}$$

where $(\boldsymbol{W}_v, b_v)$ and $(\boldsymbol{W}_u, b_u)$ are weight matrices and bias terms for annotator and instance embeddings, and σ is a sigmoid function. To simplify our notations, we collectively refer the parameters in this auxiliary network as $\boldsymbol{W}^a$. To avoid the magnitude of learnt $\boldsymbol{u}$ and $\boldsymbol{v}$ becoming extremely large or small, which causes numerical issues in estimating ω_i^r, we normalize the learnt annotator and instance embeddings before computing their inner product.

Based on the above full specifications of our probabilistic modeling using neural networks, we are ready to estimate the network parameters. We can easily verify that, maximizing the likelihood of observed annotations given the input feature vectors as defined in Eq (2.1) is equivalent to minimizing the cross-entropy loss between the observed annotations and predicted annotation distributions,

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{W}^g, \boldsymbol{W}^{1:R}, \boldsymbol{W}^a) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{r=1}^{R} \sum_{j=1}^{C} y_{ij}^r \log p_j(\hat{y}_i^r | \boldsymbol{x}_i).$$

where $y_{ij}^r = 1$ if $y_i^r = j$; otherwise $y_{ij}^r = 0$; and $p_j(\hat{y}_i^r|\boldsymbol{x}_i)$ refers to the j-th entry of the predicted annotation distribution. All parameters can be trained by back-propagation using gradient descent techniques, such as Adam [96] and SGD [61]. Once trained, in the testing phase, we can directly use the classifier to make predictions on new instances.

The gradient flow in back-propagation reveals how our common confusion modeling handles crowdsourced data. In the context of classification, we can simply view the introduced noise adaptation layer as performing a projection of gradients; and with a slight abuse of notations, we denote the output of our noise adaptation layers as $h_i^r = \omega_i^r \boldsymbol{W}^g f_i + (1 - \omega_i^r) \boldsymbol{W}^r f_i$. Under the chain rule, the gradients are naturally decoupled with respect to different sources of noise,

$$\frac{\partial \mathcal{L}}{\partial f_i} = \sum_{r=1}^{R} \frac{\partial \mathcal{L}}{\partial h_i^r} \frac{\partial h_i^r}{\partial f_i} = \sum_{r=1}^{R} \omega_i^r \frac{\partial \mathcal{L}}{\partial h_i^r} \boldsymbol{W}^g + (1 - \omega_i^r) \frac{\partial \mathcal{L}}{\partial h_i^r} \boldsymbol{W}^r. \tag{2.7}$$

It clearly shows confusion matrices reshape the gradients, which informs the classifier layer what the true label should be on an instance given its noisy annotations. The importance of each confusion matrix in shaping the classifier is determined by ω_i^r, which infers the source of noise based on annotator expertise and instance difficulty.

The gradients in Eq (2.7) also suggest a potential bottleneck of our proposed solution: if the common and individual noise adaptation layers are unidentifiable, we cannot correctly attribute the noise, which is the key for our solution to perform according to Theorem 1. To avoid this, we add ℓ_2-norm on the difference between the common and individual noise adaptation layers as a regularization term, to enforce them to be different. This presents our final loss function,

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{W}^g, \boldsymbol{W}^{1:R}, \boldsymbol{W}^a) = &- \frac{1}{N} \sum_{i=1}^{N} \sum_{r=1}^{R} \sum_{j=1}^{C} y_{ij}^r \log p_j(\hat{y}_i^r | \boldsymbol{x}_i) \\ &- \lambda \sum_{r=1}^{R} \|\boldsymbol{W}^g - \boldsymbol{W}^r\|_2 \end{aligned}$$

where λ is a hyper-parameter to control regularization.

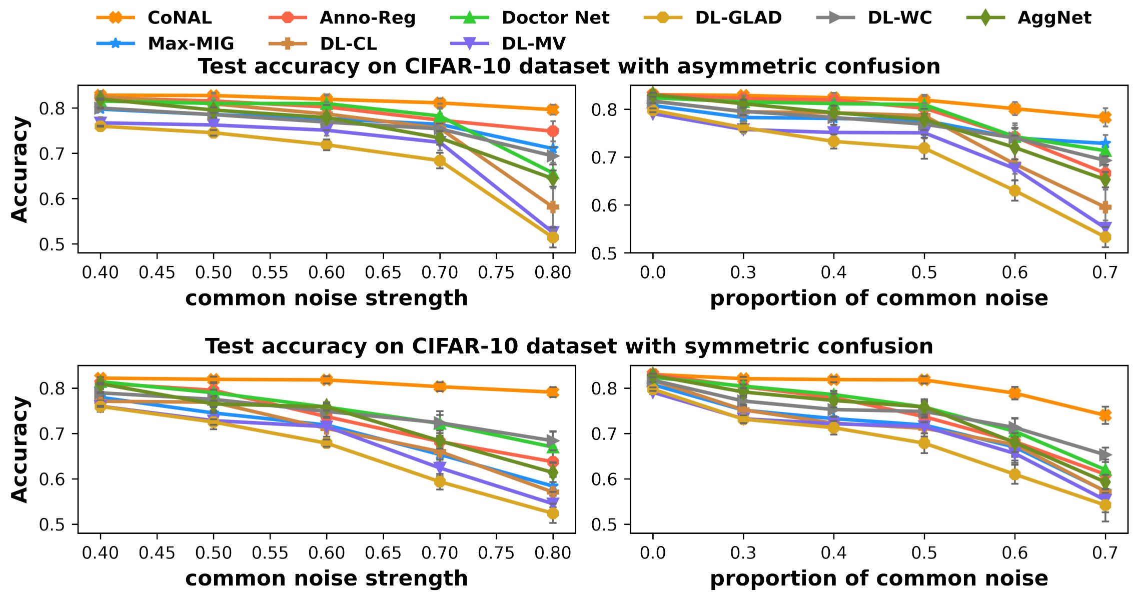


Figure 2.4: Results on CIFAR-10 dataset.

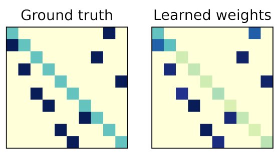
2.1.5 Experiments

We evaluate our method on both synthesized and real-world datasets. We consider a rich set of related solutions as our baselines, which can be divided into two categories:
1) Methods with simple noise models. **DL-MV**: it learns a neural network classifier with labels aggregated by majority voting. **DL-CL** [138]: it learns a neural classifier with designated layers to fit individual annotator confusions (so-called crowd layer). **Anno-Reg** [163]: it improves DL-CL by imposing additional trace regularization on individual confusion matrices. **Doctor Net** [66]: it learns a neural network for every annotator's annotations and aggregates the networks' output by weighted majority voting. **Max-MIG** [20]: it jointly estimates a neural classifier and a label aggregation network using an information-theoretical loss function.
2) Methods with complex noise models. **DL-GLAD**: it learns a neural classifier with labels aggregated by GLAD [185], where annotator ability and instance difficulty are modeled. **DL-WC**: it learns a neural classifier with labels aggregated by WC [86], where similar annotators are clustered to share the same confusion matrix. **AggNet** [2]: an EM-based deep model considering annotator sensitivity and specificity.

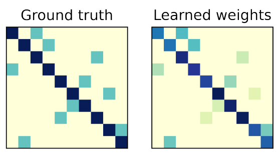
Experiments on synthesized datasets

We evaluate the proposed method under various settings of synthesized data. Particularly, we demonstrate the effectiveness of our model with different (1) *common confusion types*; (2) *common noise strength*, which is defined as the sum of off-diagonal entries in the common confusion matrix; and (3) *proportion of common noise*, which reflects the percentage of annotations introduced by common confusion.
Datasets description. We generate synthesized crowdsourced data on two datasets, where we directly manipulate the number of annotators and annotation generation under a variety of settings. On the **Synthetic** dataset, we completely synthesized everything. We first sample a mean vector for every class and then sample instance features from a multi-variate Gaussian distribution parameterized by this mean vector. In particular, we randomly generate 10,000 instances with 6 classes, which are split into a 8,000-instance training set, a 1,000-instance validation set and a 1,000-instance testing set. The **CIFAR-10** dataset is generated based on the CIFAR-10 image classification dataset [99]. It consists of 60,000 32×32 color images from 10 classes, which are split into a 40,000-instance training set, a 10,000-instance validation set and a 10,000-instance testing set. Image features are used to train the neural classifier on this dataset. In both datasets, each instance in the training set

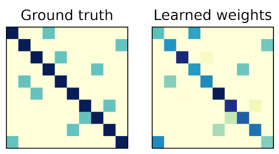
is labeled by averaging 3 randomly selected annotators out of 30 in total.



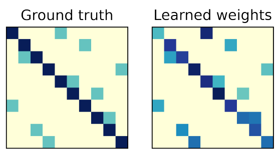
(a) common noise



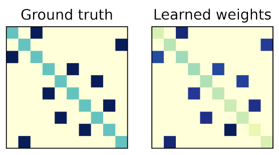
(b) annotator 1



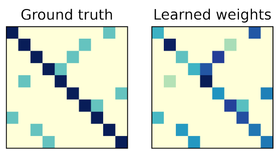
(c) annotator 2



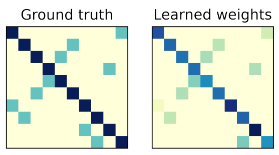
(d) annotator 3



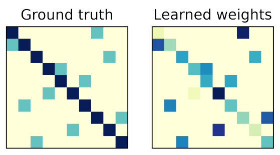
(e) common noise



(f) annotator 1



(g) annotator 2



(h) annotator 3

Figure 2.5: Comparison between ground truth confusion matrices and learned ones on CIFAR-10 dataset. The top row is the result of asymmetric common noise. The bottom row is the result of symmetric common noise.

Synthesizing annotations. We consider two representative noise patterns in common noise: (1) *Asymmetric confusion*. Every class is mapped to another uniformly chosen class on both datasets. (2) *Symmetric confusion*. On Synthetic dataset, two random classes are paired and flipped into each other. And on CIFAR-10 dataset, we manually paired similar classes (e.g., *bird* and *airplane*) to be flipped with each other. For individual confusion matrices, we use asymmetric confusion. We generate one global confusion matrix, and one individual confusion matrix for every annotator. In our experiments, the range of common noise strength is set to $[0.4, 0.8]$, while the individual noise strength of annotators is fixed to 0.7. In both noise generation patterns, the noise strength is evenly distributed among the chosen off-diagonal entries.

To control the source of noise in each annotation, i.e., s_i^r, we randomly generate a set of annotator features $\boldsymbol{u}$, which are not disclosed to the learners. Given instance feature vector $\boldsymbol{v}_i$ and annotator feature vector $\boldsymbol{u}_r$, we compute ω_i^r by Eq (2.6) with the ground-truth weight matrices $(\boldsymbol{W}_u, b_u)$ and $(\boldsymbol{W}_v, b_v)$. These weight matrices are not disclosed to the learner. The bias terms are used to control the average proportion of common noise across annotations into a range of $[0.3, 0.7]$. When we generate annotation y_i^r for instance i by annotator r, we first sample $s_i^r \sim B(\omega_i^r)$. If $s_i^r = 1$, the common confusion matrix π^g will be used; otherwise, individual confusion matrix π^r will be used. Then we sample y_i^r from the chosen confusion matrix based on the true label z_i of this instance. We also include a special case that the proportion is 0, where there is no common confusion.

In our experiments, when studying the influence of common noise strength on the learnt classifier, the average proportion of common noise is controlled to be around 0.5. When studying the influence of the proportion of common noise in each annotation, the common and individual noise strength is controlled to 0.4 and 0.7 respectively.

Backbone networks & training details. On the Synthetic dataset, we apply a simple network with only one fully connected (FC) layer (with 128 units and ReLU activations), along with a softmax output layer, using 50% dropout. On the CIFAR-10 dataset, we follow the setting of Cao et al. [20] to use VGG-16 as the backbone network. We trained the network using the Adam optimizer [96] with default parameters and learning rate searched from {0.02, 0.01, 0.005}. The dimension of annotator and instance embedding is chosen from {20, 40, 60, 80}. The regularization term λ is searched from $\{10^{-4}, 10^{-5}, 10^{-6}\}$. All experiments are repeated 5 times with different random seeds. Model selection is achieved by choosing the model with the highest accuracy on the validation set. We report mean and standard deviation of test accuracy on the five runs. To make the comparisons fair, all the evaluated methods used the same backbone networks. We implement our framework with PyTorch, and run it on a CentOS system with one NVIDIA 2080Ti GPU with 10 GB memory.

Results. We report the results on the CIFAR-10 dataset in Figure 2.4, where our solution demonstrated consistent improvement against all baselines across all settings. All the baselines assumed single source of noise, i.e., annotator-specific noise; as a result, they are heavily influenced when noise become complicated, e.g., a large proportion of mistakes from common confusion and the strength of common noise is strong. Our solution is less sensitive to the environment by decomposing and separately modeling the confusion. When there is no common confusion, the empirical result shows no significant difference between our solution and baselines in this extreme setting, which should also be expected. But we argue that this extreme setting rarely holds in reality, as annotators always share some commonsense about the world.

	DL-MV	DL-CL	Doctor Net	Anno-Reg	Max-MIG	DL-GLAD	DL-WC	AggNet	CoNAL
LabelMe	79.83±0.34	83.27±0.52	82.12±0.43	82.77±0.48	85.33±0.61	83.12±0.34	82.74±0.33	84.75±0.27	**87.12**±0.55
Music	72.53±0.41	81.46±0.53	76.58±0.47	79.12±0.36	81.37±0.33	77.82±0.37	75.76±0.24	81.92±0.41	**84.06**±0.42

Table 2.1: Test accuracy on two real-world crowdsourcing datasets.

All models are influenced by symmetric common noise, which directly makes the swapped classes similar. Based on the lower bound provided in Theorem 1, similar conditional class distributions in the confusion matrices will make the problem more difficult, so that the degeneration of all methods are expected under symmetric confusion. In the most extreme case where the proportion of common noise is set to 0.7 and the common noise strength is set to 0.6, nearly 42% annotations are pairwise flipped. However, our method can still outperform baselines with a large margin. Mix-MIG is believed to be robust to correlated mistakes if high-quality annotator exists. However, our experiments show that common confusion poisoned the classifier obtained in Max-MIG even though every annotator is of high quality (individual noise strength is set to 0.7). DL-CL and Anno-Reg failed because they could not differentiate the source of noise, such that the gradients from the modeled annotations cannot be properly adjusted to update the classifier. Both Doctor Net and DL-MV are based on majority vote, so that they fail when the annotations across annotators are no longer independent, i.e., caused by the common confusion. Compared to methods with complex noise models, DL-GLAD directly models the annotation accuracy, which is not suitable for class-dependent confusion. DL-WC clusters correlated annotators to share confusion matrix, which can reduce the influence of common confusion. But the expertise of each annotator is missing, which leads to its bad performance. AggNet shows the advantage of directly learning from annotations rather than from aggregated labels. But it still assumes the only noise source thus cannot handle common noise well.

To understand how accurate our solution can distinguish common and individual noise, we report the learnt weights of noise adaptation layers against the ground-truth confusion matrices on the CIFAR-10 dataset in Figure 2.5. In this experiment, we set the common noise strength to 0.7 and the proportion of common noise to 0.5. We can find that in most cases the ground-truth common noise pattern is well recovered, especially under the asymmetric noise pattern.

Experiments on real-world datasets

Datasets description. We consider two real-world datasets. **LabelMe** [138, 141] is an image classification dataset, consists of 2,688 images from 8 classes, where 1,000 of them are labeled by annotators from Amazon Mechanical Turk (AMT)[1] and the remainings are used for validation and testing. Each image is labeled by an average of 2.5 annotators, with a mean accuracy of 69.2%. Standard data augmentation techniques are used on training data, including horizontal flips, rescaling and shearing, to enrich the training set to 10,000 images. **Music** [137] is a music genre classification dataset, consisting of 1,000 samples of songs with 30 seconds length from 10 music genres, where 700 of them are labeled by AMT annotators and the rest are used for testing. Each sample is labeled by an average of 4.2 annotators, with a mean annotation accuracy of 73.2%.

[1]https://www.mturk.com/

Backbone networks & training details. For LabelMe dataset, we followed the setting of Rodrigues and Pereira [138]: we apply a pre-trained VGG-16 network followed by a FC layer with 128 units and ReLU activations, and a softmax output layer, using 50% dropout. For Music dataset, we use the same FC layer and softmax layer as LabelMe. Batch normalization [87] is performed in each layer. Other hyper-parameters are the same as the synthesized experiments.

Results. As reported in Table 2.1, CoNAL achieved new state-of-the-art performance on both real-world datasets. In particular, we looked into the accuracy on classes where commonly made mistakes across annotators are observed (see Figure 2.1). For example, for *open country* on LabelMe, its accuracy in CoNAL is 67.21%, while the best baseline Max-MIG only achieved 54.19%. The good performance aligns with our analysis in Theorem 1, by differentiating common and individual confusions, it is easier to find the true labels.

Influence of the regularization term λ. We studied the influence of different λ in Table 2.2. The results show by enforcing the noise adaptation layers to be different, the performance is improved on both datasets. The value of λ also matters, and 10^{-5} achieves best performance empirically.

λ	0	10^{-4}	10^{-5}	10^{-6}
LabelMe	85.68 ± 0.38	86.61 ± 0.41	87.12 ± 0.55	86.26 ± 0.47
Music	82.14 ± 0.31	83.52 ± 0.25	84.06 ± 0.42	82.98 ± 0.37

Table 2.2: Model performance under different λ.

2.2 Mitigating sparsity issue in crowdsourcing via generative augmentation

2.2.1 Introduction

Modern machine learning systems are data hungry, especially for labeled data, which unfortunately is expensive to acquire at scale. Crowdsourcing provides a label collection schema that is both cost- and time-efficient [15]. It spurs the growing research efforts in directly learning a classifier with only crowdsourced annotations, aka the *learning from crowds* problem.

In practice, to minimize annotation cost, the instances in crowdsourced data are typically labeled by a small number of annotators; and each annotator will only be assigned to a few instances. This introduces serious sparsity in crowdsourced data. We looked into two widely-used public crowdsourced datasets for multi-class classification, one for image labeling (referred to as LabelMe [138, 141]) and one for music genre classification (referred to as Music [137]). On the LabelMe dataset, each instance is only labeled by 2.5 annotators on average (out of 59 annotators), while 88% annotators provide less than 100 annotations (out of 1,000 instances). On the Music dataset, each instance is labeled by 4.2 annotators on average (out of 44 annotators), while 87.5% annotators provide less than 100 annotations (out of 700 instances). Such severe sparsity hinders the utility of crowdsourced labels. On the instance side, annotations provided by non-experts are noisy, which are expected to be improved by redundant annotations. But subject to the budget constraint, redundancy is also to be minimized. This conflict directly limits the quality of crowdsourced labels. On the annotator side, most existing crowdsourcing algorithms model annotator-specific confusions, which are used for label aggregation [43], task assignment [46, 107] and annotator education [152]. But due to the limited observations per annotator, such modeling can hardly be inaccurate, and thus various approximations (e.g., strong independence assumptions [43]) have to be devised.

A straightforward solution to address annotation sparsity is to recruit more annotators or increase their assignments, at the cost of an increasing budget. This however is against the goal of crowdsourcing, i.e., to collect labeled data at a low cost. We approach the problem from a different perspective: we perform data augmentation using generative models to fill in the missing annotations. Instead of collecting more real annotations, we generate annotations by modeling the annotation distribution on instances and annotators. Given our end goal is to obtain an accurate classifier, the key is to figure out *what annotations best help the classifier's training*. We propose two important criteria. First, the generated annotations should follow the distribution of authentic ones, such that they will be consistent with the label confusion patterns observed in the original annotations. Second, the generated annotations should well align with the ground-truth labels, e.g., with high mutual information [73, 193], so that they will be informative about ground-truth labels to the classifier.

We realize our criteria for annotation augmentation in crowdsourced data using Generative Adversarial Networks (GAN) [62]. The end product of our solution is a classifier, which predicts the label of a given instance. We set a discriminative model to judge whether an annotation is authentic or generated. Meanwhile, a generative model aims to generate annotations following the distribution of authentic annotations under the guidance of the discriminative model. On a given instance, the generator takes the classifier's output and the annotator and instance features as input to generate the corresponding annotation. To ensure the informativeness of generated annotations, we maximize the mutual information between the classifier's predicted label and the generated annotation on each instance [27]. A two-step training strategy is proposed to avoid model collapse. We name our framework as *CrowdInG* - learning with **Crowd**sourced data through **In**formative **G**enerative augmentation. Extensive experiments on three real-world datasets demonstrated the feasibility of data augmentation for the problem of learning from crowds. Our solution outperformed a set of state-of-the-art crowdsourcing algorithms; and its advantage becomes especially evident with extremely sparse annotations. It provides a new opportunity for low-budget crowdsourcing in general.

2.2.2 Related works

Our work studies the learning from crowds problem. Raykar et al. [133] employed an EM algorithm to jointly estimate the expertise of different annotators and a logistic regression classifier on crowdsourced data. They followed the well-known Dawid and Skene (DS) model [43] to model the observed annotations. Albarqouni et al. [2] extended this solution by replacing the logistic classifier with a deep neural network classifier. Rodrigues and Pereira [138] further extended the solution by replacing the confusion matrix in the DS model with a neural network to model annotators' expertise, and trained the model in an end-to-end manner. Guan et al. [66] used a neural classifier to model each annotator, and aggregated the predictions from the classifiers by a weighted majority vote. Cao et al. [21] proposed an information-theoretical deep learning solution to handle the correlated mistakes across annotators. However, all the mentioned solutions only use the observed annotations, such that their practical performance is limited by the sparsity of annotations.

Another research line focuses on modeling the annotators. Whitehill et al. [185] proposed a probabilistic model which considers both annotator accuracy and instance difficulty. Rodrigues, Pereira, and Ribeiro [137] modeled the annotation process by a Gaussian process. Imamura, Sato, and Sugiyama [86] and Venanzi et al. [173] extended the DS model by sharing the confusion matrices among similar annotators to improve annotator modeling with limited observations. Confusions of annotators with few annotations are hard to be modeled accurately, and Kamar, Kapoor, and Horvitz [90] proposed to address the issue with a shared global confusion matrix. Chu, Ma, and Wang [33] also set a global confusion matrix, which is used to capture the common confusions beyond individual ones. However, the success of the aforementioned models relies on the assumed structures among annotators or annotations. Such strong assumptions are needed, because the sparsity in the annotations does not support more complicated models. But they also restrict the modeling of crowdsourced data, e.g., introducing bias in the learnt model. We lift such restrictions by directly generating annotations, such that our modeling of crowdsourced data even does not make any class- or annotator-dependent assumptions.

Benefiting from their powerful modeling capabilities, deep generative models have been popularly used for data augmentation purposes. Most efforts have been spent on problems in a continuous space, such as image and video generations. Semi-supervised GAN [3, 126, 154] augments training data by generating new instances from labeled ones. Chae et al. [22] employed GAN to address the data sparsity in content recommendation, with their proposed real-value, vector-wise recommendation model training. Recently, GAN has also been adopted in data augmentation for discrete problems. Wang et al. [180] designed a two-step solution to perform GAN training for collaborative filtering. Wang et al. [176] unified generative and discriminative graph neural networks in a GAN framework to enhance the graph representation learning. Irissappane et al. [88] reduced the needed labeled data to fine-tune the BERT-like text classification models via GAN-generated examples.

2.2.3 Background

Generative adversarial networks [62] introduced the GAN framework for training deep generative models as a *minimax* game, whose goal is to learn a generative distribution $P_G(x)$ that aligns with the real data distribution $P_{\text{true}}(x)$. The generative distribution is imposed by a generative model G, which transforms a noise variable $\boldsymbol{\varepsilon} \sim P_{noise}(\boldsymbol{\varepsilon})$ into a sample $G(\boldsymbol{\varepsilon})$. A discriminative model D is set to distinguish between the authentic and generated samples. The generator G is trained by playing against the discriminator D. Formally, G and D play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim P_{\text{true}}}[\phi\big(D(x)\big)] + \mathbb{E}_{\boldsymbol{\varepsilon}\sim P_{\text{noise}}}[\phi\big(1 - D(G(\boldsymbol{\varepsilon}))\big)],$$

where ϕ is a function of choice and $\log(\cdot)$ is typically the choice. The optimal parameters of the generator and the discriminator can be learned by alternately maximizing and minimizing the value function $V(G, D)$. In this paper, we adopt this idea to model the annotation distribution: a generator is used to generate annotations on specific instances and annotators; and a discriminator is set to distinguish the authentic annotations from the generated ones.

Information maximizing generative adversarial networks [27] extended GAN with an information-theoretic loss to learn disentangled representations for improved data generation. Aside from the value function $V(G, D)$, InfoGAN also maximizes the mutual information between a small subset of latent variables (referred to as latent code z) and the

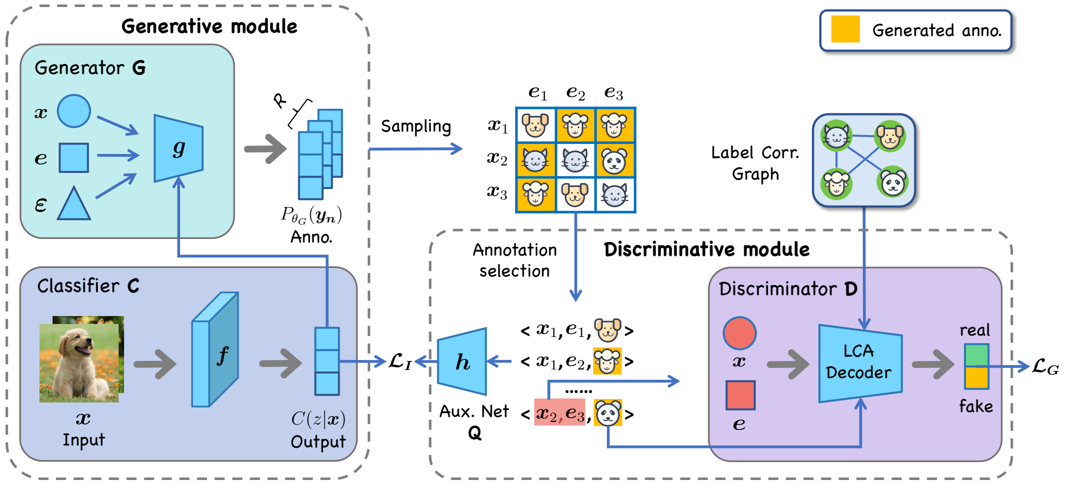


Figure 2.6: Overview of CrowdInG framework. We first sample annotations from annotation distributions provided by the generator. The discriminator and the auxiliary network are trained on the selected annotations. Then, the classifier is first fixed and the generator is updated according to $\mathcal{L}_G$ and $\mathcal{L}_I$. The generator is fixed and the classifier is updated according to $\mathcal{L}_G$.

generated data. The generator takes both random noise ε and latent code z as input, where the latent code is expected to capture the salient structure in the data distribution. The minimax game then turns into an information-regularized form,

$$\min_G \max_D V_I(G, D) = V(G, D) - \lambda I(z; G(\varepsilon, z)),$$

where $I(x; y)$ is the mutual information between random variables x and y, and λ is the regularization coefficient.

2.2.4 The CrowdInG framework

Let $\mathcal{S} = \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$ denote a set of N instances labeled by R annotators out of $|\mathcal{C}|$ possible classes. We define $\boldsymbol{x}_n \in \mathbb{R}^d$ as the feature vector of the n-th instance and $y_n^r \in \mathcal{C}$ as its annotation provide by the r-th annotator. $\boldsymbol{y}_n$ is thus the annotation vector (with missing values) from R annotators for the n-th instance. When available, the feature vector of the r-th annotator is denoted as $\boldsymbol{e}_r$; otherwise, we use a one-hot vector to represent an annotator. Each instance is associated with an unobserved ground-truth label $z \in \mathcal{C}$. The goal of learning from crowds is to obtain a classifier $C(z|\boldsymbol{x})$ that is directly estimated from $\mathcal{S}$.

The framework of CrowdInG is depicted in Figure 3.12. It consists of two main components: 1) a generative module, including a classifier and a generator; and 2) a discriminative module, including a discriminator and an auxiliary network. In the generative module, the classifier first takes an instance $\boldsymbol{x}_n$ as input and outputs its predicted label distribution $P_{\theta_C}(z_n|\boldsymbol{x}_n)$. For simplicity, we collectively denote classifier's output for an instance $\boldsymbol{x}_n$ as $\hat{\boldsymbol{z}}_n$. And then the generator takes the instance $\boldsymbol{x}_n$, annotator $\boldsymbol{e}_r$, the classifier's output $\hat{\boldsymbol{z}}_n$, together with a random noise vector $\boldsymbol{\varepsilon}$, to generate the corresponding annotation distribution $P_{\theta_G}(y_n^r|\boldsymbol{x}_n, \boldsymbol{e}_r, \hat{\boldsymbol{z}}_n, \boldsymbol{\varepsilon})$. The discriminative module is designed based on our criteria of high-quality annotations to evaluate the generations. On one hand, the discriminative module uses a discriminator to differentiate whether the annotation triplet $(\boldsymbol{x}_n, \boldsymbol{e}_r, y_n^r)$ is authentic or generated. On the other hand, the discriminative module penalizes the generation based on the mutual information between the generated annotation and classifier's output measured by an auxiliary network. Following the idea of InfoGAN, we treat the classifier's output $\hat{\boldsymbol{z}}$ as the latent code in our annotation generation. And the auxiliary network measures the mutual information between $\hat{\boldsymbol{z}}$ and y. The two modules play a minimax game in CrowdInG. A better classifier is expected as the discriminative module faces more difficulties in recognizing the generated annotations during training.

Generative module. The output of the generative module is an annotation distribution for a given annotator-instance pair $(\boldsymbol{x}_n, \boldsymbol{e}_r)$. Sampling is applied to obtain the final annotations. As shown in Figure 3.12, this is a two-step procedure. First, the classifier $C(z_n|\boldsymbol{x}_n; \theta_C)$ predicts the label of a given instance $\boldsymbol{x}_n$ by

$$P_{\theta_C}(z_n = c|\boldsymbol{x}_n) \propto \exp[f(\boldsymbol{x}_n, z_n = c)],$$

where $f(\cdot)$ is a learnable scoring function chosen according to the specific classification tasks. Then the generator G takes the classifier's output $\hat{\boldsymbol{z}}$ as input to predict the underlying annotation distribution for the given annotator-instance pair. Moving beyond the classical class-dependent annotation confusion assumption [43, 138], we impose a much more relaxed generative process about the annotations. We consider the confusions can be caused by instance difficulty, or annotator expertise, or true labels of the instances (e.g., different annotation difficulty in different label categories), or even some random noise. To realize the idea, we provide the feature vector $\boldsymbol{x}_n$ of the instance, the annotator $\boldsymbol{e}_r$ and the classifier's output $\hat{\boldsymbol{z}}_n$ to the generator as input, and the corresponding annotation distribution is modeled as,

$$P_{\theta_G}(y_n^r = c|\boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n) \propto \exp[g(y_n^r = c, \boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n)], \tag{2.8}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, 1)$ is a random noise vector, $g(\cdot)$ is a learnable scoring function implemented via a neural network. The generated annotations are sampled from the resulting distribution P_{θ_G}. To simplify our notations, we use $G(\boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n)$ to represent the predicted annotation distribution; and when no ambiguity is invoked, we denote $G(y_n^r)$ as its c-th entry when $y_n^r = c$. Thanks to our data augmentation framework, we can afford a more flexible modeling of the annotation noise, e.g., dropping the hard independence assumptions made in previous works [43, 138]. This in turn helps us boost the quality of generated annotations.

Discriminative module. We realize our principles of high-quality annotations in the discriminative module. First, the discriminator D aims to differentiate whether an annotation y_n^r is authentic from annotator $\boldsymbol{e}_r$ to instance $\boldsymbol{x}_n$, i.e., $D(y_n^r|\boldsymbol{x}_n, \boldsymbol{e}_r; \theta_D)$ predicts the probability of annotation y_n^r being authentic. In a crowdsourcing task, an annotator might confuse a ground-truth label with several classes, such that all of the confused classes could be independently authentic. For example, if an annotator always confuses "birds" with "airplanes" in low resolution images, his/her annotations might be random between these two categories. And thus both types of annotations should be considered as valid, as there is no way to tell which annotation is "correct" only based on the observations of his/her annotations. As a result, we realize the discriminator as a multi-label classifier, which takes an annotation triplet $(\boldsymbol{x}_n, \boldsymbol{e}_r, y_n^r)$ as input and calculates the discriminative score by a bilinear model,

$$D(y_n^r = c|\boldsymbol{x}_n, \boldsymbol{e}_r; \theta_D) = \sigma(\boldsymbol{u}_r^\top \boldsymbol{M}_c \boldsymbol{v}_n), \tag{2.9}$$

$$\boldsymbol{u}_r = \boldsymbol{W}_u \boldsymbol{e}_r + b_u, \boldsymbol{v}_n = \boldsymbol{W}_v \boldsymbol{x}_n + b_v,$$

where $\sigma(\cdot)$ is the sigmoid function, $\boldsymbol{M}_c$ is the weight matrix for class c, $(\boldsymbol{W}_v, b_v)$ and $(\boldsymbol{W}_u, b_u)$ are weight matrices and bias terms for annotator and instance embedding layers. For simplicity, we denote $D(y_n^r)$ as the discriminator's output on annotation y_n^r.

However, Eq (2.9) does not consider the correlation among different classes in the annotations, as it still evaluates each possible label independently. The situation becomes even worse with sparse observations in individual annotators. For example, when an annotator confuses "bird" with "airplanes", the discriminator might decide the label of "bird" is more authentic for this annotator, simply because this category appears more often in the annotator's observed annotations. To capture such "*equally plausible*" annotations, we equip the discriminator with additional label correlation information [101]. Specifically, we use a graph convolution network (GCN) [98] to model label correlation. Two labels are more likely to be correlated if they are provided to the same instance (by different annotators) in the authentic annotations. We calculate the frequency of label co-occurrence in the observed annotations to construct the adjacency matrix $\boldsymbol{A}$. Then we extend the weight matrix $\boldsymbol{M}_c$ in Eq (2.9) by $\hat{\boldsymbol{M}}_c = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{M}_c\boldsymbol{W}$, with $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix, $\hat{D}$ is the diagonal node degree matrix of $\hat{\boldsymbol{A}}$. We name this component as the label correlation aggregation (LCA) decoder. We also enforce sparsity on the discriminator by applying L2 norm to its outputs.

To realize our second criterion, an auxiliary network Q is used to measure the mutual information between the classifier's prediction $\hat{\boldsymbol{z}}_n$ and the generated annotation y_n^r on instance $\boldsymbol{x}_n$. To simplify our notations in the subsequent discussions, we denote $G(\boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n)$ as $G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}})$ to represent the annotation distribution predicted on pair $(\boldsymbol{x}_n, \boldsymbol{e}_r)$. As our generator design is very flexible to model complex confusions, it however becomes useless for classifier training if the learnt confusions are independent from the classifier's outputs. For example, if the generator learnt to generate a particular annotation only by the annotator's features (e.g., the most frequently observed label in this annotator), such a generation contributes no information to classifier training. We propose to penalize such generations by maximizing the mutual information between classifier's output and the generated annotations for an instance, i.e., $I(\hat{\boldsymbol{z}}; G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}))$.

In practice, mutual information is generally difficult to optimize, because it requires the knowledge of posterior $P(\hat{z}|y)$. We follow the design in [27] to maximize the variational lower bound of $I(\hat{\boldsymbol{z}}; G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}))$ by utilizing an auxiliary distribution P_Q to approximate $P(\hat{z}|y)$:

$$\begin{aligned}\mathcal{L}_{\mathcal{I}}(G, Q) &= \mathbb{E}_{\hat{z}\sim P(\hat{\boldsymbol{z}}), y\sim G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}})}[\log P_Q(\hat{z}|y)] + H(\hat{z}) \\ &\leq I(\hat{\boldsymbol{z}}; G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}})).\end{aligned} \tag{2.10}$$

We refer to $\mathcal{L}_I$ as the information loss, which can be viewed as an information-theoretical regularization to the original minimax game. The auxiliary distribution $P_Q(\hat{z}|y)$ is parameterized by the auxiliary network Q. In our implementation, we devise a two-step training strategy for the entire pipeline (details in Section 2.2.5), where we fix the classifier when updating the generator. As a result, $H(\hat{z})$ becomes a constant when updating the generator by Eq (2.10). Since the posterior $P(\hat{z}|y)$ can be different when the annotations are given by different annotators on different instances, we also provide the instance and annotator features to the auxiliary network,

$$P_{\theta_Q}(\hat{z}_n = c|\boldsymbol{x}_n, \boldsymbol{e}_r, y_n^r) \propto \exp[h(\hat{z}_n = c, \boldsymbol{x}_n, \boldsymbol{e}_r, y_n^r)],$$

where $h(\cdot)$ is a learnable scoring function. To reduce model complexity, we reuse the annotator and instance encoding layers from the discriminator here. The class-related weight matrix $\hat{\boldsymbol{M}}_c$ is flatten and transformed to a low-dimension embedding $\boldsymbol{m}_c$ by an embedding layer for each annotation type $y_n^r = c$.

Putting the generative and discriminative modules together, we formalize the value function of our minimax game for learning from crowds in CrowdInG as,

$$\min_{C,G,Q} \max_{D} V_{\text{CrowdInG}}(C, G, D, Q) = V(C, G, D) - \lambda \mathcal{L}_I(G, Q) \tag{2.11}$$
$$V(C, G, D) = \mathbb{E}_{y \sim P_{\text{true}}}[\log\big(D(y)\big)] + \mathbb{E}_{\boldsymbol{\varepsilon} \sim P_{\text{noise}}, y \sim P_{G(\boldsymbol{\varepsilon}, \hat{\boldsymbol{z}})}}[\log\big(1 - D(y)\big)],$$

where λ is a hyper-parameter to control the regularization. The value function is maximized by updating the discriminator to improve its ability in differentiating the authentic annotations from the generated ones, and minimized by updating the classifier, generator and the auxiliary network to generate more high-quality annotations.

2.2.5 Model optimization

In this section, we introduce the training strategy for CrowdInG, which cannot be simply performed via vanilla end-to-end training. First, the number of unobserved annotator-instance pairs is much larger than the observed ones. Blindly using all the generated annotations overwhelms the training of our discriminative module, and simply leads to trivial solutions (e.g., classifying all annotations as generated). As our solution, we present an entropy-based annotation selection strategy to select informative annotations for discriminative module update. Second, due to the required sampling procedure when generating the annotations, there are non-differentiable steps in our generative module. We resort to an effective counterfactual risk minimization (CRM) method to address the difficulty. Finally, the classifier and the generator in the generative module might change dramatically to fit the complex training signals, which can easily cause model collapse. We propose a two-step training strategy to prevent it in practice.

Entropy-based annotation selection. We borrow the idea from active learning [146]: *the discriminator should learn to distinguish the most difficult annotations.* A generated annotation is more difficult for the discriminator if the generator is more confident about it. Formally, the selection strategy is designed as,

$$P(y_n^r) \propto \frac{1}{H(G(\boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n))},$$

where $H(G(\boldsymbol{x}_n, \boldsymbol{e}_r, \boldsymbol{\varepsilon}, \hat{\boldsymbol{z}}_n))$ is the entropy of the annotation distribution. To reduce training bias caused by annotation sparsity in individual annotators, we sample the same number of generated annotations as the authentic ones in each annotator. As a by-product, our instance selection also greatly reduces the size of training data for the discriminative module. It makes discriminator training a lot more efficient. To fully utilize the power of discriminative module, we use all generated annotations for the generator updating.

Gradient-based optimization. The gradient for the discriminator and the auxiliary network is easy to compute by calculating the derivative on trainable parameters. However, due to the required sampling steps for generating specific annotations, there are non-differentiable steps in the generative module. Previous works [176, 180] use Gumbel-softmax trick or policy gradient to handle the non-differentiable functions. However, once the generator is updated, we need to re-sample the annotations and evaluate them again using the discriminative module, which is time-consuming. To accelerate our model training, we perform batch learning from logged bandit feedback [89, 161]. In each epoch, we treat the generative module from the last epoch as the logging policy G_0, and sample annotations from it. Because the discriminator only evaluates the sampled annotations from the (last) generative module, rather than the entire distribution of annotations predicted by the module, training signals received on the generative module side are in the form of logged bandit feedback.

When updating the generator, the training signals are from both the discriminator $\mathcal{L}_G = \log\big(1 - D(y)\big)$ and the information loss $-\lambda\mathcal{L}_I$. We collectively denote them as loss $\delta = \mathcal{L}_G - \lambda\mathcal{L}_I$. In each epoch, we update the generator G_{θ_G} as follows,

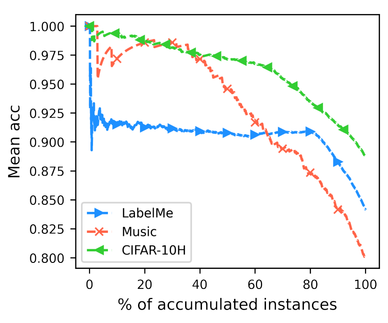
$$\theta_G = \underset{\theta_G}{\text{argmin}} \frac{1}{NR}\sum_{n=1}^{N}\sum_{r=1}^{R}\frac{\big(\delta(y_n^r) - \mu\big)G_{\theta_G}(y_n^r)}{G_0(y_n^r)}, \tag{2.12}$$

where μ is a Lagrange multiplier introduced to avoid overfitting to the logging policy [89]. The optimization of Eq (2.12) can be easily solved by gradient descent. When updating the classifier, we only use the discriminator's signals. Intuitively, even though annotations should contain the information about the true labels, the inverse is not necessary. The classifier is updated in a similar fashion,

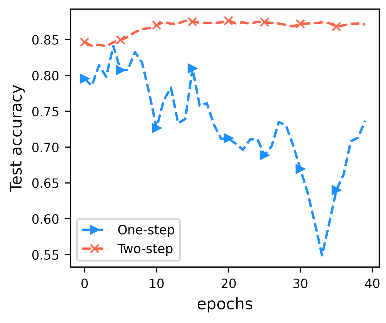
$$\theta_C = \underset{\theta_C}{\text{argmin}} \frac{1}{NR}\sum_{n=1}^{N}\sum_{r=1}^{R}\frac{\big(\mathcal{L}_G(y_n^r) - \mu\big)G_{\theta_C}(y_n^r)}{G_0(y_n^r)}. \tag{2.13}$$

We follow the suggestions in [89] to search the best μ in practice.

Two-step update for the generative module. The generative process is controlled by the generator and the classifier. However, the coupling between the two components introduces



(a)



(b)

Figure 2.7: Performance of two-step strategy. (a) Mean accuracy of accumulated instances with ascending order of entropy on three real-world datasets. (b) Comparison between one-step and two-step strategy on LabelMe dataset.

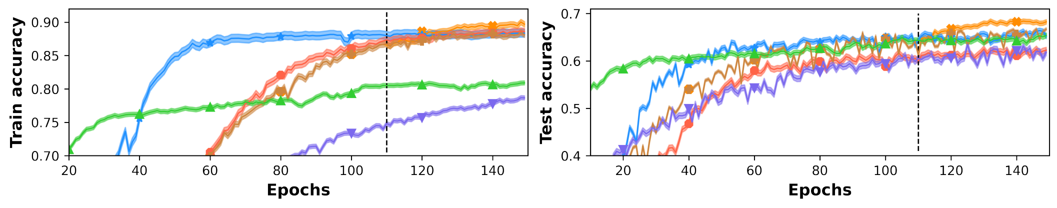
difficulties in the estimation of them. For example, one component might overfit a particular pattern in the discriminative signal, and cause model collapse in the entire pipeline. In our empirical studies reported in Figure 2.7(b), we observed test accuracy fluctuated a lot when we simply used gradients calculated by Eq (2.12) and (2.13) to update these two components together.

Based on this finding, we adopt a two-step strategy to update the generator and the classifier alternatively. First, we found that the principle behind our annotation selection also applied to our classifier: the entropy of the classifier's output strongly correlates with its accuracy. According to Figure 2.7(a), the classifier obtains higher accuracy on instances with lower prediction entropy. Therefore, we decided to use the instances with low classification entropy to update the generator by Eq (2.12), as there the classifier's predictions are more likely to be accurate. Then, we use the updated generator on the rest of instances to update the classifier by Eq (2.13), where the classifier still has a high uncertainty to handle them.

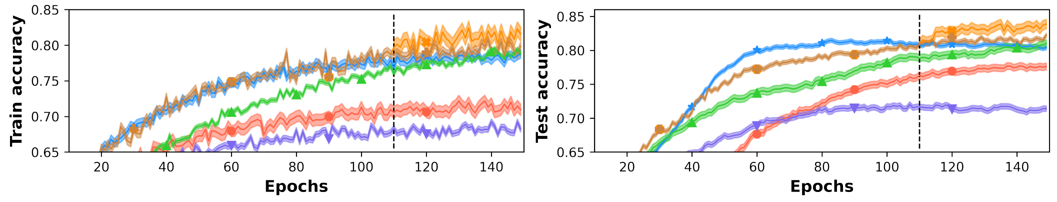
A threshold t is pre-selected to separate the instances; and we will discuss its influence on model training in Section 2.2.6. Besides, to make the entire training process stable, we pre-train the classifier with the observed annotations using neural crowdsourcing algorithm proposed in [138], which is included as one of our baselines. With the initialized classifier, we also pre-train the generator and discriminator to provide good initialization of these components.

2.2.6 Experiments

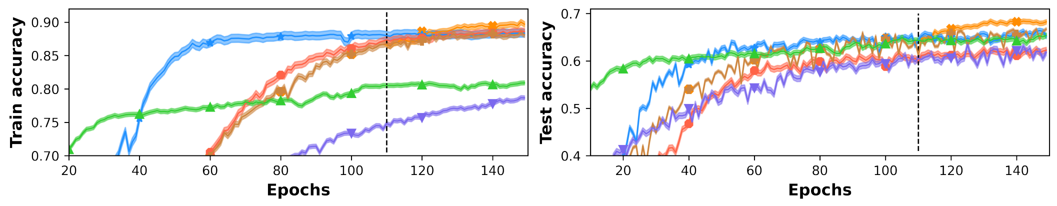
In this section, we evaluate our proposed solution framework on three real-world datasets. The annotations were originally collected from Amazon Mechanical Turk (AMT) by the



(a) Results on LabelMe dataset.



(b) Results on Music dataset.



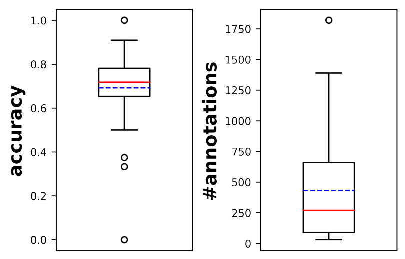
(c) Results on CIFAR-10H dataset.

Figure 2.8: Results on three real-world datasets. Full CrowdInG training is applied after the dashed line.

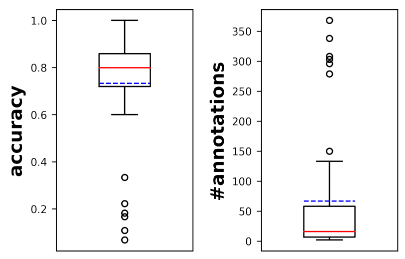
dataset creators. We compared with a rich set of state-of-the-art crowdsourcing algorithms that estimate the classifiers only with observed annotations. We are particularly interested in investigating *how much human labor can be saved by our data augmentation solution*? We gradually removed an increasing number of annotations and compared with baselines. The result suggests significant annotation cost can be reduced with our generated annotations, while still maintaining the quality of the learnt classifier. Besides, since our model is the first effort to augment crowdsourced data for classifier training, we compared with models trained with annotations from other generative models for crowdsourced data. Finally, we performed extensive ablation analysis about our proposed model components and hyper-parameters to better understand the model's behavior.

Main results

Datasets & implementation details. We employed three real-world datasets for evaluations. **LabelMe** [138, 141] is an image classification dataset, which consists of 2,688 images from 8 classes, e.g., *inside city*, *street*, *forest*, etc. 1,000 of them are labeled by 59 AMT annotators and the rest are used for validation and testing. Each image is labeled by 2.5 annotators on average. To enrich the training set, standard data augmentation techniques are applied on the training set, including horizontal flips, rescaling and shearing, following the setting in [138]. We created 10,000 images for training eventually. **Music** [137] is a music genre classification dataset, which consists of 1,000 samples of songs with 30 seconds in length from 10 classes, e.g., *classical*, *country*, *jazz*, etc. 700 of them are labeled by 44 AMT annotators and the rest are left for testing. Each sample is labeled by 4.2 annotators on average. Figure 2.9 shows several important statistics of these two datasets. Specifically, we report the annotation accuracy and the number of annotations among the annotators. Both statistics vary considerably across annotators in these two datasets, which cause serious difficulties in classical crowdsourcing algorithms. **CIFAR-10H** [129] is another image classification dataset, which consists of 10,000 images from 10 classes, e.g., *airplane*, *bird*, *cat*, etc., collected from the CIFAR-10 image dataset [99]. There were 2,571 annotators recruited and each annotator was asked to label 200 images. However, such large-scale annotations are typically expensive and rare in practice. To make this dataset closer to a realistic and challenging setting, we only selected a subset of low-quality annotators. The modified dataset has 8,687 images annotated by 103 AMT annotators. Each annotator still has 200 annotations with an average accuracy of 78.2%; and each image has 2.37 annotations on average. The original 10,000 images validation set of CIFAR-10 is used as our testing set.



(a) LabelMe



(b) Music

Figure 2.9: Boxplots for the number of annotations and the accuracy of the AMT annotators for two real-world crowdsourcing datasets.

To make the comparisons fair, all evaluated methods used the same classifier design (in both CrowdInG and baselines). On the LabelMe dataset, we adopted the same setting as in [138]: we applied a pre-trained VGG-16 network followed by a fully connected (FC) layer with 128 units and ReLU activations, and a softmax output layer, using 50% dropout. On the Music dataset, we also used a 128 units FC layer and softmax output layer. Batch normalization was performed in each layer. We disabled LCA on Music since there is no meaningful label correlation patterns. On the CIFAR-10H dataset, we used a VGG-16

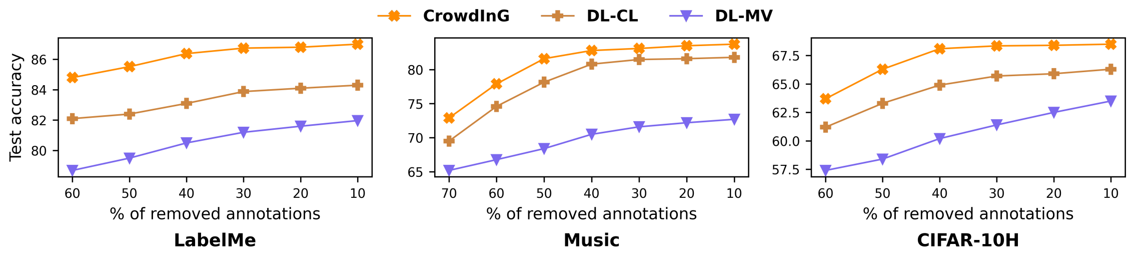


Figure 2.10: Test accuracy with various proportion of removed annotations.

network for the classifier. Scoring functions $g(\cdot)$ and $h(\cdot)$ are implemented by two-layer neural networks with 64 and 128 hidden units. In each epoch, we update the generative and discriminative modules for 5 times. With pre-training, we execute the training procedures for CrowdInG in the last 40 epochs. All experiments are repeated 5 times with different random seeds, and mean accuracy and standard derivation are reported.

Table 2.3: Test accuracy of different augmentation methods.

	LabelMe	Music	CIFAR-10H
Doctor Net	82.12±0.43	75.41±0.42	67.23±0.54
DL-CL$_{\text{+Self}}$	85.24±0.51	82.56±0.49	64.94±0.84
DL-CL$_{\text{+GCN}}$	82.74±0.34	81.42±0.74	65.02±0.61
DL-CL$_{\text{+GAN}}$	85.16±0.26	83.17±0.48	65.34±0.32
DL-CL$_{\text{+InG}}$	85.42±0.57	83.38±0.59	66.17±0.35
CrowdInG	**87.03**±0.55	**83.73**±0.62	**68.85**±0.47

Baselines. We compared with a rich set of state-of-the-art baselines, which we briefly introduce here. **DL-MV**: annotations are first aggregated by majority vote, and then it trains a classifier based on the aggregated labels. **DL-CL** [138]: a set of designated layers that capture annotators' confusions (the so-called Crowd Layer) are connected to the classifier, aiming to transform the predicted classifier's outputs to annotation distributions. **Anno-Reg** [163]: trace regularization on confusion matrices is applied to improve the confusion estimation. **Max-MIG** [21]: a neural classifier and a label aggregation network are jointly trained using an information-theoretical loss function, correlated confusions among annotators are captured. **AggNet** [2]: an EM-based deep model considering annotator sensitivity and specificity.

Results & analysis. The classification accuracy of the learnt classifiers from different models on the three datasets are reported in Figure 2.8. Two things we should highlight: 1) as all models are learnt from crowdsourced data, the ground-truth labels on instances are *unrevealed* to them in training. Therefore, a classifier's accuracy on training set is still a meaningful performance metric. 2) CrowdInG starts with the same classifier as obtained in DL-CL (as we used DL-CL to pre-train our classifier). On all datasets, we observe that even though DL-CL did not outperform the other baselines, after the training in CrowdInG starts, the classifier's performance got significantly improved. This proves the utility of our generated annotations for classifier training. Besides, we also looked into the accuracy in individual classes and found by generating more annotations, CrowdInG's performance on those easily confused classes got more improvement than the baselines. For example, for the class of *open country* on LabelMe, the original annotation accuracy was only 51.5%. DL-CL achieved 49.6% (i.e., the starting point of CrowdInG), and it was improved to 58.9% after CrowdInG training. Compared with models that are designed for complex confusions, such as Max-MIG and AggNet, CrowdInG still outperformed them with a large margin. This indicates our generator has a stronger advantage in capturing complex confusions.

Utility of augmented annotations

Experiment setup. We study the utility of augmented annotations from CrowdInG. On each dataset, we gradually removed an increasing number of observed annotations to investigate how different models' performance changes. We ensure that each instance has at least one annotation, such that we will only remove annotations rather than instances for classifier training. We compared with two representative baselines: 1) DL-MV, a typical majority-vote-based method, and 2) DL-CL, a typical DS-model-based method, to study their sensitivity on the sparsity of annotations.
Results & analysis. We present the results in Figure 2.10. All models suffered from extreme sparsity when we removed a large portion of annotations (e.g., 60%), but CrowdInG still enjoyed a consistent improvement against all baselines. DL-MV performed the worst, because with less redundant annotations, the quality of its aggregated labels deteriorated seriously. When we looked into the detailed model update trace of CrowdInG, we found that the performance gain became larger after CrowdInG training. Again, because we used the classifier obtained from DL-CL as our starting point for CrowdInG, low-quality annotations were generated at the beginning of CrowdInG update. However, CrowdInG quickly improved once its discriminative module started to penalize those low-quality annotations. The results strongly support that a great deal of human labor can be saved. On LabelMe and CIFAR-10H, CrowdInG performed closely to the baselines' best performance even with 60% less annotations. Even on the most difficult dataset Music, about 10% annotations can be saved by CrowdInG to achieve similar performance as DL-CL.

Comparison with other augmentations

Baselines. As no existing method explicitly performs data augmentation for crowdsourced data, we consider several alternative data augmentation methods using various self-training or generative modeling techniques. Arguably, any generative model for crowdsourced data can be used for this purpose. In particular, we chose the following baselines. **Doctor Net** [66]: each annotator is modeled by an individual neural network. When testing, annotations are predicted by annotator networks and then aggregated by weighted majority vote. **DL-CL$_{\text{+Self}}$**: we complete the missing annotations using a pre-trained DL-CL model, and then train another DL-CL model based on the completed annotations. **DL-CL$_{\text{+GCN}}$**: we construct an annotator-instance bipartite graph based on the observed annotations, and fill in the missing links using a Graph Convolution Network (GCN) [10, 98]. Then we train a DL-CL model using the expanded annotations. **DL-CL$_{\text{+GAN}}$**: we follow the same design in [178], which unifies generative and discriminative models into a GAN framework. We use DL-CL as the generative model. **DL-CL$_{\text{+InG}}$**: we directly train a DL-CL model on the expanded dataset provided by CrowdInG.

Results & analysis. We present the test accuracy on all three datasets in Table 2.3. Doctor Net trains individual classifiers for each annotator, so that on datasets where annotations from each annotator are sufficient, such as CIFAR-10H, this model obtained satisfactory performance with the generated annotations. But on the other datasets where annotations are sparse in each annotator, its performance dropped a lot. In DL-CL type methods, the performance is generally improved. However, due to the simple class-dependent confusion assumption, such models' capacity to capture complex confusions is limited. As a result, even though GCN could capture more complex annotator-instance interactions, DL-CL still failed to benefit from it in DL-CL$_{\text{+GCN}}$. The added discriminator in DL-CL$_{\text{+GAN}}$ improved the performance; however, DL-CL still could not fully utilize the complex discriminative signals and failed to further improve the performance. DL-CL$_{\text{+InG}}$ performed better than the other baselines by directly using the annotations generated by CrowdInG, which suggests the annotations generated under our criteria are generically helpful for other crowdsouring algorithms.

Ablation study

Analysis of different components in CrowdInG. To show the contributions of different components in CrowdInG, we varied the setting of our solution. We already showed the one-step training variant in Figure 2.7, which suffered from serious model collapsing. To investigate the other components, we created the following variants. **CrowdG**: the infor-

Table 2.4: Test accuracy of different variants of CrowdInG

	LabelMe	Music	CIFAR-10H
CrowdG	85.89±0.47	83.14±0.28	66.15±0.34
CrowdInG$_\mathrm{U}$	83.12±0.39	81.28±0.51	67.12±0.59
CrowdInG$_\mathrm{I}$	84.34±0.72	82.24±0.47	66.90±0.31
CrowdInG$_\mathrm{R}$	86.17±0.44	82.74±0.58	67.88±0.62
CrowdInG	**87.03**±0.55	**83.73**±0.62	**68.85**±0.47

mation loss defined in Eq (2.10) is removed. **CrowdInG$_\mathrm{U}$**: the generator only considers classifier's outputs, annotator features and random noise, but not the instance features. **CrowdInG$_\mathrm{I}$**: the generator only considers classifier's outputs, instance features and random noise, but not the annotator features. **CrowdInG$_\mathrm{R}$**: the annotation selection is kept, but instead we randomly select an equal number of generated annotations as the authentic ones for discriminator update.

We reported the test accuracy on three datasets in Table 3.6. By maximizing the mutual information, CrowdInG outperformed CrowdG with a considerable margin. We further investigated the generated annotations and found the annotations generated by CrowdG were more random, which could not be easily linked to the classifier's output. CrowdInG$_\mathrm{U}$ performed poorly when the number of annotations per annotator was limited, such as on LabelMe and Music datasets, but worked better when annotations per annotator are adequate, such as on CIFAR-10H. This again proves more annotations are needed to better model annotators' confusions. CrowdInG$_\mathrm{I}$ performed better because by taking instance features, the generator can model more complicated confusions with respect to instance features. CrowdInG$_\mathrm{R}$ bypassed the data imbalance issue; but without focusing on difficult annotations, it still cannot fully unleash the potential of generated annotations.

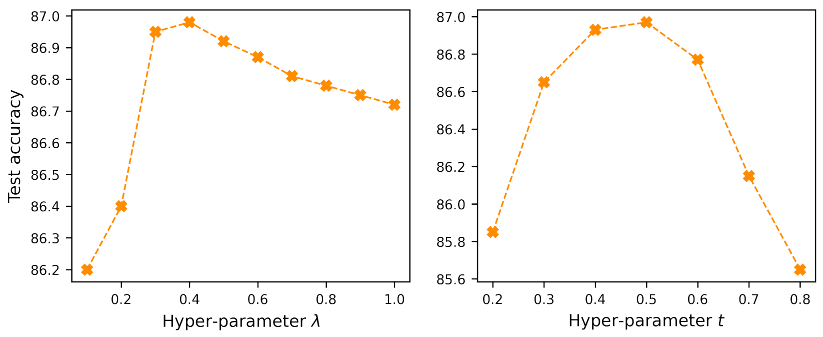


Figure 2.11: Performance under different hyper-parameter settings on LabelMe dataset.

Hyper-parameter analysis. We studied the sensitivity of hyper-parameters λ and t in CrowdInG. Specifically, λ controls the degree of the information regularization in Eq

(2.11), we varied it from 0.1 to 1. t controls the grouping of instances used for classifier update; and we varied it from 0.2 to 0.8. Due to space limit, we only report the results on LabelMe, similar observations were also obtained on the other two datasets.

The model's performance under different hyper-parameter settings is illustrated in Figure 2.11. We can clearly observe that the performance is boosted when appropriate hyper-parameters are chosen. Small λ poses weak information regularization to the generator, and thus the generated annotations are less informative for classifier training. Large λ slightly hurts the performance because strong regularization weakens the ability of the generator to capture complex confusions related to instance and annotator features. We can observe similar trend on t. To avoid model collapse, a moderate t is needed to restrict the classifier training, but a large t will hurt the performance more. Because with a large t, very few instances will be selected for classifier training, so that the classifier can hardly be updated.

2.3 Enhancing low-fidelity LLMs to high-fidelity performance via weak comparisons

2.3.1 Introduction

Large Language Models (LLMs) have shown remarkable capabilities across a wide range of domains that require intricate natural language understanding and reasoning, such as mathematical reasoning/problem solving [40, 183], code generation/programming [5, 25], creative writing [197], etc. A significant step in refining LLMs is the post-pretraining *alignment*, which involves reinforcing behaviors highly rated by human annotators and penalizing behaviors that evaluators rate poorly [31, 59, 127]. There are two mainstream alignment methods: Supervised Fine-Tuning (SFT) [127, 168] based on human-written demonstrations on given prompts, and Reinforcement Learning from Human Feedback (RLHF) [8, 31, 127] based on human preferences on pairs of responses.

The aforementioned alignment methods require a substantial amount of high-quality human annotated data, posing challenges for those with limited budgets and resources to gather such data. This motivates us to study a more realistic setting where human annotators exhibit different levels of *fidelity* (i.e., expertise), and a challenge is naturally arised: *how to improve the LLMs trained with low-fidelity data?* As shown in Figure 2.12, LLMs trained with data from annotators of varying fidelity exhibit differing performance levels. To improve low-fidelity LLMs, we propose to utilize guidance from high-fidelity LLMs. To simplify our discussion, we refer to the fidelity of LLMs as indicative of the fidelity of their training data and we consider a set of LLMs with varing fidelity. To realize our

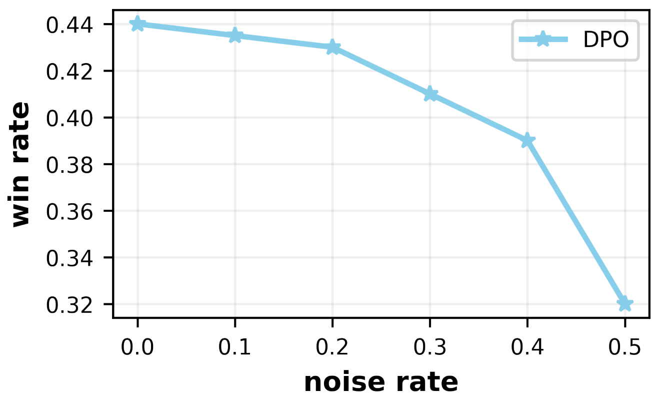


Figure 2.12: Win rates vs. chosen responses on the Anthropic-HH dataset. An `opt-1.3b` model, trained with data of varying fidelity (indicated by noise rates) using the DPO algorithm [130], shows a significant drop in performance on low-fidelity datasets.

idea, we develop a Guided Fine-Tuning (**`GFT`**) framework where we derive guidance by comparing the responses to same prompts from both low-fidelity and high-fidelity models. And this comparative guidance is used as weak supervisions to fine-tune low-fidelity models, enhancing their performance through informed adjustments. Low-fidelity models are anticipated to filter out noise learned from low-fidelity datasets with the guidance provided by high-fidelity models. Experiments on the single-turn dialogue generation task shows the effectiveness and efficiency of the proposed method.

2.3.2 Related works

Alignment from human feedback. Typical alignment methods include Supervised Fine-Tuning (SFT) based on human-written demonstrations and Reinforcement learning from human feedback (RLHF) based on human preferences [127]. The standard RLHF assumes the human preferences follow a Bradley-Terry model [14] for pairwise comparisons or a Plackett-Luce model [67] for multi-wise comparisons, and minimize the cross-entropy loss to learn a reward model [216]. After learning the reward model, RLHF further fine-tunes the language model using RL techniques (e.g., PPO [144]) according to the scores from the learned reward model, aligning the model with human preferences. Rafailov et al. [130] introduce direct policy optimization (DPO) which bypasses the reward model learning stage. It optimizes an implicit reward model derived from the likelihood ratio between chosen responses and rejected responses. Both RLHF and DPO need a reference model to avoid large model updates. Hong, Lee, and Thorne [79] develop a reference model-free method to learn from preference data. However, the aforementioned methods rely on either high-quality human-written demonstrations, human preferences, or a combination of both. In this paper, we propose a framework to improve low-fidelity LLMs through the outputs of high-fidelity LLMs. It requires no additional human labors for writing demonstrations or judging LLMs' responses.

Student-teacher training. In student-teacher training, we first train a teacher model and then train a student model based on teacher's pseudo-labels. This framework is widely used in semi-supervised learning [100, 164], domain adaptation [56, 149], and knowledge distillation [64, 78]. Recently, Burns et al. [16] discuss a specific setting where the student is much more capable than the teacher. And they find the *weak-to-strong generalization* phenomenon where the strong student can surpass the weak teacher in some scenarios. In this work, we study the setting where the teacher model is trained on high-fidelity data, while the student model is trained on low-fidelity data, while the teacher model is not necessary to have more parameters than the student model. We anticipate that the noise present in the student model can be mitigated by comparing the outputs of both the student and teacher models, thereby refining the student model's performance.

2.3.3 Preliminaries

We consider a large language model (LLM) f_θ parameterized by θ. Essentially, f_θ represents a probability distribution over a predefined vocabulary space $\mathcal{V}$. The model takes a sequence $\boldsymbol{x} = [x_1, ..., x_n]$ as input, commonly referred as the prompt, to generate the corresponding response $\boldsymbol{y} = [y_1, ..., y_m]$. The response $\boldsymbol{y}$ can be seen as a sample from the conditional probability distribution $f_\theta(\cdot|\boldsymbol{x})$. The autoregressive model f_θ generate tokens sequentially for a given position, leveraging only the sequence of previously generated tokens. This model there for constitutes a Markov process, where the log-likelihood of generating $\boldsymbol{y}$ can be expressed as follows,

$$\log f_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{j=1}^{m} \log f_\theta(y_j|\boldsymbol{x}, \boldsymbol{y}_{<j}), \tag{2.14}$$

where $\boldsymbol{y}_{<1}$ is null and $\boldsymbol{y}_{<j} = [y_1, ..., y_{j-1}]$ for $j = 2, ..., m$. In the following, we review several mainstream alignment approaches.

Supervised fine-tuning. Supervised fine-tuning (SFT) [127] is employed to tailor a pre-trained LLM to specific downstream tasks, by leveraging a high-quality dataset $\mathcal{D}_{\text{SFT}} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i\in[|\mathcal{D}_{\text{SFT}}|]}$. In this context, $\boldsymbol{y}$ is typically a high-quality response written by human annotators for the corresponding prompt $\boldsymbol{x}$. Consequently, the objective of SFT is to minimize the following negative log-likelihood loss,

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{\text{SFT}}}\Big[\log f_\theta(\boldsymbol{y}|\boldsymbol{x})\Big]. \tag{2.15}$$

After the SFT training, the LLM is expected to produce outputs similar to the responses provided by humans. Thus, if the quality of SFT datasets is not sufficiently high, it becomes a bottleneck for the quality of the trained LLM.

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{\boldsymbol{x},\sim\mathcal{D}_{\text{GFT}},\hat{\boldsymbol{y}}\sim g_\phi(\boldsymbol{x})}\Big[\log f_\theta(\hat{\boldsymbol{y}}|\boldsymbol{x})\Big]. \tag{2.16}$$

Reinforcement learning from human feedback. Reinforcement learning from human feedback (RLHF) [8, 31] offers a method to align pretrained LLMs with human preferences. A reward function $r_\psi(\boldsymbol{x},\boldsymbol{y})$ is essential for RLHF, which is parameterized by ψ. The reward function reflects human preferences for the response $\boldsymbol{y}$ given the prompt $\boldsymbol{x}$, where a higher value indicates a better response. The objective of the RLHF is to maximize the following objective function,

$$\mathcal{L}_{\text{RLHF}}(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{\text{RLHF}}}\Big[r_\psi(\boldsymbol{x},\boldsymbol{y}) - \lambda \text{KL}(f_\theta(\boldsymbol{y}|\boldsymbol{x}) \parallel f_{\theta_{\text{ref}}}(\boldsymbol{y}|\boldsymbol{x}))\Big], \tag{2.17}$$

where the Kullback-Leibler (KL) regularization term enforces the trained model f_θ to be close to the reference model $f_{\theta_{\text{ref}}}$, and $\lambda > 0$ is the hyper-parameter to control the degree of regularization. θ_{ref} is typically set as the model parameters after the SFT training. $\mathcal{D}_{\text{RLHF}}$ is another prompt-response dataset held out for RLHF training.

To train the reward function, an additional preference dataset $\mathcal{D}_{\text{pref}} = \{\boldsymbol{x}_i, \boldsymbol{y}_i^w, \boldsymbol{y}_i^l\}_{i\in[|\mathcal{D}_{\text{pref}}|]}$ is typically required. For the same prompt $\boldsymbol{x}$, given two responses $\boldsymbol{y}^w$ and $\boldsymbol{y}^l$, a human annotator evaluates that $\boldsymbol{y}^w$ is preferred over $\boldsymbol{y}^l$. Hence, the loss function for the reward model is,

$$\mathcal{L}_{\text{rew}}(\psi) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^w,\boldsymbol{y}^l)\sim\mathcal{D}_{\text{pref}}}\Big[\log(\sigma(r_\psi(\boldsymbol{x},\boldsymbol{y}^w) - r_\psi(\boldsymbol{x},\boldsymbol{y}^l)))\Big], \tag{2.18}$$

where σ is the sigmoid function. For successful RLHF training, a high-quality preference dataset is essential.

Direct policy optimization. Rafailov et al. [130] propose the direct policy optimization (DPO) algorithm to bypass the reward function training,

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^w,\boldsymbol{y}^l)\sim\mathcal{D}_{\text{pref}}}\Big[\sigma(\lambda \log \frac{f_\theta(\boldsymbol{y}^w|\boldsymbol{x})}{f_{\theta_{\text{ref}}}(\boldsymbol{y}^w|\boldsymbol{x})} - \lambda \log \frac{f_\theta(\boldsymbol{y}^l|\boldsymbol{x})}{f_{\theta_{\text{ref}}}(\boldsymbol{y}^l|\boldsymbol{x})})\Big]. \tag{2.19}$$

Odds ratio preference optimization. Hong, Lee, and Thorne [79] develop a reference model-free method ORPO to learn from preference data,

$$\mathcal{L}_{\text{ORPO}}(\theta) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}^w,\boldsymbol{y}^l)\sim\mathcal{D}_{\text{pref}}}\Big[\log f_\theta(\boldsymbol{y}^w|\boldsymbol{x}) + \lambda \log\sigma\Big(\log \mathbf{OR}_\theta(\boldsymbol{y}^w,\boldsymbol{y}^l)\Big)\Big], \tag{2.20}$$

where $\mathbf{OR}_\theta(\boldsymbol{y}^w, \boldsymbol{y}^l)$ refers to the odds ratio, indicating the likelihood that the model θ generates $\boldsymbol{y}^w$ compared to $\boldsymbol{y}^l$. The definition can be found in the original paper [79]. From the form of the above objective, we can see ORPO as a combination of SFT and DPO.

Discussion. From the above discussion, it becomes clear that high-quality human annotators are indispensable for all fine-tuning methods, whether for demonstrations in SFT or preference labels in preference optimization methods. However, given the high costs associated with recruiting human annotators and implementing rigorous quality control, data quality remains a challenge for parties with limited budgets.

2.3.4 Methodology

In this section, we describe our Guided Fine-Tuning (`GFT`) framework (as shown in Figure 2.13) in detail. We consider a low-fidelity dataset $\mathcal{D}_{lofi}$ and the corresponding LLM pretrained on it f_θ which is parameterized by θ. Meanwhile, we have another high-fidelity dataset $\mathcal{D}_{hifi}$ and the corresponding LLM pretrained on it g_ϕ which is parameterized by ϕ. In this paper, we assume the LLMs are capable of fully learning the patterns in the datasets in the pretraining stage, thus the fidelity of the datasets determines the fidelity of the corresponding LLMs. Considering data is now a vital asset for companies, which may share trained models without revealing their datasets, we assume the pretraining data for LLMs is inaccessible. Thus, our goal is to improve a low-fidelity LLM with the guidance from a high-fidelity LLM. The high-fidelity model could be either an open-source model (e.g., LLaMA[2]) or a closed-source model (e.g., ChatGPT[3]).

We present the algorithm of `GFT` in Algorithm 1. As previously discussed, the high-fidelity datasets are inaccessible, making it challenging to fine-tune low-fidelity LLMs using SFT or RLHF. Thus, we acquire a prompt dataset $\mathcal{D}_{\text{GFT}} = \{(\boldsymbol{x})_i\}_{i\in[N]}$. Here, we employ the high-fidelity LLM to generate responses to the prompt $\boldsymbol{x}$, using these predictions to fine-tune the low-fidelity LLM.

For a given prompt dataset $\mathcal{D}_{\text{GFT}}$, we excute the following two steps:

1. **Create weak supervision using the high-fidelity LLM.** In line 5 of Algorithm 1, we generate response $\boldsymbol{y}_i$ to a prompt $\boldsymbol{x}_i$ using g_ψ. These responses do not constitute *strong* supervision due to potential biases and limitations inherent in g_ψ. However,

[2]`https://llama.meta.com/llama2/`
[3]`https://openai.com/gpt-4`

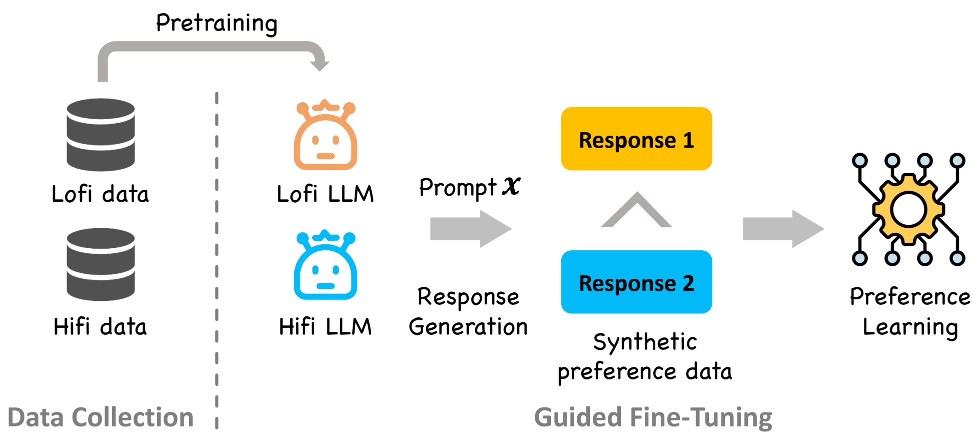


Figure 2.13: Overview of Guided Fine-Tuning (GFT).

they can still serve as *weak* supervision, given their relatively better performance compared to the low-fidelity model.

2. **Train the low-fidelity LLM with weak supervision.** The responses from the high-fidelity model can be directly utilized for the SFT training. Also, we can pair the response for the same $\boldsymbol{x}$ from the low-fidelity model as synthetic preference datasets for the RLHF training. We report the results of a set of *off-the-shelf* optimization algorithms discussed in Section 2.3.3 in this step.

As demonstrated in Algorithm 1, the procedure can be repeated for T iterations. Considering f_θ is continuously improving, g_ψ could provide less useful training signals, thus we set a stop criterion ϵ in line 10.

Remark. `GFT` is a general framework as it can pair any low-fidelity and high-fidelity models, and improve the low-fidelity model without the need for access to high-fidelity data or the requirement of extra human effort for data labeling. However, it also has some limitations: 1) The potential bias and limitations of the high-fidelity model could be learned by the low-fidelity model. 2) The high-fidelity model could not be rigorously better than the low-fidelity model in any domains, thus the general performance of the low-fidelity model could decrease.

2.3.5 Experiments

In this section, we empirically evaluate GFT's ability to improve low-fidelity LLMs.

Dataset. We consider the **single-turn dialogue** generation task and utilize the Anthropic Helpful and Harmless dialogue dataset (abbr., Anthropic-HH) [8]. Given a human query $\boldsymbol{x}$, which could be anything from a question about quantum physics to recipe of ice cream

Algorithm 1: Guided Fine-Tuning (`GFT`)

```
1  Input: $\mathcal{D}_{\text{GFT}} = \{(\boldsymbol{x}_i)\}_{i\in[N]}$: Prompt dataset, $f_{\theta_0}$: low-fidelity model with
     parameter $\theta_0$, $g_\phi$: high-fidelity model with parameter $\phi$, $T$: Number of iterations,
     $\epsilon$: Stop criterion;
2  for $t = 0, ..., T-1$ do
3      $D_t = \emptyset$;
4      for $i = 1, ..., N$ do
5          Generate synthetic chosen response $\boldsymbol{y}_i^w \sim g_\phi(\cdot|\boldsymbol{x}_i)$;
6          Generate synthetic rejected response $\boldsymbol{y}_i^l \sim f_{\theta_t}(\cdot|\boldsymbol{x}_i)$;
7          $D_t = D_t \cup \{\boldsymbol{x}_i, \boldsymbol{y}_i^w, \boldsymbol{y}_i^l\}$;
8      end
9      Update $\theta_{t+1}$ using SFT or preference optimization methods;
10     if $\|\theta_{t+1} - \theta_t\| \leq \epsilon$ then
           // Quit the loop if the model stops updating
11         break;
12     end
13 end
```

cake, an LLM is expected to produce an engaging and helpful response $\boldsymbol{y}$ to the user's query. This dataset contains 70k dialogues between human users and an automated assistant. Each dialogue ends with a pair of responses generated by a large (although unknown) language model along with a preference label denoting the human preferred response.

Evaluation. We follow [130] to evaluate algorithms with their *win rate* against a baseline policy, using GPT-4 as a proxy for human evaluation of response helpfulness. Specifically, we use the preferred response in the test dataset as the baseline. To save the evaluation cost, we hold out 100 samples from the test set to calculate the win rate. Besides win rate, we also report *lose rate* and *tie rate* for a more comprehensive analysis.

Implementation details. We use a fine-tuned `opt-1.3b` model[4] as the base model. To simulate the low-fidelity dataset, we randomly flip the preference labels of the Anthropic-HH dataset with a 50% probability and then fine-tune the base model on this corrupted dataset using the DPO algorithm. We denote this model as `opt-1.3b-noise`. We train another `opt-1.3b` on the raw dataset and use it as the high-fidelity model, denoted by `opt-1.3b-dpo`. We also incorporate different optimization algorithms in `GFT`: We experiment with SFT and DPO independently, and also explore a combined approach where we first apply SFT followed by DPO (SFT+DPO). Additionally, we also report the results with ORPO, which is another way to combine SFT and preference optimization.

[4]`https://huggingface.co/AdamG012/chat-opt-1.3b-sft-deepspeed`

Table 2.5: Win rate on Anthropic-HH. We use `opt-1.3b-noise` as the low-fidelity LLM, and `opt-1.3b-dpo` as the high-fidelity LLM.

		Win rate	Lose rate	Tie rate
`opt-1.3b-noise`		0.32	0.44	0.24
`GFT`	SFT	0.39	0.41	0.20
	DPO	0.43	0.35	0.22
	SFT+DPO	0.44	0.36	0.20
	ORPO	0.46	0.38	0.16
`opt-1.3b-dpo`		0.44	0.3	0.26

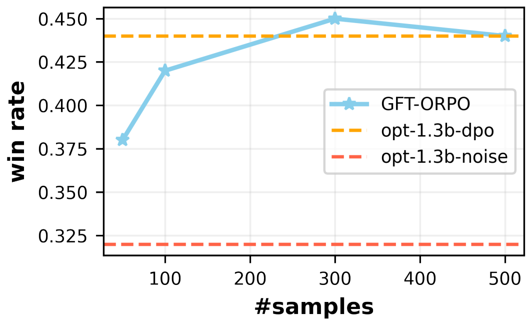
Table 2.6: Win rate on Anthropic-HH. We use `opt-1.3b-sft` as the low-fidelity LLM, and `mistral-7b-dpo` as the high-fidelity LLM.

		Win rate	Lose rate	Tie rate
`opt-1.3b-sft`		0.29	0.39	0.32
`GFT`	SFT	0.59	0.18	0.23
	DPO	0.60	0.31	0.09
	SFT+DPO	0.69	0.16	0.15
	ORPO	0.71	0.13	0.16
`mistral-7b-dpo`		0.84	0.02	0.14

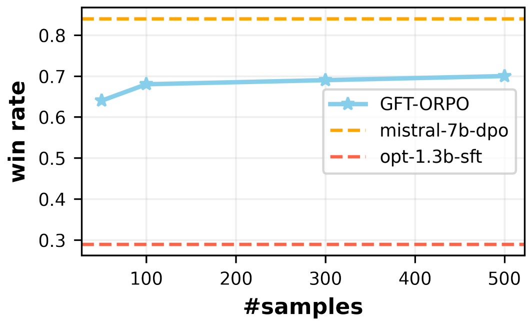
Results & analysis. We report the results in Table 2.5. After the `GFT` training, the performance of the low-fidelity model significantly improved, even slightly surpassing that of the high-fidelity model with respect to win rate. In `GFT`, we construct weak preference labels using generated responses from the low-fidelity model, serving as a form of on-policy evaluation in contrast to off-policy evaluation, which would utilize rejected responses in the original dataset. Thus our approach opens up possibilities to outperform the high-fidelity model, especially when their capacities are comparable, as we observed in the experiments. The experiment results indicate without denoising the raw noisy training data, `GFT` is able to improve the low-fidelity model from the guidance of the high-fidelity model.

Can `GFT` effectively learn from more capable model? We also consider a setting where the high-fidelity model is the more capable `mistral-7b-dpo`, which comprises 7 billion parameters. Concurrently, we employ `opt-1.3b-sft` as the low-fidelity model. The results are reported in Table 2.6. However, a performance gap from the high-fidelity model remains, which we hypothesize is attributable to both the capacity and the quality of the pre-training data of the base model. We also observe that combining SFT and preference optimization methods yields better performance than using them separately, such as SFT+DPO and ORPO. This suggests that when the high-fidelity model is more capable than the low-fidelity model, SFT is an important step to mimic the behavior of the high-fidelity model. After SFT, synthetic preference data play an important role to further fine-tune the model.

Sample efficiency. We further study the sample efficiency of `GFT` in Figure 2.14, where we vary the number of samples from the high-fidelity model. We report the results with the best optimization algorithm ORPO observed in Table 2.5. The performance of the low-fidelity model was observed to converge rapidly with just a few hundred examples provided by the high-fidelity model. The phenomena indicates the noise contained in a low-fidelity model can be efficiently mitigated with the guidance from a high-fiedlity model.



(a) Low-fidelity LLM is `opt-1.3b-noise` and high-fidelity LLM is `opt-1.3b-dpo`.



(b) Low-fidelity LLM is `opt-1.3b-sft` and high-fidelity LLM is `mistral-7b-dpo`.

Figure 2.14: Win rates w.r.t. number of samples.

2.4 Conclusion

In this chapter, we investigate the problem of learning from noisy human feedback from various aspects. Firstly, aside from the widely employed independent noise assumptions across annotators, in [33], we decompose annotation noise into common and individual confusions. We used neural networks to realize our probabilistic modeling of crowdsourced data, and estimate each component in our solution in an end-to-end fashion. Extensive empirical evaluations confirm the advantage of our solution in learning from complicated real-world crowdsourced data.

Secondly, data sparsity poses a serious challenge to current learning from crowds solutions. In [35], we present a data augmentation solution using generative adversarial networks to handle the issue. We proposed two important principles in generating high-quality annotations: 1) the generated annotations should follow the distribution of authentic ones; and 2) the generated annotations should have high mutual information with the ground-truth labels. We implemented these principles in our discriminative model design. Extensive experiment results demonstrated the effectiveness of our data augmentation solution in improving the performance of the classifier learned from crowds, and it sheds light on our solution's potential in low-budget crowdsourcing in general.

Thirdly, data quality determines the performance of fine-tuned LLMs. We propose leveraging the knowledge from high-fidelity LLMs to eliminate the noise present in low-fidelity LLMs. To realize it, we develop a guided fine-tuning framework by comparing the responses to same prompts from both low-fidelity and high-fidelity models. Experiments on a single-turn dialogue benchmark show that our method not only effectively improves the performance of low-fidelity models but also efficiently achieves this with only a few samples provided by high-fidelity models.

In conclusion, our study offers valuable insights and practical techniques for addressing challenges in learning from noisy human feedback. Through detailed analysis and investigation of noise generation, issues of sparsity, and strategies for noise reduction in human feedback, we lay the groundwork for creating algorithms that allow modern information systems to effectively and efficiently learn from real-world human feedback.

Chapter 3

Learning from Interactive Human Feedback

As we discussed in Chapter 1, human users play a more important part in modern human-system ecosystems. A system's ability to remain competitive in the market is significantly dependent on its mechanism for interpreting and eliciting user feedback. Without this, the system risks losing efficacy over time. In this chapter, we focus on improving the the sustainability of human-system interactions. Our approaches involve the development of more sophisticated and efficient algorithms designed to learn and adapt personalized systems, aiming to optimize user engagement and system responsiveness over time.

3.1 Multi-objective intrinsic reward learning for conversational recommendation systems

3.1.1 Introduction

Conversational recommender systems (CRS) leverage interactive conversations to delineate a user's preferences [37, 105, 208]. The conversations revolve around questions aimed at discerning users' preferences on specific item attributes (e.g., music genres). Through an interactive process of questions and answers, a profile about a user's intended item can be depicted. Numerous CRS formulations have been proposed [26, 29, 30]. In this work, we investigate a prevalent CRS setting known as the multi-round conversational recommendation [47, 105], where a CRS agent can ask a question or recommend an item in consecutive rounds of conversations. The conversation continues until the user accepts the recommendation (indicating a successful conversation) or quits the conversation (considered as a failed conversation).

CRS fundamentally embodies a sequential decision making problem, for which numerous reinforcement learning (RL)-based solutions have been proposed [37, 105]. However, as the users only provide textual or binary responses (e.g., accepting or rejecting the inquired attributes), existing RL-based solutions heavily rely on heuristic reward functions that are manually defined to train CRS policies. These reward functions, such as promoting attributes accepted by a user and penalizing those rejected, may not accurately reflect user intent due to their heuristic nature. This becomes problematic since the effectiveness of CRS policy learning largely depends on the quality of pre-defined reward function – an inadequately designed reward function can lead to solutions that significant deviates from optimality. Additionally, these arbitrary reward functions can inadvertently distort the modeling of conversation states, influencing the subsequent actions taken by the RL agent.

Arguably, an effective reward function should promote actions that lead to more precise modeling of users' preferences. As a result, different attributes or items, including those rejected, can each uniquely contribute to user preference modeling. As an example illustrated in Figure 3.1, even though *Heavy metal rock* is rejected by the user, it still, to certain extent, contributes to identifying the target item, *Hey Jude*. However, existing handcrafted heuristic reward functions fall short in delivering information at this granularity, as they assign uniform rewards to all accepted or rejected actions. This motivates us to *learn a reward function* that enables more fine-grained policy learning.

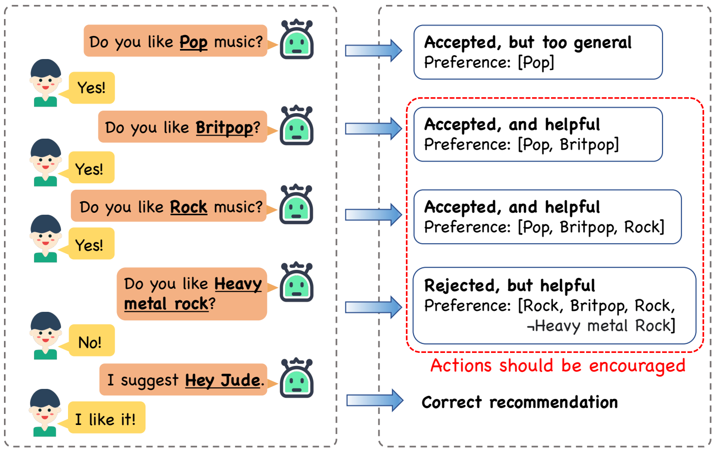


Figure 3.1: Motivating example of intrinsic reward learning.

Instead of manually define reward functions, we introduce a principled approach to reward learning for CRS, where we learn a *intrinsic reward* for each action taken by the agent utilizing the optimal rewards framework [150]. This framework delineates the optimal intrinsic reward function as the one that, when employed by an RL agent, fosters behaviors that optimize the task-specific or *extrinsic rewards* – in the case of CRS, successful recommendations.

Two notable technical challenges stand out when learning intrinsic rewards for CRS. First, explicit extrinsic rewards in CRS are extremely sparse, which complicates the intrinsic reward learning. Despite that the agent interacts with the user in each round, the only clear extrinsic reward signal, which is whether the overall conversation is successful or not, is only revealed from the user at the conclusion of the conversation. The significance of each accepted or rejected attribute/item prior to the conversation's ends remain ambiguous. For instance, an inquired attribute that is rejected by the user does not necessarily imply a negative reward for policy learning, as it can signify what the user is not looking for. Second, the assessment of CRS is multi-dimensional, entailing various factors that contribute to the overall effectiveness and user experience, such as recommendation quality and user effort. On the one hand, asking more questions may be necessary to accurately profile user preferences to facilitate a successful recommendation. On the other hand, reducing user effort in conversations (i.e., fewer conversation turns) is essential to ensure users' engagement and maintain their satisfaction. Balancing these factors is a delicate task.

To tackle the challenges for improving CRS from a reward learning perspective, we develop an online algorithm for learning intrinsic reward functions via multi-objective bi-level optimization. We name the proposed solution **CRSIRL**, meaning **CRS** with **I**ntrinsic **R**eward **L**earning. In the inner loop of CRSIRL, the policy is optimized with the learned intrinsic reward function. In the outer loop, the intrinsic reward function is updated to satisfy two specific criteria designed for CRS. The first criterion aims to maximize the sparse extrinsic reward, augmented by a reward shaping strategy to encourage actions that promote the target item as quickly as possible. The second criterion involves tailoring the learnt reward function to promote successful trajectories over the failed ones. The results of our extensive experiments demonstrate that CRSIRL not only improves the success rate of CRS but also achieves it with shorter conversations.

3.1.2 Related works

Conversational Recommder Systems. Christakopoulou, Radlinski, and Hofmann [30] pioneered the concept of Conversational Recommender Systems (CRS). Their approach primarily focused on determining which items to solicit feedback on and applied off-the-shelf metrics such as the upper confidence bound [4] for this purpose. This laid the groundwork for reinforcement learning (RL) based methods, which have recently become the prevalent solutions for CRS. For example, Sun and Zhang [159] developed a policy network to decide whether to recommend an item or inquire about an item attribute at each conversation turn. However, these initial studies terminated the conversation upon making a recommendation, regardless of user acceptance. Lei et al. [105] studied multi-round conversational recommendation, where CRS can ask a question or recommend an item in multiple rounds before the user accepts the recommendation or quits. This is also the setting of our work in this paper. To better address multi-round CRS, Lei et al. [106] leveraged knowledge graphs to select more relevant attributes to ask across turns. Xu et al. [191] extended [105] by revising user embeddings dynamically based on users' feedback on attributes and items. And Deng et al. [47] unified the question selection module and the recommendation module in an RL-based CRS solution. However, all the aforementioned works depend on heuristically crafted reward functions, which may lead policies to deviate from the optimal solution. In this work, we propose to learn intrinsic rewards which can maximize the recommendation performance.

Intrinsic Reward Learning in Reinforcement Learning. Intrinsic reward learning has emerged as a promising approach to enhance the performance and efficiency of reinforcement learning algorithms. Singh et al. [150] introduced the Optimal Reward Framework which aims to find a good reward function that allows agents to solve a distribution of tasks using exhaustive search. Pathak et al. [128] introduced the concept of curiosity-driven intrinsic rewards, where the agent is rewarded for actions that lead to novel states, improving its ability to explore complex environments. Zheng, Oh, and Singh [213] proposed a meta-gradient method named LIRPG to learn intrinsic rewards via a bi-level optimization framework. Zheng et al. [212] extended LIRPG by learning intrinsic rewards on a distribution of tasks. Liu et al. [118] developed another meta-gradient method to learn intrinsic rewards from trajectory preferences. In this work, we propose a novel intrinsic reward learning framework designed for CRS, where we learn intrinsic reward functions to satisfy multiple CRS-sepcific objectives from users' extremely sparse explicit reward feedback.

3.1.3 Preliminaries

In this section, we define the notations to be used in our technical discussions and some basic notions in multi-objective optimization.

Problem definition. Similar to traditional recommender systems, CRS serves a set of users $\mathcal{U}$ with a set of items $\mathcal{V}$; and we denote a specific user as u and an item as v. Each item v is associated with a set of pre-defined attributes $\mathcal{P}_v$. Attributes describe basic properties of the items, such as genres in movie recommendations and cuisine type in restaurant recommendations.

We formulate the CRS problem using a Markov decision process (MDP) [36, 47, 106], which can be fully described by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. S denotes the state space, which summarizes the conversation between the system and user so far. $\mathcal{A}$ denotes the action space for the system, which includes recommending a particular item or asking for feedback on a specific attribute. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function.

With this formulation, a conversation in CRS can be represented as $d = \{(a_1, r_1), ...(a_T, r_T)\}$, where T is the maximum number of allowed turns. A conversation (or an episode in the language of RL, which we will use exchangeablely) terminates when: (1) the user accepts the recommended item; or (2) the CRS agent runs out of maximum number of allowed turns. At each time step t, the CRS agent, which follows a policy $\pi_\theta(a_t|s_t)$ parameterized by θ, selects an action a_t based on the current state s_t. The training objective of a CRS policy is to maximize the expected cumulative rewards over the set of observed episodes $\mathcal{D}$, i.e., minimizing the loss

$$\mathcal{L}(\pi) = -\mathop{\mathbb{E}}_{d \sim P(\mathcal{D})} \big[\sum_{t=0}^{T} R_t \big], \tag{3.1}$$

where $R_t = \sum_{t'=t}^{T} \gamma^{T-t'} r(a_t)$ is the accumulated reward from turn t to the final turn T, and $\gamma \in [0, 1]$ is a discount factor to emphasize rewards collected in a near term.

Instead of using handcrafted reward functions $\mathcal{R}$ as in previous works [37, 47, 105], we learn an intrinsic reward function defined as $r_\phi^{in}(s, a)$ parameterized by ϕ to enhance CRS policy learning. In this context, the original CRS-specific reward is referred to as the extrinsic reward, denoted as $r^{ex}(s, a)$. The extrinsic reward is inherently sparse, as the only discernible and useful reward signal is the success or failure of an episode, with the intermediate actions' contributions remaining ambiguous. We assign a positive extrinsic reward at the conclusion of a successful episode and a negative reward otherwise. All intermediate actions are assigned a zero extrinsic reward.

Multi-objective optimization. We utilize multi-objective optimization (MOO) to achieve the multi-dimensional goal of CRS, i.e., maximizing the success rate and reducing the length of conversations. MOO aims to simultaneously optimize multiple objectives, possibly conflicting ones. This results in a trade-off among objectives, making the CRS problem more complex and challenging to solve. In these cases, the Pareto-optimal solutions represent different optimal trade-offs between the objectives [44].

Consider M objective functions $\{\mathcal{L}^1, ..., \mathcal{L}^M\}$, a model parameterized by θ is optimized

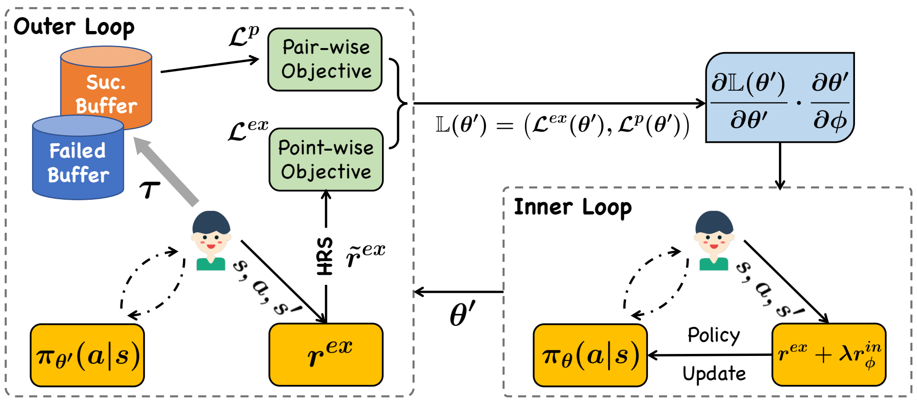


Figure 3.2: Overview of CRSIRL,which consists of two modules, a policy parameterized by θ and an intrinsic reward function parameterized by ϕ. The optimization of CRSIRL has two levels. In the inner level, a policy is trained to maximize the return defined by both intrinsic and extrinsic rewards. In the outer level, the intrinsic reward function is trained to optimize two CRS-specific objectives realized by the learnt policy's behaviors.

towards them. We specify the multi-objective optimization using a vector valued loss $\mathbb{L}$,

$$\min_\theta \mathbb{L}(\theta) = \min_\theta \left(\mathcal{L}^1(\theta), ..., \mathcal{L}^M(\theta)\right)^\top \tag{3.2}$$

The goal of multi-objective optimization is achieving Pareto optimality.

Definition 1 (Pareto optimality)**.**

(a) A solution θ dominates a solution $\bar{\theta}$ if $\mathcal{L}^t(\theta) \leq \mathcal{L}^t(\bar{\theta})$ under all objectives and $\mathbb{L}(\theta) \neq \mathbb{L}(\bar{\theta})$.

(b) A solution θ^* is called Pareto optimal if there exists no solution θ that dominates θ^*.

The set of Pareto optimal solutions is called the Pareto front $\mathcal{P}_\theta$.

3.1.4 Methodology

To tackle the challenges for improving CRS from a reward learning perspective, we develop an online algorithm for learning intrinsic reward functions via multi-objective bi-level optimization. As shown in Figure 3.2, CRSIRL operates on two tiers of optimization: the inner optimization, which improves the policy using both the extrinsic reward and the learned intrinsic rewards, and the outer optimization, which refines the intrinsic reward function based on the policy assessment derived from the inner optimization. Given the absence of supervision for the intrinsic reward function, we establish the relationship between it and the refined policy via gradient descent in the inner optimization. Specifically, we compute meta-gradient for the intrinsic reward function using chain rule in the outer optimization. In the outer optimization, we design a point-wise objective striving to enhance extrinsic rewards in the learnt intrinsic reward function, through the use of hindsight reward shaping (HRS). This objective aids in pinpointing pivotal actions that significantly improve the target item's ranking, thereby shortening the conversation length. In parallel, we introduce a pair-wise objective that favors successful trajectories over unsuccessful ones, which assists in identifying actions that result in preferred conversations. This objective is named as recommendation preference matching (RPM). Finally, we introduce a holistic multi-objective bi-level optimization framework that optimizes intrinsic rewards to meet both objectives.

Hindsight reward shaping

As we discussed before, the extrinsic reward is extremely sparse in CRS. The only clear and informative signal from the extrinsic reward is whether the conversation is successful, making it hard to judge the progress of user preference elicitation during the conversation. Reward shaping, as proposed by Ng, Harada, and Russell [122], serves as a valuable tool for incorporating task-specific knowledge to estimate the reward function. We leverage reward shaping within the outer loop of our model to imbue the process of intrinsic reward learning with more nuanced, task-specific guidance. We use the following hindsight reward shaping to augment the extrinsic reward,

$$\tilde{r}^{ex}(s_t, a_t) = r^{ex}(s_t, a_t) + \gamma w(s_{t+1}, v) - w(s_t, v), \tag{3.3}$$

where w is a scoring function, v is the target item and γ is a discount factor. $\tilde{r}^{ex}(s_t, a_t)$ encourages actions which promote the target item. In turn, it helps shorten the conversation length. In our experiments, we use $w = \log(\rho(s_t, v)+1)$, where $\rho(s_t, v)$ is the rank of target item v under state s_t.

Lemma 2. *Consider any reward shaping function $\mathcal{F} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, we say $\mathcal{F}$ is a **potential-based** reward shaping function (PBRS) if there exists a real-valued function $\Phi : \mathcal{S} \to \mathbb{R}$ satisfying,*

$$\mathcal{F}(s, a, s') = \gamma\Phi(s') - \Phi(s), \tag{3.4}$$

Then $\mathcal{F}$ being PBRS is a necessary and sufficient condition for it to guarantee the consistency of the optimal policy, i.e., the optimal policy of ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$+$\mathcal{F}$) is the same as ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$).

The proof is based on [122] and omitted in this paper. By matching the form of Eq.(3.4) and Eq.(3.3), we can conclude the hindsight reward shaping satisfies the PBRS condition, and thus the optimal policy is consistent. The resulted objective induced by HRS is

$$\mathcal{L}^{ex}(\theta) = -\mathbb{E}\big[\sum_{t=0}^{T} \tilde{R}_t^{ex}\big], \tag{3.5}$$

where $\tilde{R}_t^{ex} = \sum_{t'=t}^{T} \gamma^{T-t'} \tilde{r}_{t'}^{ex}$. Note that the information of target item is unknown beforehand, we can only use HRS after the target item is hit, which is why we call it *hindsight*. Otherwise HRS degenerates to the original extrinsic reward.

Recommendation preference matching

Even though the contributions of intermediate actions to a conversation are undefined in the extrinsic reward, it is still feasible to discern valuable intermediate actions that could potentially lead to a successful conversation, and the learned intrinsic reward should help us identify them. We realize this by contrasting successful and failed episodes by the learnt intrinsic reward: a preferred episode should have a higher likelihood under the optimal policy, comparing to a less preferred one; and this optimal policy should be achieved by the correct reward function. Given a policy π_θ, the probability of conversation τ^0 is preferred over τ^1 is computed based on the likelihood of the trajectories,

$$P_\theta\big[\tau^0 \succ \tau^1\big] = \frac{\exp \sum_{t\in\tau^0} \log \pi_\theta(a_t|s_t)}{\exp \sum_{t\in\tau^0} \log \pi_\theta(a_t|s_t) + \exp \sum_{t\in\tau^1} \log \pi_\theta(a_t|s_t)}, \tag{3.6}$$

Assume τ^0 is preferred over τ^1, the resulting loss function is given by,

$$\mathcal{L}^{p}(\theta) = -\mathbb{E}\Big[\sum_{\tau^0 \succ \tau^1} \log P_\theta\big[\tau^0 \succ \tau^1\big]\Big], \tag{3.7}$$

where $\tau^0, \tau^1 \in \mathcal{B}$ are sampled from a buffer storing past trajectories. This follows the Bradley-Terry model [14] for estimating score functions from pairwise preferences. In the context of CRS, the preference is naturally defined by whether the recommendation is successful or not; and among successful recommendations, we prefer the one shorter. We truncate the failed trajectory to match the length of the successful trajectory.

Multi-objective bi-Level optimization

The intrinsic reward function is expected to lead to a policy satisfying the above two objectives. This translates to a bi-level optimization procedure for policy learning: first update the policy with learned intrinsic rewards, and then improve the intrinsic rewards to help the resulting policy better satisfy the above two objectives. More formally, we define,

$$\begin{aligned} &\min_{\phi} \ \mathbb{L}(\theta'), \\ &\text{s.t.} \ \ \theta' = \arg\min_{\theta} \mathcal{L}^{ex+in}(\theta, \phi). \end{aligned} \tag{3.8}$$

where $\mathbb{L}(\theta') = \big(\mathcal{L}^{ex}(\theta'), \mathcal{L}^{p}(\theta')\big)$ and $\mathcal{L}^{ex+in}(\theta, \phi)$ is the negative cumulative reward calculated with weighted sum $r^{ex} + \lambda r^{in}_{\phi}$, λ is a hyper-parameter to balance two rewards. In the inner loop, we optimize the policy with both the extrinsic reward and the learned intrinsic reward function. In the outer loop, we optimize the intrinsic reward function to minimize the vector value loss. To derive the gradients for optimization, we first build the connection between θ and ϕ in the inner loop, and then derive the gradients on ϕ in the outer loop.

Inner Loop: Optimizing θ, building the connection between θ and ϕ. We update θ as follows,

$$\theta' = \theta - \eta \cdot \nabla_{\theta} \mathcal{L}^{ex+in}(\theta, \phi), \tag{3.9}$$

where $\nabla_{\theta} \mathcal{L}^{ex+in}(\theta, \phi)$ can be calculated by the policy gradient theorem [160] and η is the learning rate used in the inner loop. In this way, the updated parameter θ' becomes a function of ϕ. With the built connection, we are able to compute the gradient of ϕ by taking the gradient of gradient on θ', i.e., the *meta-gradient*.

Outer Loop: Optimizing ϕ. In the outer loop, we optimize the vector value loss $\mathbb{L}(\theta')$ to satisfy aforementioned two CRS-specific objectives. Even though we do not have supervision on ϕ, the gradient of ϕ can still be derived using the chain rule,

$$g(\phi) = \frac{\partial \mathbb{L}(\theta')}{\partial \theta'} \cdot \frac{\partial \theta'}{\partial \phi} \tag{3.10}$$

Different from single objective optimization, the first part of Eq.(3.13) is the derivative w.r.t. the multi-objective function $\mathbb{L}(\theta')$. [145] adopt the multiple gradient descent algorithm (MDGA) [48] to find a Pareto stationary point for a MOO problem. We follow their approach to solve the following optimization problem,

$$\min_{\alpha \cdot \in [0,1]} \left\{ \left\| \alpha \nabla_{\theta'} \mathcal{L}^{ex}(\theta') + (1-\alpha) \cdot \nabla_{\theta'} \mathcal{L}^{p}(\theta') \right\|_2^2 \right\}, \tag{3.11}$$

where α has the following analytical solution,

$$\alpha = \left[\frac{\nabla_{\theta'} \mathcal{L}^{p}(\theta') - \nabla_{\theta'} \mathcal{L}^{ex}(\theta')^{\top} \nabla_{\theta'} \mathcal{L}^{p}(\theta')}{\left\| \nabla_{\theta'} \mathcal{L}^{ex}(\theta') - \nabla_{\theta'} \mathcal{L}^{p}(\theta') \right\|} \right]_{+,\frac{1}{\top}}, \tag{3.12}$$

where $[\cdot]_{+,\frac{1}{\top}}$ represents clipping to $[0,1]$ as $[a]_{+,\frac{1}{\top}} = \max(\min(a,1),0)$. The resulted meta-

Table 3.1: Summary statistics of datasets.

	LastFM	**LastFM***	**Yelp***
#Users	1,801	1,801	27,675
#Items	7,432	7,432	70,311
#Attributes	33	8,438	590
#Interactions	76,693	76,693	1,368,606

gradient of ϕ becomes,

$$g(\phi) = \alpha \cdot \frac{\partial \mathcal{L}^{ex}(\theta')}{\partial \theta'} \cdot \frac{\partial \theta'}{\partial \phi} + (1-\alpha) \cdot \frac{\partial \mathcal{L}^{p}(\theta')}{\partial \theta'} \cdot \frac{\partial \theta'}{\partial \phi}. \tag{3.13}$$

Thus ϕ is updated by,

$$\phi' = \phi - \beta \cdot g(\phi), \tag{3.14}$$

where β is the learning rate used in the outer loop. We can conclude the optimization in the outer loop as an automatic trade-off between two objectives, and thus the resulted intrinsic reward function is expected to strike a good balance between two CRS objectives.

Training procedure. In the inner loop, we first rollout an episode to calculate $\mathcal{L}^{ex+in}$. In the outer loop, we also rollout an episode to calculate $\mathcal{L}^{ex}$ and sample a pair from the trajectory buffer $\mathcal{B}$ to calculate $\mathcal{L}^{p}$. We run the inner loop and outer loop alternately until the model convergence.

3.1.5 Experiments

In this section, we conduct extensive experiments on three widely-used CRS benchmarks to study the following research questions: (1) Can CRSIRL achieve better performance than state-of-the-art CRS solutions? (2) How does each proposed component contribute to the final performance of CRSIRL? (3) How does the degree of intrinsic rewards affect the policy learning?

Main results

Datasets & baselines. We evaluate CRSIRL on three multi-round CRS benchmarks [47, 105]. The **LastFM** dataset is for music artist recommendation. Lei et al. [105] manually grouped the original attributes into 33 coarse-grained attributes. The **LastFM*** dataset is the version where attributes are not grouped. The **Yelp*** dataset is for local business recommendation. We summarize their statistics in Table 3.1. Training and evaluating CRS through direct user interactions can be prohibitively expensive at scale. We address this by employing the user simulator approach from [105], simulating a conversation session for each observed user-item interaction pair (u, v). In this simulation, item v is considered the target item, and its attribute set $\mathcal{P}_v$ is treated as the oracle set of attributes preferred by user u. The session begins with the simulated user specifying an attribute, randomly selected from $\mathcal{P}_v$. This simulation adheres to the "System Ask, User Respond" paradigm in CRS, as described in [208].

We consider a rich set of state-of-the-art CRS solutions. **Max Entropy** chooses to select an attribute with maximum entropy based on the current state, or to recommend the top ranked item. **Abs Greedy** [30] continues to suggest items until it either makes a successful recommendation or reaches the maximum number of allowed attempts. **CRM** [159] is an RL-based solution. It integrates user preferences into a belief tracker, which then guides the decision-making process regarding when to ask which attribute. **EAR** [105] proposes a three-stage RL solution consisting of estimation, action and reflection. **SCPR** [106] reconceptualizes the CRS problem as an interactive path reasoning process within a user-item-attribute graph. It selects candidate attributes and items based on their relationship to attributes that have already interacted with users within this graph. **FPAN** [191] extends the EAR model by utilizing a user-item-attribute graph to enhance offline representation learning. User embeddings are revised dynamically based on users' feedback on items and attributes in the conversation. **UNICORN** [47] merges the conversation and recommendation components into a unified RL agent. To streamline the RL training process, it proposes two heuristic strategies for pre-selecting attributes and items at each turn.

Evaluation metrics. We follow previous works on multi-round CRS to evaluate the performance of CRS with success rate at turn T (SR@T) and average turns (AT) of conversations. SR@T is the average ratio of successful episodes with T turns, while AT is the average number of turns for all conversations. We also report the two-level hierarchical normalized discounted cumulative gain [47] defined as

$$hDCG@(T, K) = \sum_{t=1}^{T} \sum_{k=1}^{K} r(t, k) \left[\frac{1}{\log_2(t+2)} + \left(\frac{1}{\log_2(t+1)} - \frac{1}{\log_2(t+2)} \right) \frac{1}{\log_2(k+1)} \right],$$

where T and K represent the number of conversation turns and recommended items in each turn, $r(t, k)$ denotes the relevance of the results at the turn t and position k. Intuitively, successful sessions with fewer turns are preferable for CRS. Also, the target item is expected to be ranked higher on the recommendation list at the success turn. We report $hDCG@(15, 10)$ by default.

Training details. All datasets are split by 7:1.5:1.5 ratio for training, validation and test-

Table 3.2: Main results. For SR@15 and hDCG, higher is better. For AT, lower is better. † represents the improvement over baselines is statistically significant with p-value < 0.01.

	LastFM			LastFM*			Yelp*		
	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG
Abs Greedy	0.222	13.48	0.073	0.635	8.66	0.267	0.189	13.43	0.089
Max Entropy	0.283	13.91	0.083	0.669	9.33	0.269	0.398	13.42	0.121
CRM	0.325	13.75	0.092	0.580	10.79	0.224	0.177	13.69	0.070
EAR	0.429	12.88	0.136	0.595	10.51	0.230	0.182	13.63	0.079
SCPR	0.465	12.86	0.139	0.709	8.43	0.317	0.489	12.62	0.159
FPAN	<u>0.630</u>	<u>10.16</u>	<u>0.224</u>	0.667	7.82	<u>0.407</u>	0.236	12.77	0.116
UNICORN	0.535	11.82	0.175	<u>0.788</u>	<u>7.58</u>	0.349	<u>0.520</u>	<u>11.31</u>	<u>0.203</u>
CRSIRL	**0.772**†	**10.12**†	**0.231**†	**0.913**†	**6.79**†	**0.431**†	**0.622**†	**10.61**†	**0.228**†

ing. We used the Transformer-based state encoder proposed in [37]. We adopt TransE [13] to pretrain the node embeddings on the training set, and use the user simulator described before for online policy learning on the validation set. We first pretrain the policy with only extrinsic reward using policy gradient and then apply CRSIRL to fine-tune the pretrained policy. The learning rates in the inner and outer loop are searched from $\{1e^{-5}, 5e^{-5}, 1e^{-4}\}$ with Adam optimizer. The coefficient of intrinsic reward λ is searched from $\{0.05, 0.1, 0.5, 1.0\}$. The discount factor γ is set to 0.999. All experiments are run on an NVIDIA Geforce RTX 3080Ti GPU with 12 GB memory. RL-based baselines rely on handcrafted rewards, we follow Lei et al. [105] to set (1) $r_{\text{rec\_suc}} = 1$ for successful recommendation; (2) $r_{\text{rec\_fail}} = -0.1$ for failed recommendation; (3) $r_{\text{ask\_suc}} = 0.1$ when the inquired attribute is confirmed by the user; (4) $r_{\text{rec\_fail}} = -0.1$ when the inquired attribute is dismissed by the user; (5) $r_{\text{quit}} = -0.3$ when the user quits the conversation without a successful recommendation. We set the maximum turn T as 15 and the size K of the recommendation list as 10. We provide more implementation details in the supplementary material.

Results & analysis. We present the main results in Table 3.2. We can clearly observe the CRSIRL outperformed all baselines with a large margin. Both FPAN and EAR are policy gradient based methods, but they pretrain their policies using conversation history generated by a rule-based strategy via supervised learning. This training approach biases policies towards pre-set rules, limiting the performance of policy learning on datasets with larger action spaces (like LastFM* and Yelp*), where more exploration is necessary. SCPR and UNICORN have relatively stable performance on all the datasets. Our CRSIRL outperforms all baselines significantly with its learned intrinsic rewards. Rather than arbitrarily

Table 3.3: Ablation study of different components of CRSIRL.

	LastFM			LastFM*			Yelp*		
	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG
PG	0.724	10.42	0.217	0.882	7.41	0.401	0.598	10.95	0.186
$\neg$HRS	0.732	10.58	0.219	0.898	7.16	0.401	0.602	11.21	0.196
$\neg$RPM	0.754	10.14	0.227	0.904	6.89	0.415	0.606	10.81	0.213
MTL	0.768	**10.09**	0.224	0.908	6.92	0.426	0.613	10.73	0.207
CRSIRL	**0.772**	10.12	**0.231**	**0.913**	**6.79**	**0.431**	**0.622**	**10.61**	**0.228**

assigning the reward values, we dynamically optimize them in CRSIRL. Any action that contributes to a final successful recommendation should receive credit and thus be promoted by policy learning, regardless of whether it involves a rejected attribute or a failed recommendation.

Ablation study

Contributions of each component in CRSIRL. We evaluate different variants of CRSIRL to study the contributions of each proposed component. Firstly, we disable the fine-tuning with CRSIRL and directly report the results after the policy gradient pretraining with only extrinsic rewards, denoted as PG. Secondly, we remove HRS and RPM to evaluate their individual effectiveness. In both variants, the outer loop degenerates to a single objective optimization problem. Finally, we conduct an experiment where instead of updating the objectives with MOO, we treat the two objectives as distinct tasks and assign them equal weights, a process referred to as Multi-Task Learning (MTL).

We present the results in Table 3.6. Interestingly, we observe that directly optimizing the extrinsic rewards with policy gradient already outperformed most of baselines. It is worth noting that PG uses sparser rewards than other baselines in Table 3.2 with manually-defined rewards for intermediate actions. However, the exploratory behavior of PG enables it to outperform these baselines. We can observe that without HRS, the AT metric degenerated on all three datasets. HRS prefers actions which can increase the rank of the target item, which is the most direct metric of action utilities in CRS. Even though the asked questions could still be helpful without HRS, HRS provides explicit hints about how to ask the most useful questions, leading to a smaller AT. Besides, the performance decreases after removing RPM, which finds actions leading to successful recommendations by comparing successful and failed trajectories. Lastly, MTL shows a significant improvement compared to PG, and it occasionally outperforms CRSIRL (e.g., AT on LastFM). However, MTL

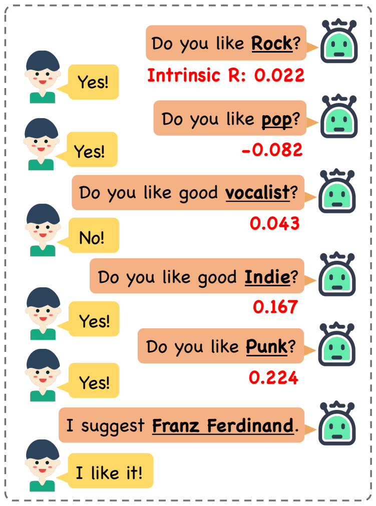


Figure 3.3: Conversations generated by CRSIRL. The values of the learned intrinsic rewards are marked in red.

has difficulty balancing the two objectives, generally resulting in worse performance than CRSIRL.

Case Study

Additionally, we performed a qualitative study to analyze the learned intrinsic rewards of CRSIRL (shown in Figure 3.3) on the LastFM dataset. The natural language questions and user responses are generated by predefined templates. We observe that the intrinsic rewards depend not only on whether the user accepts or rejects the action, but also on how well the action contributes to the final recommendation. Even though the user accepts *pop*, the intrinsic reward for this action remains negative. This is because *pop* is a very general attribute and contributes little to modeling the user's preference. Conversely, although *vocalist* is rejected by the user, it still carries a small positive value as it aids in identifying the target artist. Finally, *Indie* and *Punk* are two attributes that are accepted and best describe the target artist, *Franz Ferdinand*[1] (a band known for *indie rock* and *post-punk revival*). Consequently, they carry relatively large positive intrinsic rewards. This case shows the CRSIRL can provide more fine-grained reward signals, leading to better final performance.

[1]https://en.wikipedia.org/wiki/Franz_Ferdinand_(band)

3.2 Meta policy learning for cold-start conversational recommendations

3.2.1 Introduction

While traditional recommendation solutions infer a user's preferences only based on her historically interacted items [17–19, 75, 134, 142, 187], conversational recommender systems (CRS) leverage interactive conversations to adaptively profile a user's preference [30, 105, 159]. The conversations in CRS focus on questions about users' preferences on item attributes (e.g., brands or price range), in the form of pre-defined question templates [47, 105, 159] or timely synthesized natural language questions [112, 215]. Through a series of question answering, a profile about a user's intended item can be depicted, even when the user is new to the system [30], i.e., the cold-start users, which gives CRS an edge in providing improved recommendations.

Christakopoulou, Radlinski, and Hofmann [30] first proposed the idea of CRS. Their solution focused on deciding what item to ask for feedback; and off-the-shelf metrics, such as upper confidence bound [4], were leveraged for the purpose. Following this line, reinforcement learning (RL) based methods become the mainstream solution recently for CRS. Sun and Zhang [159] built a policy network to decide whether to recommend an item, or otherwise which item attribute to ask about in each turn of a conversation. However, in these two early studies, the conversation is terminated once a recommendation is made, no matter whether the user accepts it or not. Lei et al. [105] studied multi-round conversational recommendation, where CRS can ask a question or recommend an item multiple times before the user accepts the recommendation (considered as a successful conversation) or quits (considered as a failed conversation). This is also the setting of our work in this paper. To better address multi-round CRS, Lei et al. [106] leveraged knowledge graphs to select more relevant attributes to ask across turns. Xu et al. [191] extend [105] by revising user embeddings dynamically based on users' feedback on attributes and items. And Deng et al. [47] unified the question selection module and the recommendation module in an RL-based CRS solution, which simplifies the training of CRS. However, all aforementioned RL-based methods rely on existing user embeddings to conduct conversations and recommendations, which are not applicable to new users.

Although CRS is expected to address the cold-start problem in recommendation, by profiling a new user via eliciting her preference about item attributes, how to acquire the most effective feedback to profile a single user still encounters the cold-start problem. More specifically, due to the heterogeneity of different users' preferences, the same policy can

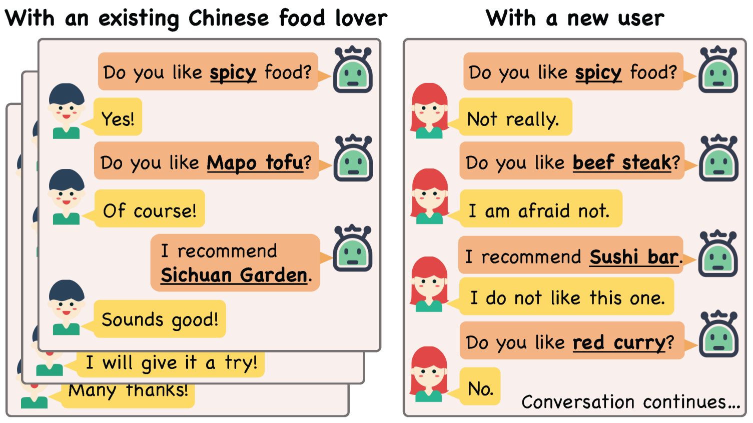


Figure 3.4: Example of cold-start CRS.

hardly be optimal in finding the sequence of interactions (asking questions or making recommendations) for all users, especially for those who do not contribute to the policy training. Consider the example shown in Fig.3.4, a policy trained with a population of Chinese food lovers cannot effectively serve new users who do not have any preferences on Chinese food. Once the interaction trajectory deviates from those often encountered during training, the effectiveness of the globally learnt CRS policy deteriorates, so does the quality of its recommendations.

We attribute this new challenge as cold-start policy learning in CRS, which is completely non-trivial but unfortunately ignored in most previous CRS studies. The goal is clear, i.e., adapt a CRS policy for each new user; but there are at least three main technical barriers blocking us from the goal. Firstly, *how to efficiently adapt a policy to new users?* The tolerance of users about a prolonged conversation or bad recommendations is limited [34, 35, 58, 111, 143], since all users wish to get high-quality recommendations with the least effort (e.g., shorter conversations) [165]. Hence, one cannot expect a large number of observations for CRS policy learning in a single user. Secondly, *how to effectively explore user preferences for policy adaptation?* As shown in Fig.3.4, successfully adapting a CRS policy to a new user depends on the user's preference, which however is elicited by the policy itself. This forms a *chicken-and-egg* dilemma [115] and adds another layer of consideration when acquiring user feedback: before identifying what item the user is looking for, one first needs to figure out what policy best suits for the inquiry. Thirdly, *how to decouple the adaptation of the conversation component and recommendation component in a CRS policy?* The conversation component (i.e., conversational policy) in CRS is to profile a user by actively eliciting her feedback, while the recommendation component (i.e., item recommender) is to identify the most relevant recommendations based on the profile.

Adaptation in both components is needed for new users, but the strategy for adapting them could be different for respective goals.

In this paper, we address the problem of CRS policy learning for cold-start users via meta reinforcement learning [85, 115, 177], and name the proposed solution **MetaCRS**. For the first challenge, we propose to learn a meta policy for CRS from a population of users and adapt it to new users with only a few trials of conversational recommendations. The meta policy can be viewed as a starting point close to every single user's personalized policy. It thus builds the basis for efficient policy adaptation with only a handful of observations in each new user. Secondly, to acquire the most informative feedback for policy adaptation, we design a meta-exploration policy to identify user preferences via a few exploratory conversations. Thirdly, in addition to the CRS policy, we also adapt the recommendation module in each user to maximize the recommendation quality. To support such a decoupled adaptation strategy, we design a Transformer-based [171] state encoder as the backbone, which communicates the training signals between the conversation and recommendation components.

To evaluate the effectiveness of the proposed model, we compared MetaCRS with several state-of-the-art baselines for CRS on three public datasets. The results strongly demonstrated the advantage of our solution in making satisfactory recommendations to new users in CRS with a reduced number of conversations. We also conducted extensive ablation analysis on each proposed component to inspect its contribution on the improved performance: 1) the meta-exploration policy elicit informative user feedback for fast policy adaptation; and 2) the adapted recommendation component makes better recommendations by cooperating with the adapted conversation component.

3.2.2 Related works

Exploration-exploitation trade-off in CRS. CRS take advantage of conversations with users to elicit their preferences in real time for improved recommendations. The main research effort in CRS focuses on addressing the explore-exploit trade-off in collecting user feedback. The first attempt made by Christakopoulou, Radlinski, and Hofmann [30] employed multi-armed bandit models to acquire users' feedback on individual items. A follow-up study [206] set an additional bandit model to select attributes to collect user feedback and employed a manually crafted function to decide when to ask questions or make recommendations. Li et al. [113] unified attributes and items in the same arm space and let a bandit algorithm determine when to do what. Follow-up works [188, 210] also explored clustered and knowledge-aware conversational bandits.

Meta learning for recommendation. Meta learning [24, 54] has been widely used to solve the cold-start problem in recommender systems. Vartak et al. [170] studied the item cold-start problem (i.e., how to recommend new items to users). They proposed two adaptation approaches. One learns a linear classifier whose weights are determined by the items represented to the user before and adapts the classifiers' weights for each user. Another one learns user-specific item representations and adapts the bias terms in a neural network recommender for the purpose. Lee et al. [103] separated the representation layer and decision-making layer in a neural recommendation model, and executed local adaptation on the decision-making layer for each new user. Zou et al. [218] focused on interactive item recommendation, where the meta model is optimized by maximizing the cumulative rewards in each user. Kim et al. [95] deployed meta learning to online update recommender, where the meta learning rates are adaptively tuned on a per parameter and instance basis. To the best of our knowledge, we are the first to propose to tackle with cold-start CRS policy learning using meta reinforcement learning.

3.2.3 Preliminaries

In this section, we first formulate the problem of multi-round CRS as a reinforcement learning problem, and then illustrate the concept of meta reinforcement learning and how we use it to address the cold-start challenge in CRS.

Problem definition. In this work, we study the problem of multi-round conversational recommendation [105], where CRS can ask questions or make recommendations multiple times before the user accepts the recommendation or quits the conversation. Similar to traditional recommender systems, CRS face a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$; and we denote a specific user as u and a specific item as v. Each item v is associated with a set of pre-defined attributes $\mathcal{P}_v$. Attributes describe basic properties of items, such as movie genres in movie recommendations and authors in book recommendations.

We formulate the CRS problem by a Markov decision process (MDP) [47, 84, 106, 195], which can be fully described by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. S denotes the state space, which summarizes the conversation between the system and user so far. $\mathcal{A}$ denotes the action space for the system, which includes recommending a particular item or asking a specific attribute for feedback. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [-R_{max}, R_{max}]$ is a bounded reward function suggesting a user's feedback on the system's actions. As we focus on meta policy learning for CRS in this work, how to best define reward is not our objective. We follow the reward function defined in [47, 105, 106]. In particular, we include the following rewards: (1) $r_{\text{rec\_suc}}$, a large positive reward when the recommended item is accepted; (2) $r_{\text{rec\_fail}}$, a negative reward when the recommended item is rejected; (3) $r_{\text{ask\_suc}}$, a positive reward when the inquired attribute is confirmed by the user; (4) $r_{\text{ask\_fail}}$, a negative reward when the inquired attribute is dismissed by the user; (5) r_{quit}, a large negative reward when the user quits the conversation without a successful recommendation.

With this formulation, a conversation in CRS can be represented as $d = \{(a_1, r_1), ...(a_T, r_T)\}$, where T is the maximum number of allowed turns. A conversation (or an episode in the language of RL, which we will use exchangeablely) will terminate when (1) the user accepts the recommended item; or (2) the agent runs out of maximum allowed turns. At each time step t, the CRS agent, which can be fully described by a policy $\pi(a_t|s_t)$, selects an action a_t based on the current state s_t. The training objective of a CRS policy is to maximize the expected cumulative rewards over the set of observed episodes D, i.e.,

$$\mathcal{L}(\pi) = -\mathop{\mathbb{E}}_{d\sim P(D)}\big[\sum_{t=0}^{T} R_t\big],$$

where $R_t = \sum_{t'=t}^{T} \gamma^{T-t'} r(a_t)$ is the accumulated reward from turn t to the final turn T, and $\gamma \in [0, 1]$ is a discount factor to emphasize rewards collected in a near term.

Meta Reinforcement Learning for CRS. Instead of learning a single global policy π, we propose to learn personalized policy π_u for each user u (new or existing) to address the cold-start challenge for CRS. The fundamental reason that almost all previous works [47, 105, 106] focused on global policy learning is that they (implicitly) assumed users know all attributes of their desired items and share the same responses over those attributes; in other words, user feedback is fully determined by the item. This assumption is unrealistically strong and naive, since different users can describe the same item very differently, because of their distinct knowledge and preferences. For example, some users choose a mobile phone for its appearance while others choose it because of its brand. As a result, a global policy can hardly be optimal for every single user, especially the new users whose preferences are not observed during global policy training. In this work, we impose a weaker and more realistic assumption about users' decision making by allowing user-specific feedback $\mathcal{R}_{\mathcal{U}}$, which calls for personalized policies. Therefore, a personalized policy for user u should minimize,

$$\mathcal{L}_u(\pi) = -\mathop{\mathbb{E}}_{d\sim P(\mathcal{D}_u)}\big[\sum_{t=0}^{T} R_u(a_t)\big], \tag{3.15}$$

where $\mathcal{D}_u$ is a collection of conversations from user u and $R_u(a_t) = \sum_{t'=t}^{T} \gamma^{T-t'} r_u(a_{t'})$. To find the best personalized policy π_u (parameterized by θ_u) for each new user u, instead of learning from scratch every time, we choose to learn a meta policy parameterized by θ and use it as a starting point to look for θ_u. Following the convention of meta learning, we assume a set of conversations $\mathcal{D}_u^s$ (i.e., the support set) for policy adaptation , in addition to the set $\mathcal{D}_u^q$ (i.e., the query set) for policy evaluation. Hence, the size of support set $\mathcal{D}_u^s$ in each user u denotes the conversation budget for us to find θ_u when serving a single user. Given limited tolerance of an ordinary user to prolonged conversations, a performing solution should find the optimal θ_u with the size of $\mathcal{D}_u^s$ *as small as possible*. In the meta-train phase, we conduct local adaptation from the meta policy on the $\mathcal{D}_u^s$ of the existing users (i.e., training users), and then evaluate and update the meta policy on training users' $\mathcal{D}_u^q$. In the meta-test phase, we test the meta policy on the new users (i.e., testing users) by executing local adaptation on their support sets, and then test the obtained local policy on their query sets.

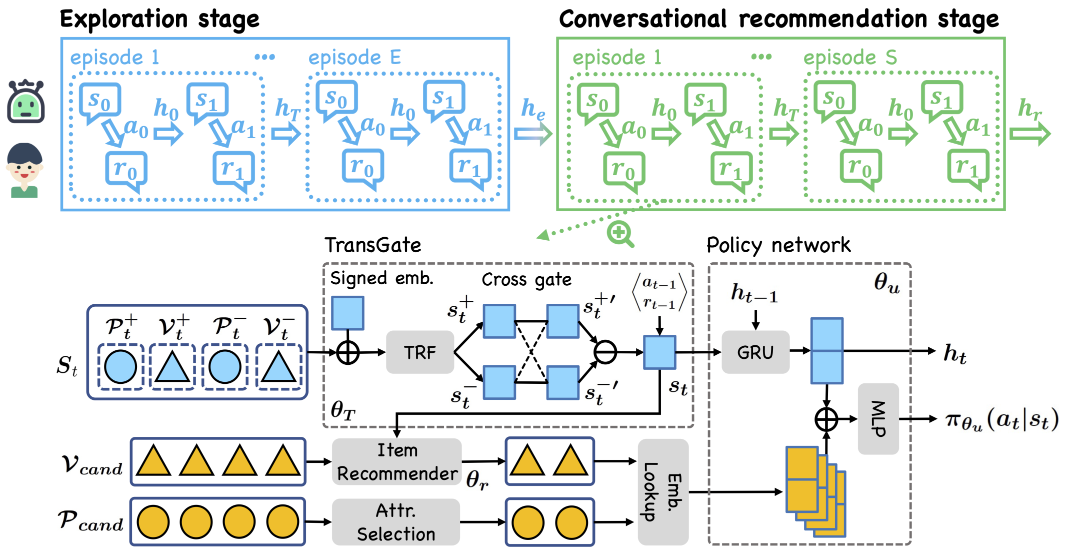


Figure 3.5: The workflow of MetaCRS training. Each user's support set is separated into the exploration stage and conversational recommendation stage. The last hidden state from the previous episode is passed to the next episode as its initial state throughout the course of MetaCRS in each user.

3.2.4 Methodology

In this section, we describe the design of MetaCRS in detail. We first introduce our two-stage meta policy learning framework designed for cold-start CRS. Then, we describe the details of the state-based item recommender, which is separately adapted to maximize the recommendation quality. The Transformer-based state encoder, which aims to rapidly capture a user's preference from her both positive and negative feedback in a conversation, is lastly explained. Fig.3.12 shows the overview of MetaCRS.

Two-stage Meta Policy Learning for CRS

Motivated by the seminal work Model-Agnostic Meta-Learning (MAML) [54], we propose to first learn a meta CRS policy from a population of users; and then for each individual user, we adapt the meta policy to a user-specific policy with only a few trails of conversational recommendations with the user. We obtain the meta policy by maximizing the policy adaptation performance in a given set of training users. Specifically, in each user u, we perform policy adaptation on her support set $\mathcal{D}_u^s$, where θ_u is initialized with θ and then updated by optimizing Eq.(3.15) via gradient descent. The gradient for Eq.(3.15) is computed by the REINFORCE [186] algorithm,

$$\nabla_{\theta_u}\mathcal{L}_u(\pi_{\theta_u}) = -\mathbb{E}\Big[\sum_{t=0}^{T} R_u(a_t)\nabla_{\theta_u}\log\pi_{\theta_u}(a_t|s_t)\Big]. \tag{3.16}$$

The gradient of the meta policy with respect to θ (i.e., $\nabla_\theta\mathcal{L}_u(\theta_u)$) is computed in the same way as Eq.(3.16), but on the corresponding query set $\mathcal{D}_u^q$. In this way, the meta policy is optimized for generalization, akin to cross-validation. Note that to exactly compute the gradient for θ, we need to take a higher-order derivative in $\nabla_{\theta_u}\mathcal{L}_u(\theta_u)$ with respect to θ on the support set as well, since θ_u is a function of θ. In this work, we followed the the first-order approximation methods proposed in [54, 124] to simplify the gradient computation.

In meta-learning for supervised learning tasks, e.g., image classification [54, 124, 174], the support set and query set are predefined and thus not affected by the learnt models. Therefore, gradient-based optimization alone is sufficient for meta model learning and adaptation. But in our problem, what we will observe in $\mathcal{D}_u^s$ and $\mathcal{D}_u^q$ are completely determined by the employed policy π_{θ_u}, which however is supposed to be derived from $\mathcal{D}_u^s$ and $\mathcal{D}_u^q$. This causes the so-called *chicken-and-egg* dilemma [115] for meta policy learning which we discussed in the introduction, and calls for additional treatments beyond gradient-based policy optimization.

Potential bias in the currently learnt policy prevents it from being effective in acquiring the most informative feedback for meta policy learning and adaptation. Hence, we propose to separate policy adaptation in each user into an *exploration* stage and a *conversational recommendation* stage, and design their corresponding policies. To avoid ambiguity, we refer to the policy for the exploration stage as the meta-exploration policy (denoted as π_{θ_e}), and the policy for the conversational recommendation stage as the CRS policy (denoted as π_{θ_u}). This is similar to the explore-then-commit strategy [57, 102] in bandit literature. But note that our meta-exploration policy is *not* personalized, as its sole goal is to quickly identify what kind of user the system is interacting with. Hence, we choose to estimate it from the whole set of training users. In particular, we reserve the first few episodes in each user's support set for our exploration stage, denoted as the exploration set $\mathcal{D}_u^e$. The size of $\mathcal{D}_u^e$ is a hyper-parameter to be tuned for different CRS applications. $\mathcal{D}_u^e$ will only be used to estimate the meta-exploration policy π_{θ_e}.

In MetaCRS, the meta-exploration policy π_{θ_e}, meta CRS policy π_θ and personalized CRS policies $\{\pi_{\theta_u}\}_{u\in\mathcal{U}}$ are realized by the same RNN-based policy network architecture [51, 80, 177] with Gated Recurrent Units (GRUs) [7, 51] to better encode the conversation history, but we estimate different parameters for them respectively. To avoid ambiguity, we will use the learning of meta-exploration policy π_{θ_e} as an illustrating example; and the same procedure applies to the learning of other policies.

Specifically, at the t-th turn of an episode, we observe a new state and encode it using a state encoder. We leave the discussion about our state encoding in Section 3.2.4. The encoded state s_t is provided as input to the policy network. The output h_t of the GRU is fed to a fully connected layer followed by a softmax function to produce the action distribution for $\pi_{\theta_e}(a_t|s_t)$. In each user, by the end of each episode, the GRU's last output hidden state h_T [2] is passed to the user's next episode as its initial state, such that this user's conversation history with the system can be continuously used to jump start her next conversation. Enabled by this design, information collected from the exploration stage is passed over to the conversational recommendation stage to profile who the user is (denoted as h_e), and then to the query set (denoted as h_r) to suggest what the user's preference could be. This sequential process is depicted in Fig.3.12.

The policy networks for π_θ and $\{\pi_{\theta_u}\}_{u\in\mathcal{U}}$ are trained via the meta learning procedure described at the beginning of this section, on top of the rewards defined in Section 3.2.3. But the meta-exploration policy π_{θ_e} is trained with a specially designed reward function, as its sole purpose is to identify what kind of user the system is serving. Inspired by [85, 91, 115], we adopt pre-trained user embeddings $\{e_u\}_{u\in\mathcal{U}}$ obtained on users with historical observations (i.e., training users) to design the exploration reward,

$$r_e(s_t) = \log P(e_u|s_t) - \log P(e_u|s_{t-1}), \tag{3.17}$$

where $P(e_u|s_t) = \frac{\exp(h_t^\top e_u)}{\sum_{u\in\mathcal{U}} \exp(h_t^\top e_u)}$. Note here we use the GRU's output hidden state h_t to

[2] If the conversation ends before the maximum turn, h_T stands for the latent state at the successful recommendation.

predict the user embedding, just as how we use it to construct the policies. Specifically, we obtain $\{e_u\}_{u \in \mathcal{U}}$ from a Factorization Machine model [105] trained on observed user-item interactions in training users. The insight behind our exploration reward design is that we promote the actions that help us identify a specific user during the exploration stage. Following the suggestion from [85, 91, 115], we also add a cross entropy loss on the meta-exploration policy network's latent state h_t to regularize the estimation of θ_e,

$$\mathcal{L}_e(\pi_{\theta_e}) = -\mathbb{E}\Big[\sum_{t=0}^{T} R_e(s_t) + \sum_{t=0}^{T} \log P(e_u|h_t)\Big], \tag{3.18}$$

where $R_e(s_t)$ is the accumulated discounted reward based on Eq.(3.17) from turn t. The gradient of the first term is also computed by the REINFORCE algorithm.

State-aware Item Recommender

Previous studies use a pre-trained recommender through the course of CRS [105, 106, 191], as their focus is mostly on deciding when to make a recommendation or otherwise what question to ask, i.e., the conversation component. A pre-trained recommender restricts the CRS policy to accommodate the recommender's behavior, which adds unnecessary complexity for policy adaptation. Such a black box design slows down personalized policy learning. For example, a user wants a phone of a specific brand, but the recommender regards *brand* as an unimportant attribute. It is difficult for CRS to recommend successfully even though the policy already elicits her preferences, which will in turn hurts the policy adaptation since the episode is failed. Hence, it is crucial to also local adapt the recommender to learn to make high-quality recommendations in cooperation with the conversation component.

In MetaCRS, we set a learnable item recommender to rank candidate items based on the state embedding from the state encoder, which will be explained in Section 3.2.4. The ranking score of an item v is calculated by,

$$w_t(v) = e_v^\top(\boldsymbol{W}_1 s_t + \boldsymbol{b}_1),$$

where $\{\boldsymbol{W}_1, \boldsymbol{b}_1\}$ are learnable parameters for the recommender, collectively denoted as θ_r; s_t is the state embedding obtained from the state encoder. We perform local adaptation on θ_r to obtain a personalized recommender, by minimizing the following cross-entropy loss once a successful conversation concludes,

$$\mathcal{L}_r(\theta_r) = -\frac{1}{T_s}\mathbb{I}(T_s \le T)\sum_{t=0}^{T_s}\log\frac{\exp(w_t(v_s))}{\sum_{|\mathcal{V}_t^+|}\exp(w_t(v))}, \tag{3.19}$$

where T_s is the index of the successful turn and v_s is the accepted item. This loss function encourages the adapted recommendation component to identify the finally accepted item as early as possible in a conversation. We denote the meta parameters of θ_r as θ_R.

TransGate State Encoder

Previous solutions [191, 207, 211] have shown the power of negative feedback in CRS state modeling. It is even more important for cold-start CRS, especially in the early stage of policy adaptation when the policy is more likely to collect negative feedback. Ineffective modeling of negative feedback will slow down policy adaption. Moreover, positive and negative feedback posits distinct information about users' preference, and thus calls for different treatments. We employ a Transformer to model such complicated relations in an ongoing conversation into a state, with a cross gate mechanism to differentiate the impact from positive and negative feedback. We name this state encoder as TransGate.

At turn t, we accumulate four kinds of feedback from a user in this conversation: (1) $\mathcal{P}_t^+$, attributes confirmed by the user; (2) $\mathcal{V}_t^+$, candidate items satisfying all accepted attributes; (3) $\mathcal{P}_t^-$, attributes dismissed by the user; (4) $\mathcal{V}_t^-$, items rejected by the user. Collectively, we denote $\mathcal{S}_t = \{\mathcal{P}_t^+, \mathcal{V}_t^+, \mathcal{P}_t^-, \mathcal{V}_t^-\}$. We first map elements in $\mathcal{S}_t$ into vectors e with an embedding layer, where attribute and item embeddings are pre-trained with training users' historical observations. Candidate items and rejected items are aggregated separately to reduce the sequence length,

$$e_{\mathcal{V}}^+ = \frac{1}{|\mathcal{V}_t^+|}\sum_{v\in\mathcal{V}_t^+} e_v^+,\ e_{\mathcal{V}}^- = \frac{1}{|\mathcal{V}_t^-|}\sum_{v\in\mathcal{V}_t^-} e_v^-.$$

In the original Transformer [171], elements are encoded with position embeddings. In our case, the order among the elements is not important, but encoding the sign of user feedback (i.e., accepted or rejected) is critical. Inspired by position embeddings, we propose to encode user feedback into signed embeddings $\{e^+, e^-\}$. We add e^+ to positive elements and e^- to negative elements in $\mathcal{S}_t$. We use the current candidate items to provide positive context. Then, we feed the obtained embeddings into L Transformer layers. For simplicity, we keep the notations of transformed embeddings unchanged. We then aggregate the positive and negative elements separately to obtain an embedding for positive feedback and an embedding for negative feedback,

$$s_t^+ = \frac{1}{1+|\mathcal{P}_t^+|}\Big(e_{\mathcal{V}}^+ + \sum_{p\in\mathcal{P}_t^+} e_p^+\Big),\ s_t^- = \frac{1}{1+|\mathcal{P}_t^-|}\Big(e_{\mathcal{V}}^- + \sum_{p\in\mathcal{P}_t^-} e_p^-\Big).$$

The positive and negative feedback embeddings may contain overlapped information, which will confuse policy learning. For example, an item that already satisfies all confirmed attributes so far can still be rejected by the user. We propose a cross gate mechanism to further differentiate the positive and negative information $s_t^{+\prime} = s_t^+ \odot g^-,\ \ s_t^{-\prime} = s_t^- \odot g^+$, where $\odot$ denotes the element-wise product and $\{g^+, g^-\}$ are defined as

$$g^+ = \sigma(\boldsymbol{W}_2 s_t^+ + \boldsymbol{b}_2),\ \ g^- = \sigma(\boldsymbol{W}_3 s_t^- + \boldsymbol{b}_3),$$

where $\sigma(\cdot)$ is the sigmoid function and $\{\boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{b}_2, \boldsymbol{b}_3\}$ are learnable parameters. We obtain the final state by $s_t = s_t^{+\prime} - s_t^{-\prime}$. The set of parameters for the TransGate encoder is denoted as θ_T, which is learnt from the conversations with training users. We should note once learnt this encoder is shared globally by all users without personalization. The state embedding is then concatenated with the encoding of $\langle a_{t-1}, r_{t-1}\rangle$ as the input to the RNN-based policy network. In particular, the action embedding is directly read off based on the pre-trained attribute and item embeddings, and we set a linear layer to encode the reward.

Algorithm 2: Optimization algorithm of MetaCRS

```
1  **Input:** User population $\mathcal{U}$, learning rates $\alpha, \beta$, meta parameters $\theta$, $\theta_e$, $\theta_R$, $\theta_T$;
2  **while** *not* Done **do**
3      Sample a batch of users $\mathcal{U}_b \sim P(\mathcal{U})$;
4      **for** each $u \in \mathcal{U}_b$ **do**
5          Collect $\mathcal{D}_u^e$ and $h_e$ by executing $\pi_{\theta_e}$ ;
6          Initialize $\theta_u = \theta$, $\theta_r = \theta_R$;
7          Collect $\mathcal{D}_u^s$ and $h_r$ by executing $\pi_{\theta_u}$ with $h_e$;
8          Evaluate $\nabla_{\theta_u}\mathcal{L}_u$ and $\nabla_{\theta_r}\mathcal{L}_r$ using $\mathcal{D}_u^s$;
9          Compute adapted parameters with gradient descent: $\theta_u = \theta_u - \alpha\nabla_{\theta_u}\mathcal{L}_u$,
            $\theta_r = \theta_r - \alpha\nabla_{\theta_r}\mathcal{L}_r$ ;
10         Collect $\mathcal{D}_u^q$ by executing $\pi_{\theta_u}$ with $h_r$;
11     **end**
12     Update $\theta, \theta_R, \theta_T$ using each $\mathcal{D}_u^q$ by minimizing $\mathcal{L}_u$, $\mathcal{L}_r$;
13     Update $\theta_e, \theta_T$ using each $\mathcal{D}_u^e$ by minimizing $\mathcal{L}_e$;
14 **end**
```

Optimization Algorithm

Now we are finally equipped to illustrate the complete learning solution for MetaCRS in Algorithm 2. In the inner for-loop, we perform policy adaption to obtain the personalized CRS policy (including item recommender). In the outer while-loop, we update all meta parameters. To simplify the gradient computation, we stop the gradients on the inherited initial hidden state h_T from the latest episode in back-propagation. In practice, we update the local parameters once an episode is executed, as we find empirically it works better than updating once after the whole $\mathcal{D}_u^s$ is finished. When serving new users in the meta-test phase, we fix $\{\theta_e, \theta_T\}$ and only execute local adaptation (the inner for-loop part in Algorithm 2) with the corresponding parameters initialized by $\{\theta, \theta_R\}$.

In each turn, we use all the candidate items $\mathcal{V}_{cand}$ (i.e., $\mathcal{V}_t^+$) and attributes $\mathcal{P}_{cand}$ to construct the action space, where $\mathcal{P}_{cand}$ is the entire attribute set excluding $\mathcal{P}_t^+$ and $\mathcal{P}_t^-$. Deng et al. [47] reported that a very large action space always slowed down policy learning. To generate a reasonable action space, we follow the manually crafted rules from [47] to select K_A attributes from $\mathcal{P}_{cand}$ and select the top-K_I items provided by the state-based item recommender.

3.2.5 Experiments

To fully demonstrate the effectiveness of MetaCRS in solving the cold-start CRS problem, we conduct extensive experiments and study the following four research questions (RQ):

- **RQ1**: Can MetaCRS achieve better performance than state-of-the-art CRS solutions when handling new users?
- **RQ2**: Does our meta reinforcement learning based adaptation strategy work better than other adaptation strategies?
- **RQ3**: How quickly can MetaCRS obtain a good personalized policy for each user?
- **RQ4**: How does each proposed component contribute to the final performance of MetaCRS?

Experiment Settings

Datasets. We evaluate MetaCRS on three multi-round conversational recommendation benchmark datasets [47, 105, 106, 207] and summarize their statistics in Table 3.4. **LastFM** [11] is for music recommendation. Lei et al. [105] manually grouped the original attributes into 33 coarse-grained attributes. **BookRec** [74] is for book recommendation. We further processed it by selecting top 35 attributes according to their TF-IDF scores across items

Table 3.4: Summary statistics of datasets.

	LastFM	**BookRec**	**MovieLens**
#Users	1,801	1,891	3,000
#Items	7,432	4,343	5,974
#Attributes	33	35	35
#Interactions	72,040	75,640	120,000
Avg. $\lvert\mathcal{P}_u\rvert$	7	8	12
Avg. $\lvert\mathcal{P}_v\rvert$	4.07	8.15	5.02
Avg. $\lvert\mathcal{P}_o\rvert$	5.44	5.30	4.25

Table 3.5: Comparison of CRS performance among models on three datasets. * stands for the best performance in each group.

		MaxE	EAR	SCPR	UNI	ConUCB	ConTS	FPAN	UR	F-FT	F-IA	UR-FT	UR-IA	MetaCRS
LastFM	SR@10	0.137	0.428	0.432	0.441*	0.237	0.270	0.508	0.641*	0.533	0.529	0.613	0.678*	**0.713**
	AT	9.71	8.62	8.70	8.52*	8.69	8.93	8.08	7.02*	8.01	8.13	7.19	6.85*	**6.18**
BookRec	SR@10	0.206	0.320	0.329	0.358*	0.181	0.243	0.397*	0.384	0.405	0.411	0.417	0.420*	**0.487**
	AT	9.64	9.01	9.11	9.00*	9.52	9.17	8.31*	8.55	8.36*	8.52	8.54	8.41	**8.06**
MovieLens	SR@10	0.262	0.552	0.545	0.596*	0.272	0.434	0.589	0.681*	0.603	0.612	0.677	0.704*	**0.745**
	AT	9.46	7.98	7.89*	8.01	8.36	8.08	7.81	7.00*	7.78	7.69	7.14	6.88*	**6.27**

and filter out items with too few attributes. **MovieLens** [71] is for movie recommendation. We performed the same pre-processing as on the BookRec dataset.

We randomly split users for training, validation and testing with the ratio 8:1:1, such that the evaluation set only contains new users. On each benchmark dataset, we obtained user, item and attribute embeddings (denoted as $e_{\mathcal{U}}, e_{\mathcal{V}}, e_{\mathcal{P}}$) using a variant of Factorization Machine (FM) proposed in [105] on observed user-item interactions in the training set. Similar to [47, 105, 106], we developed a user-simulator to generate conversations based on the observed user-item interactions in the dataset. However, the number of observed interactions in each user is not even, which may cause the learned meta policy biased toward users with more observed interactions. To better study the problem of personalized CRS policy learning, as part of our simulation, we generated 40 user-item interactions for each user by sampling items proportional to the score $e_u^\top e_v$ to augment the interaction data for our evaluation purpose. We should note such simulation design will not ease the necessity of personalized CRS policy learning, since e_u is never directly disclosed to the policy. We provide our code and generated data to facilitate follow-up research and ensure the producibility of our reported results [3].

User simulator. CRS needs to be trained and evaluated via interactions with users. Pre-

[3]`https://github.com/zdchu/MetaCRS.git`

vious simulator designs are item-centric [47, 105, 106], enforcing all users to respond in the same way to all attributes of target item v (i.e., confirming every entry in $\mathcal{P}_v$). This setting is unrealistically restrictive and eliminates the necessity of personalized policies. To demonstrate the utility of personalized CRS policy learning, we design a *user-centric* simulator that supports user-specific feedback in each conversation.

In detail, we used the pre-trained user and attribute embeddings to generate each user's preferred attribute set $\{\mathcal{P}_u\}_{u \in \mathcal{U}}$, by selecting the top-ranked attributes for each user based on the score $e_u^\top e_p$. During the course of CRS, the simulated user will only confirm the overlapped attributes in $\mathcal{P}_o = \mathcal{P}_u \cap \mathcal{P}_v$, and dismiss all others. On the BookRec dataset, because the original entries in $\mathcal{P}_v$ is too generic to be informative, i.e., too many attributes appear in almost all items, we decided to also increase $\mathcal{P}_v$ on this dataset by adding top-ranked attributes for each item based on the score $e_v^\top e_p$. We report the mean value of $|\mathcal{P}_u|$, $|\mathcal{P}_v|$ and $|\mathcal{P}_o|$ resulted from our simulation on each dataset in Table 3.4.

Baselines. To fully evaluate the effectiveness of MetaCRS, we compared it with a set of representative baselines. We categorized the baselines into three groups for different comparison purposes. In the first group, we compared MetaCRS with a rich set of state-of-the-art CRS methods to answer RQ1:

- **Max Entropy (MaxE)** is a rule-based method suggested in [105]. In each turn, the attribute with maximum entropy is to be asked or top-ranked items are to be recommended based on the rule.
- **EAR** [105] is a three-stage solution consisting estimation, action and reflection steps. It updates the conversation and recommendation components using reinforcement learning.
- **SCPR** [106] reformulates the CRS problem as an interactive path reasoning problem on the user-item-attribute graph. Candidate attributes and items are selected according to their relations with collected user feedback on the graph.
- **UNICORN (UNI)** [47] integrates the conversation and recommendation components into a unified RL agent. Two heuristics for pre-selecting attributes and items in each turn are proposed to simplify its RL training.

Baselines in this group rely on pre-trained user embeddings to make recommendations or compute states, which are not available in new users. To apply them to new users, we used the average embedding of all training users as the embedding for new users. This group of baselines are learnt on training users and then evaluated on the testing users.

In the second group, we compared MetaCRS with solutions which handle new users by updating user embeddings dynamically within a conversation, such that they can provide adaptive recommendations. We consider the following algorithms:

- **ConUCB** [206] introduces the concept of super arms (i.e., attributes) to traditional bandit algorithms. Items and attributes with the highest upper confidence bound are selected. The attributes are asked in a fixed frequency by a hand-crafted function.
- **ConTS** [113] overcomes ConUCB's limitation by replacing the hand-crafted function with a Thompson sampling procedure. The user embeddings of cold-start users are updated with users' feedback on the asked attributes and items.
- **FPAN** [191] extends the EAR model by utilizing a user-item-attribute graph to enhance offline representation learning. User embeddings are revised dynamically based on users' feedback on items and attributes in the conversation.

Our TransGate and state-based item recommender can also dynamically capture user preferences, and provide adaptive recommendations within a conversation. To further study their value in learning personalized CRS policies, we integrated them with UNICORN, and denoted this variant as **UR**, which is also included in the second group. All baselines in this group used the same pre-trained embeddings as MetaCRS. To improve the practical performance of ConUCB and ConTS, we adopted the heuristics in [47] to pre-select arms according to the similarity with accepted attributes.

To answer RQ2, we equip FPAN and UR with the ability to adapt policies on new users via the following two widely used strategies, which forms the third group of baselines:

- **Fine-tuning (FT)**: We first pre-train a global policy on all training users. During testing, we fine-tune the policy on the whole support set of all new users.
- **Independent adaptation (IA)**: We first pre-train a global policy on all training users. For each new user, we perform continual training on her support set to obtain a personalized policy.

We denoted the resulted variants as **F-FT**, **F-IA** and **UR-FT**, **UR-IA** respectively. As we found policy gradient was more effective and efficient than UNICORN's original Q-learning based algorithm in our experiments, we applied policy gradient for model update in all UNICORN-based baselines.

Evaluation metrics. We followed the widely-used metrics in previous works [47, 105, 106] to evaluate the CRS solutions. We evaluated the average ratio of successful episodes within T turns by success rate (SR@T). We also evaluated average turns in episodes (AT). A better policy is expected to recommend successfully with less turns. The length of failed conversations is counted as T.

Implementation details. We performed the training of meta policy on training users, and local adaptation on validation and testing users. We selected the best model according to its validation performance. The query sets of testing users are used to obtain the final performance for comparison. We set the rewards as: $r_{\text{rec\_suc}} = 1$, $r_{\text{rec\_fail}} = -0.1$, $r_{\text{ask\_suc}} = 0.1$, $r_{\text{rec\_fail}} = -0.1$, $r_{\text{quit}} = -0.3$. The action embedding size and the hidden size are set to be 64 and 100, while reward embedding size is set to 10. We set 1 Transformer layer in the TransGate encoder. In MetaCRS, we took 5 episodes in the exploration stage and 10 episodes in the conversational recommendation stage by default. We set K_I, K_A and K_{rec} to 10. We performed standard gradient decent in local adaptation with a learning rate of 0.01, and updated the meta parameters using the Adam optimizer with a learning rate of 0.005 and L_2 regularization coefficient 1e-6. The discount factor γ is set to 0.999. To make a fair comparison, we run 15 episodes in new user for adaptation in the second group of baselines. The size of query set is fixed to 10. The maximum turn T in each episode is set to 10. We sample 5 users in each epoch when training MetaCRS. For all baselines, we used implementations provided by the papers and modified them as described before to support cold-start evaluation.

Overall Performance

We report the comparison results across all methods in Table 3.5. We can clearly observe that MetaCRS outperformed all baselines with large margins. First of all, the results of the first group of baselines confirmed a single global policy cannot handle new users. By learning personalized policies, MetaCRS showed advantages in the final recommendation performance. In the second group of baselines, bandit-based algorithms select actions according to simple linear models, which are not capable to capture complicated relations between the algorithms' actions and user feedback, especially when positive rewards are discrete and sparse. Hence, such solutions performed much worse than other deep learning based methods. Both FPAN and UR can provide adaptive recommendations like MetaCRS, and thus they outperformed all baselines in the first group. Interestingly, we observe that FPAN outperformed EAR considerably. Different from EAR, FPAN updates user embeddings dynamically with users' positive and negative feedback on attributes and items by two gate modules, which enable dynamic item recommendation as in MetaCRS. This improved performance proves the necessity of adaptive recommendation during the course of CRS. UR showed general improvement against UNICORN and FPAN, both of which rely on a fixed FM model to recommend items (UNICORN also uses the FM model to pre-select items as actions) when performing policy training. Our state-based item recommender is able to provide improved recommendations at each time step by utilizing training signal from accepted items, which brings concrete benefits.

In the third group, with the adaptation on new users, FT and IA led to general improvement, which indicates the necessity of policy adaptation in new users. Specifically, IA outperformed FT in most cases, which again proves that personalized polices are beneficial. Even though FPAN is able to dynamically capture users' preference with its gating modules, such knowledge did not generalize well on new users, which limited the improvement in F-FT and F-IA. By adapting both conversation and recommendation components in CRS, MetaCRS and UR-IA showed improvements against other baselines, which validates the necessity of our decoupled adaptation strategy. But as UR is not trained for generalization, it is hard to find optimal personalized policies for new users starting from such a global policy in UR-FT and UR-IA, while the meta model in MetaCRS is trained for generalization. In addition, the meta-exploration policy provides useful information for fast adaptation. Thus MetaCRS is able to perform better even with fewer adaptation episodes (first 5 episodes are used to explore user preferences).

Ablation Study

Impact of support set size. Since policy adaptation is performed on the support set, it is important to study how many episodes are needed to obtain a good personalized policy (RQ3). In this experiment, we gradually increased the size of support sets with a step size 5. We kept the size of exploration episodes unchanged since 5 episodes are empirically sufficient for pinning down the target user's preference. We reported the results on the LastFM and BookRec dataset in Figure 3.6. With a larger support set, the success rate increases considerably and the number of average turn also reduces. This is expected since more observations can be collected to better adapt the meta policy for each user. And this result also demonstrates the promise of personalized CRS policy learning: the quality of recommendation increases rapidly as the users get engaged with the system, which leads to a win-win situation for both users and system.

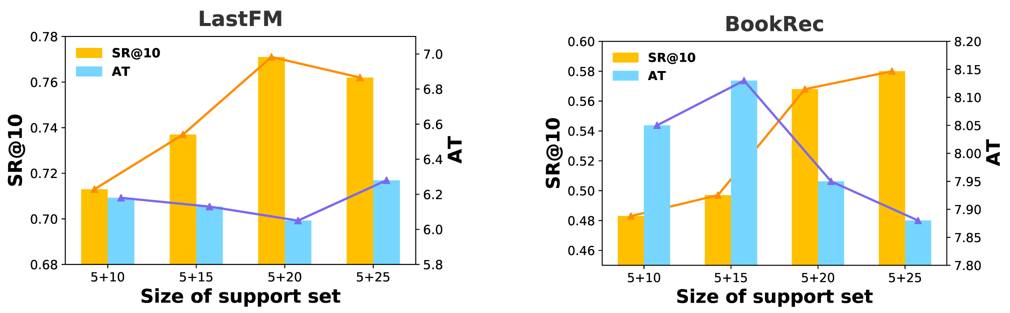


Figure 3.6: Performance comparisons w.r.t. size of support set.

Impact of different MetaCRS components. In this section, we study the contribution of different components in MetaCRS to answer RQ4. Firstly, we evaluated the model's performance without local adaptation, which essentially evaluated the learnt meta policy. Secondly, we removed the meta-exploration policy and directly executed policy adaptation. This setting shows how a dedicated exploration strategy affects policy adaptation. Finally, we replace the TransGate module with a linear layer similar to [211] to study how state representation learning affects the CRS performance. In particular, the positive and negative embeddings are obtained by taking the average of all positive and negative feedback separately.

Table 3.6: Ablation analysis in MetaCRS.

	LastFM		BookRec		MoiveLens	
	SR@10	AT	SR@10	AT	SR@10	AT
¬adaptation	0.632	6.68	0.378	8.56	0.630	7.31
¬exploration	0.677	6.64	0.411	8.37	0.738	6.65
¬TransGate	0.678	6.41	0.428	8.51	0.724	6.95
MetaCRS	**0.713**	**6.18**	**0.487**	**8.06**	**0.745**	**6.27**

We present the results in Table 3.6. Firstly, we can observe the performance before adaptation is not bad, or even better than most of our baselines in Table 3.5, which suggests the meta policy in MetaCRS already captured some important patterns for interacting with users. We can further compare the learnt meta policy with UR in Table 3.5, which shares the same state encoder and item recommender, but was trained globally. UR is slightly better than the meta policy in MetaCRS. The reason is UR is trained to maximize performance on training users and generalized by the i.i.d. assumption. But the meta policy is trained to maximize the adapted policies' performance, not its own performance on new users. Hence, when testing users share reasonable similarity with training users, UR can be effective in serving the testing users. But we can observe a large performance gain after adaptation, which proves the meta policy successfully serves as a good starting point for fast adaptation. Next, it is clear that without the exploration stage the performance degenerates. It confirms recognizing who the system is serving is critical for a successful adaptation. We finally evaluate the effectiveness of TransGate, without which the performance degenerates on all three datasets. This demonstrates the necessity of fine-grained modeling of user feedback, especially the negative feedback, for understanding users' preferences.

3.3 Meta-reinforcement learning via exploratory task clustering

3.3.1 Introduction

Conventional reinforcement learning (RL) is notorious for its high sample complexity, which often requires tremendous amount of interactions with an environment to learn a performing policy for a new task [84, 195]. Inspired by the learning process of humans, meta-reinforcement learning (meta-RL) is proposed to quickly learn new tasks by leveraging knowledge shared by related tasks [51, 54, 177]. The key research question in meta-RL is task modeling for identifying transferable knowledge among tasks. For example, Finn, Abbeel, and Levine [54] proposed to learn a set of shared meta parameters which are used to initialize the local policy when a new task arrives. Duan et al. [51] and Wang et al. [177] trained an RNN encoder to characterize prior tasks according to the interaction history in those tasks.

Little attention has been paid to the structures in the transferable knowledge resulted from task distributions. Aforementioned methods implicitly assume tasks follow a unimodal distribution, and thus the knowledge, once identified, can be broadly shared across all tasks. However, heterogeneity among tasks is not rare in practice. It therefore dwarfs simple sharing of global knowledge, but instead imposes subtle structures for identifying relatedness among tasks at a finer granularity, e.g., groups of tasks. For instance, the general skills required for the Go game and Gomoku game are related, such as familiarity with the board layout and stone colors. But to achieve mastery in either game, policies must acquire and internalize game-specific knowledge/rules to effectively navigate subsequent matches. For example, experience about competing against different human players in Go games can be shared within, but not over to Gomoku games. This heterogeneity motivates us to formulate a more delicate but also more general meta-RL setting where tasks are originated from various but a finite number of distributions, i.e., *tasks are clustered*. Hence, knowledge that benefits learning in new tasks becomes cluster-specific. We refer to this as *structured heterogeneity* among tasks, and propose to explicitly model it to facilitate cluster-level knowledge sharing[4].

Structured heterogeneity among tasks has been studied in supervised meta-learning [196]; but it is a lot more challenging to be handled in meta-RL, where the key bottleneck is how to efficiently discover task relatedness in a population of RL tasks. Different from

[4]We do not assume the knowledge in different clusters is exclusive, and thus each cluster can still contain overlapping knowledge, e.g., motor skills in locomotion tasks.

supervised learning tasks where static task-specific data is available for task relatedness inference before any learning starts, observations about RL tasks are collected by an agent's interactions with the task environment. As a result, successfully adapting an RL policy to a new task depends on accurate profiling of the task, which however is elicited by the policy itself. Task inference becomes a major bottleneck of sample efficiency in meta-RL [115]. Previous methods [37, 51] focus on task embedding learning under the uni-modal task distribution assumption, which are inefficient to infer clustered tasks. But structured heterogeneity provides new opportunities for efficient task inference: instead of directly identifying the new task, the coarse-grained cluster membership can be first inferred with a few observations; within the located task cluster, task inference can be performed in a designated search space, i.e., *divide-and-conquer* task inference.

To realize our idea of utilizing structured heterogeneity among tasks in meta-RL, we develop **MiLEt**: **M**eta re**I**nforcement **L**earning via **E**xploratory Task clus**T**ering. To the best of our knowledge, we are the first to propose a method for improving sample efficiency in meta-RL by utilizing cluster structures in the task distribution. Specifically, we perform cluster-based variational inference (CBVI) [49, 132] to infer the cluster of a new task according to its ongoing trajectory. To facilitate cluster inference, at the meta-train phase, we optimize a dedicated exploration policy based on a divide-and-conquer strategy: it first quickly explores the task's cluster assignment, and then refines its task modeling in the narrowed search space given the identified cluster. An exploitation policy is then trained to maximize the task rewards based on the refined task model from the exploration trajectory. We compare MiLEt against a rich set of state-of-the-art meta-RL solutions on various MuJoCo environments [166] with varying cluster structures in both reward and state dynamics. To test the generality of MiLEt, we further evaluate it on environments characterized by non-parametric cluster structures among tasks, i.e., the Meta-World tasks [201]. The experiment results confirm MiLEt can effectively discover clusters among tasks and then benefit fast adaptation to new tasks.

The main contributions of this paper are three-fold,

1. We present MiLEt, a novel approach to improve meta-RL by explicitly modeling cluster structures inherent in the task distribution.

2. We introduce a dedicated cluster-level exploratory policy, which employs a divide-and-conquer strategy, ensuring robust and effective discovery of task clusters.

3. We evaluate MiLEt on a rich set of environments with both parametric and non-parametric task clusters. The empirical results prove the effectiveness of MiLEt.

3.3.2 Related works

Task modeling in meta-learning. Task modeling is important to realize fast adaptation in new tasks in meta learning. Finn, Abbeel, and Levine [54] first proposed the model-agnostic meta learning (MAML) aiming to learn a shared model initialization, i.e., the meta model, given a population of tasks. MAML does not explicitly model tasks, but it expects the meta model to be only a few gradient updates away from all tasks. Later, an array of methods extend MAML by explicitly modeling tasks using given training data under the supervised meta-learning setting [104, 174]. Yao et al. [196] adopted a hierarchical task clustering structure, which enables cluster-specific meta model. Such a design encourages the solution to capture locally transferable knowledge inside each cluster, similar to our MILET model. However, task information is not explicitly available in meta-RL: since the true reward/state transition functions are not accessible to the agent, the agent needs to interact with the environment to collect observations about the tasks, while maximizing its return from the interactions. MILET models posterior distribution of a task's cluster assignment based on its ongoing trajectory; better yet, it is designed to behave exploratorily to quickly identify tasks' clustering structures, and then refine the task modeling in the narrowed search space conditional on the identified cluster.

Exploration in meta-reinforcement learning. Exploration plays an important role in meta-RL, as the agent can only learn from its interactions with the environment. In gradient-based meta-RL [54], the local policy is trained on the trajectories collected by the meta policy, and thus the exploration for task structure is not explicitly handled. Stadie et al. [155] and Rothfuss et al. [139] computed gradients with respect to the sampling distribution of the meta policy, in addition to the collected trajectories. Gupta et al. [69] also extended MAML by using learnable latent variables to control different exploration behaviors. The context-based meta-RL algorithms [51, 177] automatically learn to trade off exploration and exploitation by learning a policy conditioned on the current context. Zintgraf et al. [217] explicitly provided the task uncertainty to the policy to facilitate exploration. Zhang et al. [204] and Liu et al. [115] developed a separate exploration policy by maximizing the mutual information between task ids and inferred task embeddings. However, because all the aforementioned methods operate under the uni-modal assumption about the task distribution, their exploration strategy also becomes inferior to profile a given task under a heterogeneous task distribution. MILET first explores to identify the cluster of a task, which is expected to require fewer samples than detailed task identification; then the agent can explore task information within a refined search space for better sample efficiency.

3.3.3 Background

Meta-reinforcement learning. We consider a family of Markov decision processes (MDPs)[5] $p(\mathcal{M})$, where an MDP $M_i \sim p(\mathcal{M})$ is defined by a tuple $M_i = (\mathcal{S}, \mathcal{A}, R_i, T_i, T_{i,0}, \gamma, H)$ with $\mathcal{S}$ denoting its state space, $\mathcal{A}$ as its action space, $R_i(r_{t+1}|s_t, a_t)$ as its reward function, $T_i(s_{t+1}|s_t, a_t)$ as its state transition function, $T_{i,0}(s_0)$ as its initial state distribution, γ as a discount factor, and H as the length of an episode. The index i represents the task id, which is provided to agents in some works [115, 131, 204]. We consider a more general setting where the task id is not provided to the agent [217], as in general we should not expect the task id to encode any task-related information. Tasks sampled from $p(\mathcal{M})$ typically differ in the reward and/or transition functions. In each task, we run a *trial* consisting of $1 + N$ episodes [51]. Following the evaluation settings in previous works [54, 115, 139], the first episode in a trial is reserved as an *exploration* episode to gather information for task modeling, and an agent is evaluated by the returns in the following N *exploitation* episodes.

Inside a trial, we denote the agent's interaction with the MDP at time step n as $\tau_n = \{s_n, a_n, r_n, s_{n+1}\}$, and $\tau_{:t} = \{s_0, a_0, r_0, ..., s_t\}$ denotes the interaction history collected before time t. In the exploration episode, an agent should form the most informative trajectory τ^ψ by rolling out an exploration policy π_ψ parameterized by ψ. In the exploitation episodes, the agent executes the exploitation policy π_ϕ parameterized by ϕ (in some prior work, π_ψ and π_ϕ are the same [217]) conditioned on τ^ψ and, optionally, the history collected in the exploitation episodes τ^ϕ. The returns in exploitation episodes are computed as,

$$\mathcal{J}(\pi_\psi, \pi_\phi) = \mathbb{E}_{M_i \sim p(\mathcal{M}), \tau^\psi \sim \pi_\psi} \left[\sum_{t=0}^{N \times H} R_i\big(\pi_\phi(\tau^\psi; \tau^\phi_{:t})\big) \right], \tag{3.20}$$

where $R_i\big(\pi_\phi(\tau^\psi; \tau^\phi_{:t})\big)$ is the return of π_ϕ conditioned on τ^ψ and $\tau^\phi_{:t}$ at time step t in task M_i.

Clustered RL tasks. In this paper, we consider a more general and realistic setting, where the task distribution is multi-modal and thus forms a mixture,

$$p(\mathcal{M}) = \sum_{c=1}^{C} w_c \cdot p_c(\mathcal{M}), \tag{3.21}$$

where C is the number of mixing components (i.e., clusters) and w_c is the corresponding

[5]The terms of environment, task and MDP are used interchangeably in this paper, when no ambiguity is incurred.

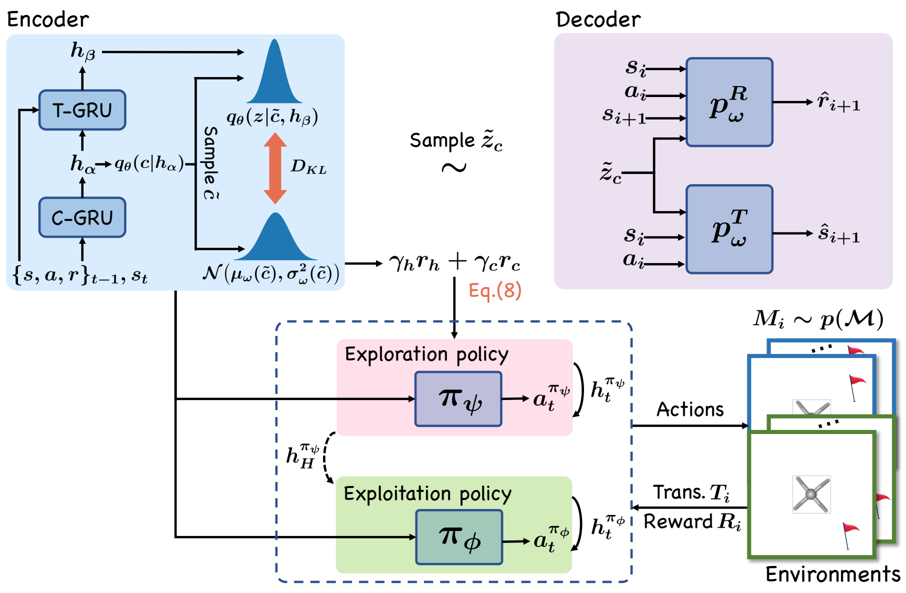


Figure 3.7: MILET architecture. The encoder processes ongoing trajectories and performs CBVI for $q_\theta(z|c, h_\beta)$. The exploration policy π_ψ is trained to find the most certain cluster assignment c when interacting with the environment. The explored information is passed to the exploitation policy π_ϕ to facilitate fast adaptation in task M_i.

weight of component c, such that $\sum_{c=1}^{C} w_c = 1$. Thus, every task is sampled as follows,

1. Sample a cluster c according to the multinomial distribution of $Mul(w_1, ..., w_C)$;

2. Sample a reward function R or a transition function T or both from $p_c(\mathcal{M})$.

The knowledge shared in different clusters could be different. For example, two clusters of distinct target positions can exist in a navigational environment, e.g., top-left vs., bottom-right. The knowledge about how an agent reaches the top-left target positions in the first cluster cannot help tasks in the second cluster; but it is crucial for learning different tasks in the first cluster. In this example, when handling a new task, a good exploration strategy should first recognize the task cluster (i.e., to move top-left or bottom-right), as it is much easier to recognize than individual tasks, and then identify the specific target position in the corresponding region of the map. This coarse-to-fine identification allows more efficient exploration of task information.

3.3.4 Methodology

In this section, we present MILET in detail, which consists of two complementary components. First, we introduce how to infer population-level task structures using the collected trajectories via cluster-based variational inference (CBVI). Then, we explain the exploration policy trained by the exploration-driven reward, which is designed to quickly identify the cluster assignment of a new task. At a high level, in each task MILET first executes the exploration policy to collect the coarse-grained cluster information; then it adapts the task policy with the help of inferred posterior cluster distribution. The architecture of MILET is shown in Figure 3.7.

Cluster-based Variational Inference with Consistency Regularization

Since the reward and transition functions are unknown to the agent, we estimate a latent random variable c_i to infer the cluster assignment of current task $M_i \sim p_c(\mathcal{M})$. Based on c_i, we infer another latent random variable z_i carrying task-level information, i.e., z_i suggests the reward/transition functions that define the task. For simplicity, we first drop the subscript i in this section, as we will only use one task as an example to illustrate our model design.

In meta-RL, all information about a given task can be encoded by z. But inferring z can be sample inefficient, as the task space can be very large. Thanks to the structured heterogeneity among tasks, inferring a task's cluster assignment c can be more sample efficient, since we should expect a much smaller number of task clusters than the number of tasks. Once c is identified, z can be more efficiently identified, i.e., divide and conquer. Hence, in MILET, when a new task arrives, we decode its characteristics by the posterior distribution $p(z, c|\tau_{:t}) = p(z|\tau_{:t}, c)p(c|\tau_{:t})$ with respect to the interaction history up to time t. The inferred task information z_c, which refers to z conditioned on c, is then provided to the policy $\pi_{\psi/\phi}(a_t|s_t, z_c)$.

Exact posterior of $p(z, c|\tau_{:t})$ defined by Eq.(3.21) is intractable. Instead, we learn an approximated variational posterior $q_\theta(z, c|\tau_{:t}) = q_\theta(z|\tau_{:t}, c)q_\theta(c|\tau_{:t})$, in which we estimate two dependent inference networks and collectively denote their parameters as θ. On top of the inference networks, we learn a decoder p_ω to reconstruct the collected trajectories. The whole framework is trained by maximizing the following objective,

$$\mathbb{E}_{\rho_\pi(\mathcal{M},\tau^+)}\Big[\log p(\tau^+|\pi)\Big], \tag{3.22}$$

where ρ_π is the distribution of trajectories induced by the policies $\pi = \{\pi_\psi, \pi_\phi\}$ within the task, and $\tau^+ = \{\tau^\psi, \tau^\phi\}$ denotes all trajectories collected in a trial, the length of which is denoted as $H^+ = (N + 1)H$. We choose to use trajectories from both exploration and exploitation episodes to best leverage information about the same underlying MDP. We omit the dependencies on π to simplify our notations in later discussions. Instead of optimizing the intractable objective in Eq.(3.22), we optimize its evidence lower bound (ELBO) w.r.t. the approximated posterior $q_\theta(z, c|\tau_{:t})$ estimated via Monte Carlo sampling [132],

$$\begin{aligned} ELBO_t = \mathbb{E}_\rho \Big[& \overbrace{\mathbb{E}_{q_\theta(z,c|\tau_{:t})}\big[\ln p_\omega(\tau^+|\tilde{z}_c)\big]}^{\text{cluster-specific reconstruction likelihood}} \\ & - \overbrace{\mathbb{E}_{q_\theta(c|\tau_{:t})}\big[\text{KL}(q_\theta(z|c, \tau_{:t}) \parallel p_\omega(z|c))\big]}^{\text{cluster-specific regularization}} \\ & - \overbrace{\text{KL}(q_\theta(c|\tau_{:t}) \parallel p(c))}^{\text{cluster regularization}} \Big], \end{aligned} \tag{3.23}$$

where $p_\omega(z|c) = \mathcal{N}\big(\mu_\omega(c), \sigma^2_\omega(c)\big)$ is a learnable cluster-specific prior, which is different from the simple Gaussian prior used in single-mode VAE [97]. This prior allows MILET to capture unique characteristics of each cluster. $p_\omega(z|c)$'s parameters are included in ω since the cluster structure is also part of the environment. $\tilde{z}_c$ is the latent variable sampled from $q_\theta(z|c, \tau_{:t}) = \mathcal{N}\big(\mu_\theta(c, \tau_{:t}), \sigma^2_\theta(c, \tau_{:t})\big)$, using the reparameterization trick [97]. $q_\theta(c|\tau_{:t})$ outputs the approximated posterior cluster distribution given $\tau_{:t}$[6]. $p(c)$ is the prior cluster distribution of tasks; when no specific prior knowledge is known about the task, we choose a fixed non-informative multinomial distribution for it. Intuitively, if discrete structures (i.e., clusters) exist in the task distribution, a uniform $q_\theta(c|\tau_{:t})$ will cause low reconstruction likelihood; thus collapsed posterior, i.e., clustering, is preferred.

Similar to [217], the first term $\ln p_\omega(\tau^+|\tilde{z}_c)$ in Eq.(3.23) can be further factorized as,

$$\begin{aligned} \ln p_\omega(\tau^+|\tilde{z}_c) = & \ln p(s_0|\tilde{z}_c) + \sum_{i=0}^{H^+-1} \Big[\ln p_\omega(s_{i+1}|s_i, a_i, \tilde{z}_c) \\ & + \ln p_\omega(r_{i+1}|s_i, a_i, s_{i+1}, \tilde{z}_c)\Big], \end{aligned}$$

where $p(s_0|\tilde{z}_c)$ is the initial state distribution in a task, and we consider it as a constant by assuming identical distribution of the initial states across clusters. The second and third terms are likelihood derived from the decoders for transition and reward functions. The density functions of $p_\omega(s_{i+1}|s_i, a_i, \tilde{z}_c)$ and $p_\omega(r_{i+1}|s_i, a_i, s_{i+1}, \tilde{z}_c)$ are difficult to estimate in continuous state and action spaces. Following [6, 204], we use L2 distance to approximate the log-likelihood functions.

In the inference networks $q_\theta(z|\tau_{:t}, c)$ and $q_\theta(c|\tau_{:t})$, we follow [51, 217] to encode the history $\tau_{:t}$ by Gated Recurrent Units (GRUs) [19, 39, 187]. We propose a stacked GRU

[6]We use the Gumbel-softmax trick to simplify the calculation.

structure (shown in Figure 3.7) to differentiate the information for cluster and task inference in the hidden space. Specifically, we set a task-GRU (T-GRU) and a cluster-GRU (C-GRU), both of which encode the history $\tau_{:t}$, but with different levels of granularity. T-GRU is set to capture fine-grained task-specific patterns in the history, as it is optimized to reconstruct trajectories of a specific task. C-GRU captures coarser-grained patterns beyond tasks, as it is set to help T-GRU reconstruct all trajectories within a cluster. To realize this difference, the output h_β of T-GRU is only provided to $q_\theta\big(z|h_\beta(\tau_{:t}, h_\alpha), c\big)$, while the output h_α of C-GRU is passed to both cluster inference $q_\theta\big(c|h_\alpha(\tau_{:t})\big)$ and task inference $q_\theta\big(z|h_\beta(\tau_{:t}, h_\alpha), c\big)$. This also reflects our dependency assumption about the task structure: cluster assignment determines tasks. We denote $h = \{h_\alpha, h_\beta\}$, which is passed across episodes in a trial.

The trajectory data is incrementally collected by the agent in meta-RL, which brings both challenges and opportunities for cluster inference. First, inside a trial, the inference improves as more observations are collected, which means the agent's belief about the ongoing task could change thereby. This is problematic, since the cluster inference result should stay consistent within a given task, no matter how trajectory changes over episodes. We attribute this property as *in-trial* consistency, which is measured by KL$(q(c|\tau_{:t_1}) \parallel q(c|\tau_{:t_2}))$, where t_1 and t_2 refer to two arbitrary timestamps in a trial. We enforce the notion of cluster inference consistency via the following regularizer,

$$\mathcal{L}_{\mathrm{I}} = \frac{1}{H^+ - 1} \sum_{t=0}^{H^+-1} \mathrm{KL}\big(q_\theta(c|\tau_{:t}) \parallel q_\theta(c|\tau_{:t+1})\big). \tag{3.24}$$

Similarly, since the cluster-specific prior $p_\omega(z|c)$ is learnable, the task inference can become inconsistent if $p_\omega(z|c)$ changes drastically across training epochs. More seriously, oscillation in the inference of latent variable z can cause the collapse of policy training, as tasks across clusters might be assigned with the same latent variable z across different training epochs. We conclude it as the *prior* consistency requirement and enforce it via the following regularization,

$$\mathcal{L}_{\mathrm{P}} = \frac{1}{C} \sum_{c=1}^{C} \mathrm{KL}\big(p_\omega(z|c) \parallel p_{\mathrm{tgt}}(z|c)\big), \tag{3.25}$$

where $p_{\mathrm{tgt}}(z|c)$ is a target network and its parameters are the same as $p_\omega(z|c)$ but updated in a much slower pace. We finally obtain the objective in CBVI as follows,

$$\mathcal{J}(\theta, \omega) = \mathbb{E}_{p(\mathcal{M})}\Big[\sum_{t=0}^{H+} ELBO_t - \lambda_{\text{I}}\mathcal{L}_{\text{I}} - \lambda_{\text{P}}\mathcal{L}_{\text{P}}\Big], \tag{3.26}$$

where λ_{I} and λ_{P} are hyper-parameters to control the strength of two regularizers.

Exploration via Reducing Inference Uncertainty

In MILET, policy adaptation in a new task has two objectives: (1) explore cluster structure; (2) explore task-specific information to solve the task. As we explained before, MILET follows a divide-and-conquer principle to realize these two objectives, which is implemented by learning two separate policies as shown in Figure 3.7. One takes exploratory behaviors to collect cluster and task information, i.e., the exploration policy π_ψ. The other is optimized to solve the task with the collected information, i.e., the exploitation policy π_ϕ.

We train a dedicated exploration policy to provide a good basis for task-solving, where cluster structures provide informative hints about task relatedness. The quality of exploration is evaluated by two principles. First, whether the trajectory of an exploration episode can reduce the uncertainty of cluster inference. Second, whether the inference result is consistent. We conclude them as *certain* and *consistent* exploration. To realize these two principles, we introduce two intrinsic rewards to encourage certain and consistent inference results. First, we use the entropy of cluster inference network $q_\theta(c|\tau^\psi_{:H})$ to measure the *uncertainty* of the inferred cluster. For a new task, we look for trajectories that provide the most certain cluster inference. We formalize the objective as follows, omitting the subscript θ and ψ for simplicity,

$$H(q(c|\tau_{:H})) = -\mathbb{E}\big[\ln q(c|\tau_0) + \sum_{t=0}^{H-1} \ln \frac{q(c|\tau_{:t+1})}{q(c|\tau_{:t})}\big].$$

We then define an intrinsic reward of each action by telescoping the second term similar to [115, 204],

$$\begin{aligned} r_h(a_t) =& \mathbb{E}\big[\ln \frac{q(c|\tau_{:t+1} = [s_{t+1}; a_t; r_t; \tau_{:t}])}{q(c|\tau_{:t})}\big] \\ =& H(q(c|\tau_{:t})) - H(q(c|\tau_{:t+1})). \end{aligned}$$

This reward favorites actions which can reduce the entropy of cluster inference; and therefore, a trajectory leading to a consistent cluster inference is preferred. To more explicitly measure the divergence between the posterior cluster distributions in two steps, we define another reward encouraging consistent cluster inference,

$$r_c(a_t) = -\mathrm{KL}(q(c|\tau_{:t}) \parallel q(c|\tau_{:t+1})).$$

Intuitively, given the inferred cluster, the exploration policy can focus on identifying task-level information within a narrowed search space, i.e., divide-and-conquer. We define the following composed reward to encourage this coarse-to-fine exploration behavior,

$$r_e(a_t) = r(a_t) + \gamma_h(t) r_h(a_t) + \gamma_c(t) r_c(a_t), \tag{3.27}$$

where $r(a_t)$ is the environment reward. $\gamma_h(t)$ and $\gamma_c(t)$ are two temporal decaying functions,

$$\gamma_h(t) = b_h - a_h \exp(-s_h(H-t)), \tag{3.28}$$

$$\gamma_c(t) = -b_c + a_c \exp(-s_c(H-t)), \tag{3.29}$$

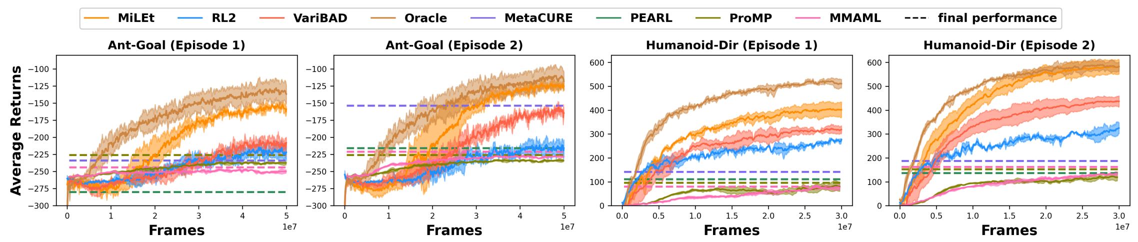
where $\{a, b, s\}_{h,c}$ are hyper-parameters controlling the rate of decay. $\gamma_h(t)$ should gradually decrease to 0, which encourages the policy to find a certain cluster at the early stage. $\gamma_c(t)$ gradually increases from a negative value to positive. At the early stage, a negative $\gamma_c(t)$ encourages the policy to try different clusters. Later, a positive $\gamma_c(t)$ enforces the policy to stick to the current cluster and focuses more on discovering task information by maximizing raw rewards.

Finally, the exploitation policy π_ϕ inherits the hidden state $h_H^{\pi_\psi}$, which encodes knowledge collected by the exploration policy, and is then trained to maximize the expected reward defined in Eq.(3.20).

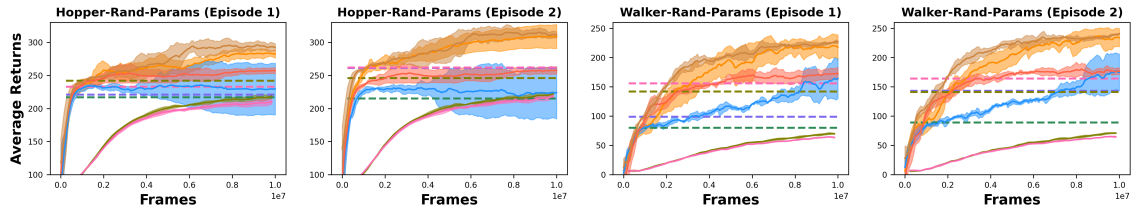
3.3.5 Experiments

In this section, we conduct extensive experiments to study the following research questions: (1) Can MiLEt achieve better performance than state-of-the-art meta-RL algorithms by exploring structured heterogeneity in the task distribution? (2) Can MiLEt effectively discover cluster structures in both rewards and state dynamics? (3) How does the number of clusters affect the final performance of MiLEt?

Environment setup. We evaluated MiLEt on two continuous control tasks with *clustered reward functions*, simulated by MuJoCo [166]. In **Ant-Goal**, the ant robot is set to move to a predetermined goal position. We created 4 clusters of the goal positions in 4 different centered areas. In **Humanoid-Dir**, the human-like robot is controlled to move towards different target directions. We created 4 clusters by distributing target directions along



(a) Environments with clustered reward functions.

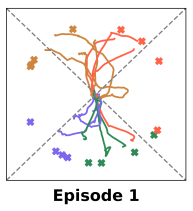


(b) Environments with clustered state transition functions.

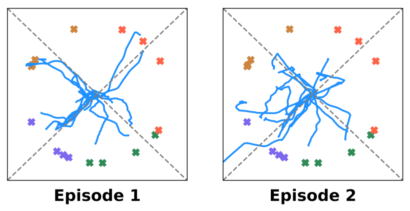
Figure 3.8: Average test performance for 2 episodes on MuJoCo environments.

4 farthest apart directions in a 2D space. We also created environments with *clustered transition functions* by adopting two movement environments **Hopper-Rand-Params** and **Walker-Rand-Params**, also simulated by MuJoCo. The physical parameters of the robot, including *body mass*, *damping on degrees of freedom*, *body inertia* and *geometry friction*, were manipulated to realize different transition functions of the robot's movement. The hopper and walker robots are set to move smoothly under different parameter settings. We created 4 clusters by manipulating one of the parameters at a time and keeping the others to the default parameters.

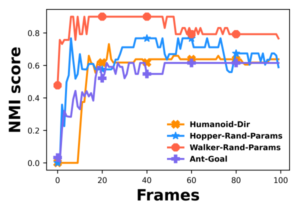
Baseline setup. We compared MiLEt with several representative meta-RL baselines, including **RL^2** [51], **PEARL** [131], **VariBAD** [217], **MetaCURE** [204], **ProMP** [139] and **MMAML** [174]. We also included an **Oracle** model, where we trained a separate VariBAD model for each ground-truth cluster. We used implementations of baselines provided by the original papers. For each environment, we created 500 tasks for meta-train and hold out 32 new tasks for meta-test. We report the performance on test tasks during the meta-train phase. In the meta-test phase, we executed 2 episodes in each new task. For algorithms with an explicit exploration policy, i.e., MiLEt and MetaCURE, we run their exploration policy in the first episode and exploitation policy in the second episode. We trained MiLEt via Proximal Policy Optimization (PPO) [144] and set the default cluster number C to 4. Because PEARL and MetaCURE are based on off-policy algorithms [70], they need less frames of data to converge in meta-train. We terminated them once the algorithm was converged and reported the final performance obtained by the moment. We report the averaged



(a) MiLEt traces.



(b) VariBAD traces.



(c) NMI score.

Figure 3.9: Qualitative analysis of MiLEt. (a) Traces of MiLEt on the meta-test tasks of Ant-Goal. Cross marks represent goal positions, and the colors represent the clusters assigned by MiLEt. The dashed lines suggest the optimal traces to the centers of ground-truth clusters. (b) Traces of VariBAD on the same meta-test tasks of Ant-Goal. The traces are in the same color as VariBAD is unaware of clusters. (c) NMI of MiLEt's inferred clusters in the exploration episode of meta-test tasks.

performance over 3 random seeds.

Results and analysis. Figure 3.8 shows the test performance of all evaluated meta-RL algorithms. We also provide qualitative analysis in Figure 3.9, including visualization of the models' behaviors and the clustering performance of MiLEt in the exploration episode, measured by the normalized mutual information score (NMI).

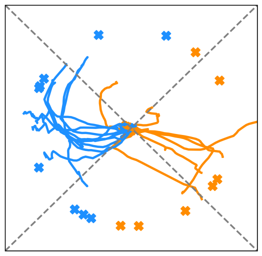
First, we clearly observed Oracle performed the best in both episodes. By directly utilizing shared knowledge within correct clusters, Oracle is able to fast adapt to individual tasks. It shows the necessity of accurate cluster modeling for fast adaptation. MiLEt showed significant improvement against baselines in the second episode in testing, approaching the performance of Oracle. Interestingly, we can observe even though the first episode of MiLEt was reserved for exploration, it still performed comparably to other methods in all four different environment setups. In the first episode, MiLEt behaved exploratorily to find the most probable cluster of the current task, and thus its traces in Figure 3.9a look like spirals from the starting point. VariBAD is also designed to explore by uncertainty in task inference, but its traces were close to random walk at the early stage, which is less effective. In Figure 3.9c, we can observe the NMI scores of the MiLEt's inferred tasks have almost converged in 20 steps, which means the cluster inference became stable in an early stage and can thereby provide the agent helpful cluster-level information to gain fine-grained task information. This also explains how MiLEt obtained comparable performance in the first episode. In the second episode, with cultivated task information, MiLEt is able to move towards the targets directly, showing significant improvements against baselines. MetaCURE guides the exploration by task IDs, which in fact provides more information of environment than what MiLEt can access. However, the exploration

	Ant-Goal	Ant-U
VariBAD	-168.6±9.6	-162.4±9.2
MILET-2	-132.3±7.6	-128.6±8.8
MILET-4	-125.4±5.1	-113.7±4.8
MILET-6	-123.6±4.4	-99.7±5.2
MILET-8	-124.2±4.7	-117.9±5.7
MILET-10	-128.6±5.2	-142.7±10.4

Table 3.7: Results on Ant-Goal and Ant-U.

empowered by task IDs does not explicitly explore the coarser but useful information at the cluster level. Both ProMP and MMAML are gradient-based methods, we found they need 5 times samples to converge, thus we also reported the final performance. Importantly, MMAML, tailored for multi-modal tasks, faces challenges during exploration. It relies heavily on its meta-policy to explore task-specific information and then formulates task embeddings. If exploration is suboptimal, the final performance suffers.

Influence of the number of clusters. We also studied how the number of clusters C set by the agent influences the final performance, especially when there is a mismatch between the ground-truth cluster size and C set by the agent. We set C to different values and denote it in suffixes of MILET. We additionally created a set of tasks on Ant-Goal, where the goal positions were uniformly sampled. We denote it as **Ant-U**.


(a) MILET-2 on Ant-G.

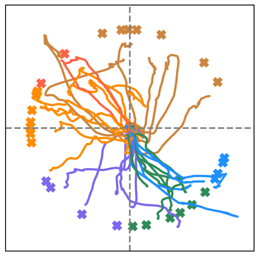

(b) MILET-6 on Ant-U.

Figure 3.10: Traces of MILET-2 and -6 in exploration episodes. Colors represent the assigned clusters.

The average final returns are shown in Table 3.7. Interestingly, we observe MILET can perform well even though there is no explicit cluster structure in Ant-U. By looking into the detailed trajectories, we found MILET segmented the circle into different parts as shown in Figure 3.10b such that knowledge from nearby tasks can be effectively shared. VariBAD mistakenly assumed all tasks can share knowledge and thus failed seriously. When C is set smaller than the ground-truth number of clusters, MILET-2 discovered more general struc-

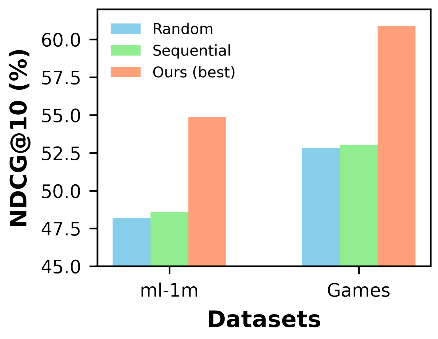


Figure 3.11: LLM-based sequential recommendation baselines show comparable performance even when historical interactions (Sequential) order is randomized (Random). `Tempura` significantly boosts performance by utilizing historical orders, *i.e.*, temporal information.

tures (as shown in Figure 3.10a). However, transferable knowledge within such structures is limited as distinct clusters are merged, causing the performance drop. Also, it does not mean more clusters than necessary is helpful, as less knowledge could be shared in each cluster.

3.4 Improve temporal awareness of LLMs for sequential recommendation

3.4.1 Introduction

Large language models (LLMs) such as ones with commercially available APIs including ChatGPT [1] and Claude[7] have emerged as one of the primary, if not the de facto, choices in a wide range of applications thanks to their remarkable capabilities in dealing with natural language and generalizing to various domains without further fine-tuning. In deed, an emerging trend is to use natural language as a uniform interface and leverage the LLMs to complete a task.

Following this trend, recent research has been exploring the use of LLMs for processing sequential data, with applications such as sequential recommendation (SRS) [9, 83], which require LLMs to comprehend temporal patterns within user historical interactions. In the case of sequential movie recommendation, historical interactions such as users' movie watching records can be represented as natural language (i.e., movie titles and other meta data) for the LLMs to process and recommend the next movie, instead of item identifiers which are typically used in traditional recommender systems [92, 158].

[7]`https://www.anthropic.com/index/claude-2`

The extensive generalization ability and vast world knowledge [151, 175] of LLMs endow them with the potential to serve as a single model for many recommendation domains without fine-tuning, making it a general, capable, and easy-to-use alternative to traditional recommender systems that usually specialize in one selected domain and require extensive training or fine-tuning.

However, recent research shows that LLMs exhibit a limited sensitivity to temporal information in the input text, particularly in discerning changes in user interests [83]. In Figure 3.11, we compare the recommendation performance of LLM-based methods using randomized (denoted as **Random**) versus correctly ordered (denoted as **Sequential**) historical interactions on two widely-used SRS datasets. Both methods show similar performance, suggesting that LLMs are not effectively utilizing the temporal information present in the input text. This limitation stems from a lack of specialized mechanisms within LLMs to automatically recognize and utilize temporal information, which is crucial for understanding the context and progression within the data.

In this paper, we focus on improving LLMs' awareness and interpretation of temporal information, particularly within the SRS scenario. Temporal information is ubiquitous in real-world applications, such as recommender systems [120], intelligent document processing [55] and financial market analysis [167]. By effectively capturing and integrating this temporal aspect, we have the opportunity to significantly enhance the understanding of user preferences via LLMs, thus providing users with better recommendations that suit their backgrounds, needs, and preferences. This improvement is also important for boosting the effectiveness of LLMs in downstream applications, where accurate user preference modeling is crucial [120]. To this end, we design a principled prompting framework inspired by human cognitive process, which is training-free and domain agnostic. We name our approach as **`Tempura`** (phonetically similar to *Temporal Prompt*). Our main contributions are:

- We propose a principled method to construct in-context examples [121] for sequential recommendation, by analyzing how Transformer-based SRS models (e.g., Kang and McAuley [92]) learn to utilize temporal information.
- Inspired by the results in neuroscience [65, 125], we add explicit structure analysis in input sequences as additional prompts, particularly temporal cluster analysis, to enhance the temporal understanding capabilities of LLMs.
- We emulate the process of *divergent thinking* [140] by aggregating ranking results derived from various prompting strategies.
- We evaluate our method on MovieLens-1M and Amazon Review datasets, the results

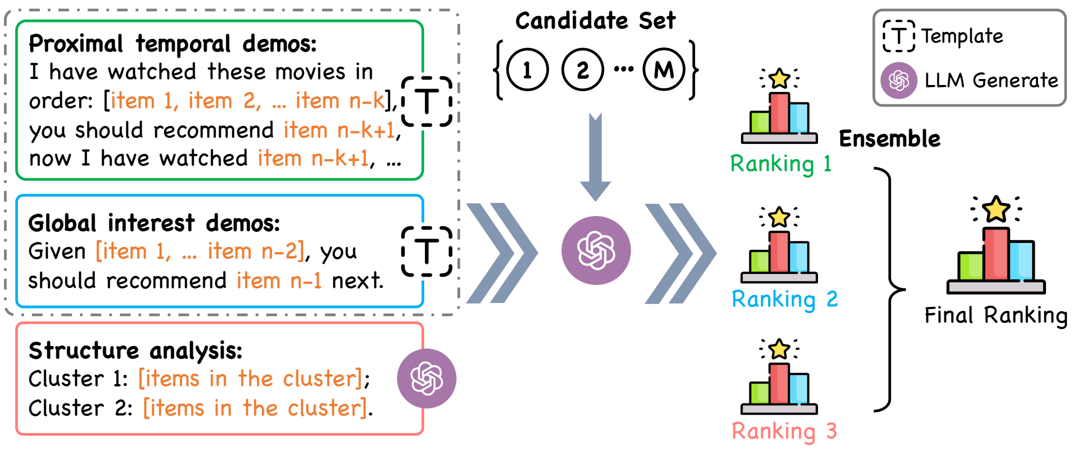

Proximal temporal demos:
I have watched these movies in order: [item 1, item 2, ... item n-k], you should recommend item n-k+1, now I have watched item n-k+1, ...
Global interest demos:
Given [item 1, ... item n-2], you should recommend item n-1 next.
Structure analysis:
Cluster 1: [items in the cluster];
Cluster 2: [items in the cluster].
Candidate Set
1
2
M
T
Template
LLM Generate
Ensemble
Ranking 1
Ranking 2
Ranking 3
Final Ranking

Figure 3.12: An illustrative overview of `Tempura`. We learn sequential recommendation via two kinds in-context demonstrations. Explicit cluster structure analysis is conducted to improve the temporal understanding capabilities of LLMs. Each prompting strategy independently generates a respective ranking by LLMs (marked by different colors). Rankings from different prompting strategies are aggregated to form the final ranking.

show that our proposed method significantly enhances the zero-shot capabilities of LLMs in sequential recommendation tasks.

3.4.2 Related Works

LLMs for recommendation. Recently, the use of LLMs in recommendation systems has garnered significant research interest due to their capability to comprehend and encapsulate a user's preferences and past interactions through natural language [53, 76]. Current LLM-based recommender systems are primarily designed for rating prediction [9, 93] and sequential recommendation tasks [83, 179, 192]. In both tasks, a user's previous interactions with items, along with other optional data like the user profile or item attributes, are concatenated to formulate a natural language prompt. This is then fed into an LLM with options for no fine-tuning [179], full-model fine-tuning [28] or parameter-efficient fine-tuning [9]. Liu et al. [116] designs a series of prompts to evaluate ChatGPT's performance over five recommendation tasks. Wang et al. [181] develops a ChatGPT-based agent to improve recommendation ability by using tools such as SQL and Web search. Contrary to existing works that focus on the tentative evaluation of LLMs' ability in recommendation, we focus on improving the LLM's inefficacy of utilizing temporal information by designing temporal-aware prompting strategies.

Sequential recommendation. Sequential recommendation (SRS) [77, 92] aims to predict the next interacted items based on historical interaction sequences. Early works follow the Markov assumption [135], by designing various neural network models to capture user preference within interaction sequences, including Recurrent Neural Network [77, 109], Convolutional Neural Network [162], Transformer [92, 158], Graph Neural Network [23, 190]. However, most of these approaches are developed based on item IDs [92] or attributes [205] defined on specific domains, making it difficult to be generalized to other domains. Recently, Hou et al. [81], Hou et al. [82] and Li et al. [108] propose to learn unified item representations for SRS based on pretrained language models. They follow the paradigm that pretraining an unified text-based sequence encoder on source domains and then fine-tune the encoder on the target domain. However, all aforementioned methods need massive user interaction sequences on a specific domains and can not be easily transfer to unseen domains. In contrast, we propose utilizing LLMs to establish a domain-agnostic learning process for sequential recommendation systems. Our approach is training-free and readily generalizable to unseen domains using only prompts.

3.4.3 Methodology

In this section, we introduce `Tempura` in detail. As shown in Figure 3.12, `Tempura` consists of three major components: 1) a in-context learning module that learns sequential recommendation tasks from sequences of historical interactions; 2) a temporal structure analysis module that enhances the model's understanding by explicitly integrating cluster structures within the sequences; 3) a prompt ensemble module that aggregates recommendation results from various prompting strategies. We begin with the definition of notations to be used in our technical discussions.

Problem Definition

Given a user's historical interactions $\mathcal{H} = \{i_j\}_{j=1}^{n}$, ordered chronologically up to timestamp n, the task of sequential recommendation involves ranking a set of *candidate* items $\mathcal{C} = \{i_j\}_{j=1}^{m}$ for the subsequent timestamp $n + 1$. Items of higher interest are expected to be ranked at more prominent positions. In practice, candidate items are typically selected from the entire item set $\mathcal{I}$, where $m \ll |\mathcal{I}|$, through candidate generation models [41]. Further, we follow the approach of Hou et al. [82] by associating each item i with a descriptive text t_i, which could be the item's name and its attributes or properties.

Different from training-based SRS models, we leverage general-purpose LLMs (e.g., ChatGPT) to solve the recommendation task in an instruction-following paradigm [182]. Specifically, for each user, we construct a history prompt from the user's historical interactions $\mathcal{H}$, and a candidate item prompt from the candidate item set $\mathcal{C}$. The aforementioned prompts are concatenated along with an instruction that explicitly describes the recommendation task, forming the final prompt for LLMs. LLMs are anticipated to generate rankings of $\mathcal{C}$, reflecting user preferences, in accordance with the format specified by the instruction. A post-hoc text parser is employed to convert the natural language rankings generated by LLMs into structured ranked lists, which is used to calculate the ranking metrics [83].

Sequential Recommendation via In-Context Learning

Given the vast scale of LLMs, fine-tuning domain-specific models becomes impractical. Thus, we propose to learn sequential recommendation via in-context learning, offering a training-free approach that can be easily adapted across various domains by leveraging the world knowledge and comprehension capabilities of LLMs [72, 83]. To this end, we first analyze the learning process of training-based SRS models, and then mapping it onto the principles of constructing effective in-context demonstrations.

The key distinction between SRS and other recommender systems lies in the SRS model's requirement to not only identify a user's preferences based on historical user-item interactions but also to track the evolution of the user's interests over time. Training-based SRSs depend on learning from large-scale user-item interaction data via GRUs [77] or Transformers [158]. We utilize In-Context Learning (ICL) [121] as a training-free alternative to learn a SRS model. We follow Dai et al. [42] to analyze the learning process of training-based SRSs. Given the historical interaction sequence of an user, a trained Transformer-based SRS, such as SASRec [92], can be represented as,

$$\mathcal{F}_{\text{SASRec}}(\boldsymbol{x}_n) = (W_0 + \Delta W)\boldsymbol{x}_n. \tag{3.30}$$

where W_0 is the initialized parameter matrix, ΔW is the update matrix and $\boldsymbol{x}_n$ is the representation of a candidate item. The output of $\mathcal{F}_{\text{SASRec}}$ is the score of the examined candidate item. In the back-propagation algorithm, ΔW is computed by accumulating the outer products of historic item representations $\boldsymbol{x}_i'^T$ and the error signals $\boldsymbol{e}_i$ of their corresponding outputs:

$$\Delta W = \sum_{i=1}^{n-1} \boldsymbol{e}_i \otimes \boldsymbol{x}_i', \tag{3.31}$$

where error signals $\boldsymbol{e}_i$ is the prediction error on the historic item $\boldsymbol{x}_i'$. Thus, the trained

SASRec can also be rewritten into,

$$\begin{aligned}
\mathcal{F}_{\text{SASRec}}(\boldsymbol{x}_n) &= (W_0 + \Delta W)\boldsymbol{x}_n \\
&= W_0\boldsymbol{x}_n + \sum_{i=1}^{n-1}(\boldsymbol{e}_i \otimes \boldsymbol{x}_i')\boldsymbol{x}_n \\
&= W_0\boldsymbol{x}_n + \text{LinAtt}(E, X', \boldsymbol{x}_n),
\end{aligned}$$

where LinAtt($V, K, \mathbf{q}$) denotes the linear attention operation, in which we regard error signals E as values and interacted items X' as keys, and the current input $\boldsymbol{x}_n$ as the query. The learning process of the SASRec model can be expained as the model predicting the next item in a sequence based on preceding items and updating itself based on the prediction error. The trained SASRec model is designed to update user preferences as the sequence expands, effectively tracking the evolution of the user's interests.

Let $\mathbf{q} = W_Q\boldsymbol{x}_n$ represent the attention query vector for the input candidate item $\boldsymbol{x}_n$. An ICL-based SRS can be represented as,

$$\mathcal{F}_{\text{ICL}}(\mathbf{q}) = (W_{\text{ZSL}} + \Delta W_{\text{ICL}})\mathbf{q}$$

where $W_{\text{ZSL}} = W_V X (W_K X)^T$ is the initialized parameters to be updated and $W_{\text{ZSL}}\mathbf{q}$ is the attention result in zero-shot learning (ZSL) setting, where no demonstration are given. X denotes the input representations of query tokens before $\boldsymbol{x}_n$, such as the task description of sequential recommendation. Based on the results of Dai et al. [42], the second term can be rewritten into,

$$\Delta W_{\text{ICL}}\mathbf{q} = \text{LinAtt}(W_V X', W_K X', \mathbf{q}),$$

where X' denotes the input representations of demonstrations. Here we observe a similar form between $\mathcal{F}_{\text{SASRec}}$ and $\mathcal{F}_{\text{ICL}}$, where $W_V X'$ can be explained as the error signal from historic items. This analogy illustrates that by utilizing historic items as in-context demonstrations, an LLM can learn to capture the temporal information within the sequence of historical interactions. Hou et al. [83] discussed using the last item in the history as an in-context demonstration. Based on our analysis, this method is equivalent to training the SASRec model solely with the last historical interaction, a practice insufficient for capturing the dynamic nature of historical interactions. Thus, we are motivated to use several historical interactions as demonstrations to improve the temporal awareness of LLMs.

Proximal temporal demonstrations (PCL). Based on the above principle, we design the following prompt to learn to capture temporal information via ICL,

Proximal temporal demonstrations

I have watched these movies in order: [item 1, item 2, ... item n-k], you should recommend item n-k+1, now I have watched item n-k+1, ...
Now recommend a new movie to me.

Placeholders, highlighted in orange, structure the input for our model. The first placeholder captures the initial $n - k$ historic items, serving as the *context* for inferring user preferences. The subsequent placeholder is designated for the $n - k + 1$ item, illustrating the next item to be recommended based on the current context. Following this, we inform the LLM that the $n-k+1$ item has been interacted with, indicating that the $n-k+2$ item is the next recommendation target. This setup is repeated to create k-shot demonstrations. We utilize the most recent k items as demonstrations to capture the proximal interest of the user. We denote this prompting strategy as PCL.

Global interest demonstrations. In previous studies [83, 92], the number of historic items was constrained by the limited input length of models. Thus the whole interaction history is typically truncated and the most recent items are remained. Empirically, we also observed that extending the context window has limited impact on improving performance and may even detract from it. The reason could be: 1) the prolong context distract LLMs [117]; 2) too old history has little impact on the current user interest in the SRS scenario. However, simply omitting distant historic items risks overlooking users' long-term interests. Hence, we randomly sample a subset of historic items from the whole history sequence to retain user's global interest. Specifically, we use the same template as PCL, but the context is filled with randomly sampled historic items. Similarly, we incorporate the most recent items as in-context examples. We denote this prompting strategy as GCL.

Temporal Structure Analysis

It has been recognized in the neuroscience area that the human brain is more sensitive to temporal structures [65, 125] - "*Embedded relationships among the attributes of events over different timescales carry predictions that guide proactive sensory and motor preparation in the brain*". Only providing item sequences may make it difficult for LLM to identify and utilize temporal patterns inside the sequence. Thus, we are motivated to explicitly provide temporal structures to LLM. Specifically, we conduct cluster analysis on the item sequence according to two criteria: items that are (1) *temporally proximate* and (2) *share similar features* should be clustered. In practice, we also use LLMs to complete the cluster tasks and find it can provide reasonable cluster results. The results are used as additional input to the LLM for ranking.

Structure analysis prompt

I have watched these movies in order: [item 1, item 2, ... item n]. Analyze the clusters within the history. Two criteria: 1) Similar items should be clustered together; 2) Temporal close similar items should be clustered.

3.4.4 Prompt Ensemble

The most straightforward way to combine various prompting strategies is to concatenate them and use the resulted long prompt. However, this approach risks exceeding the context length limitations of LLMs. Moreover, it has been observed that LLMs may lose important information within overly lengthy prompts [117]. To effectively utilize different prompting strategies, we propose ensembling the respective ranking outcomes derived from each. In this approach, we create several LLM sessions and obtain ranking lists with different prompts. Following Hou et al. [83], we explicitly define the output format for the ranking results produced by LLMs, and subsequently extract the ranking list using a post-hoc text parser. These ranking lists are aggregated to obtain the final ranking, as the process shown in Figure 3.12. Existing research also highlights the benefits of collaboration among multiple LLMs [189]. Specifically, we assign scores to each rank in the ranking list. For instance, in a ranking list of 20 items, the item in the 1st place receives 20 points, the 2nd place item gets 19 points, and so on, decreasing by one point per rank. Finally, we sum the scores for each item across all rankings.

3.4.5 Experiments

In this section, to fully demonstrate the effectiveness of `Tempura` in improving temporal awareness of LLMs, we conduct a set of extensive experiments to study the following research questions: (1) Can `Tempura` improve LLM's performance on sequential recommendation compared to other methods? (2) Can `Tempura` enhance the sensitivity of LLMs to temporal information in the input data? (3) How do factors like history length, the number of in-context examples or the choice of backbone LLMs influence the effectiveness of `Tempura`?

Table 3.8: Performance comparison on ML-1M and Amazon Review datasets. We highlight the best performance in **bold**. N@K denotes NDCG@K.

Method	ML-1M			Games			Kindle		
	N@1	N@5	N@10	N@1	N@5	N@10	N@1	N@5	N@10
BM25	4.00	13.14	20.53	16.50	30.09	37.19	6.50	18.07	24.96
UniSRec	9.00	20.08	26.72	19.50	34.86	40.82	5.00	16.21	25.03
VQ-Rec	9.50	19.52	27.11	5.50	16.76	25.27	4.30	14.22	23.58
Sequential	21.43	42.57	48.59	24.12	47.26	53.03	10.20	27.96	33.72
RF	26.56	45.99	51.27	25.63	50.02	53.72	11.11	28.77	35.71
ICL	26.40	47.51	53.32	26.00	49.68	53.63	13.07	30.82	36.41
Cluster	27.00	45.82	52.04	26.15	47.41	52.39	13.20	25.77	34.07
PCL	29.16	48.44	54.21	29.00	51.56	55.11	11.55	29.45	36.46
GCL	30.50	48.53	53.26	32.00	51.61	56.63	10.00	31.45	36.67
PCL + Cluster	30.50	48.35	**54.88**	35.50	53.89	58.74	12.00	30.15	**38.23**
`Tempura`	**31.50**	**48.64**	54.49	**39.00**	**56.51**	**60.95**	**14.00**	**32.17**	37.59

Main result

Datasets. The experiments are conducted on three widely-used public sequential recommendation datasets: (1) the movie rating dataset MovieLens-1M [71] (**ML-1M**) where user rated movies are regarded as interactions, (2) one category from the Amazon Review dataset [123] named **Games** where reviews are regarded as interactions, and (3) another category from Amazon Review dataset named **Kindle**. We sort the interactions of each user by timestamp, with the oldest interactions first, to construct the corresponding interaction sequences. The movie or product titles are used as the descriptive text of an item.

Evaluation configurations. Following existing works [83, 92, 158], we apply the leave-one-out strategy for evaluation. For each interaction sequence, the last item is used as the ground-truth item. We adopt the widely used metric NDCG@N to evaluate the ranking performance over the given candidate set $\mathcal{C}$ where $N \leq |\mathcal{C}|$. In the remainder of this paper, unless otherwise specified, $|\mathcal{C}|$ is set to 20. The candidate set consists of one ground-truth item and 19 randomly sampled negative items.

Baselines. We consider three prompt-based baselines discussed in [83]: **Sequential prompting**: Arrange the historical interactions in chronological order. **Recency-focused prompting (RF)**: In addition to the sequential interaction records, a sentence is additionally added to emphasize the most recent interaction. **In-context learning (ICL)**: Similar to PCL, but only use the most recent historic item as the in-context example. We also consider three methods designed for domain generalization: **BM25** [136] ranks items according to the

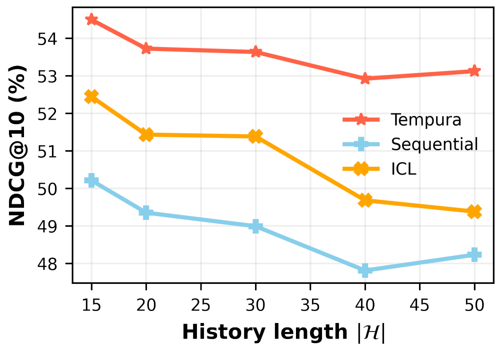


Figure 3.13: Performance vs. history length $|\mathcal{H}|$ (ML-1M).

textual similarity between candidates and historic items. **UniSRec** [82] equips textual item representations with an MoE-enhanced adaptor for domain fusion and adaptation. **VQ-Rec** [81] learns vector-quantized item representations, which can map item text into a vector of discrete indices (i.e., item codes) and use them to retrieve item representations from a code embedding table in recommendations. Additionally, we report the results with each single prompting strategy, as well as the results from ensembling PCL and cluster analysis.

Training-based methods such as Kang and McAuley [92] and Sun et al. [158] are not considered as baselines because: (1) They are designed based on item IDs, which can not be generalized to new domains with new ID spaces. (2) Our research focuses on improving the temporal awareness of LLMs, as evidenced by improved performance in sequential recommendations. Thus, our goal is not necessarily to develop a state-of-the-art sequential recommendation method.

Implementation details. Considering economic and efficiency factors, we follow [83, 192] to randomly sample 200 users along with their historical interactions for each dataset. Unless specified, we use the Azure OpenAI API `gpt-3.5-turbo`[8]. We set history length $|\mathcal{H}|$ as 15 and use the most recent 5 interactions as demonstrations in PCL. We found the length of the history significantly affects performance; therefore, we also searched for the optimal $|\mathcal{H}|$ for baselines. Empirically, $|\mathcal{H}| = 10$ yielded the best results for baselines in general. All the reported results are the average of three repeat runs to reduce the effect of randomness.

Results & analysis. We present the results on three datasets in Table 3.8. We can observe our prompting strategies in the third group improves upon existing baselines across all metrics. It is interesting to observe that PCL outperforms ICL significantly, where more demonstrations are used in PCL but ICL only use the last interaction as demonstration. This

[8]`https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/`

observation align with our analysis that more demonstrations are needed to learn to utilize temporal information in historical interaction sequences. Although the Cluster strategy exhibits limited performance on its own, it can significantly enhance performance when combined with other strategies in an ensemble. Additionally, we provide a case study of cluster analysis results in Section 3.4.5. By comparing individual prompting strategies with two ensemble-based methods, we find that ensembling consistently enhances performance by leveraging the strengths of different strategies. This suggests that different strategies emphasize various aspects, resulting in complementary results.

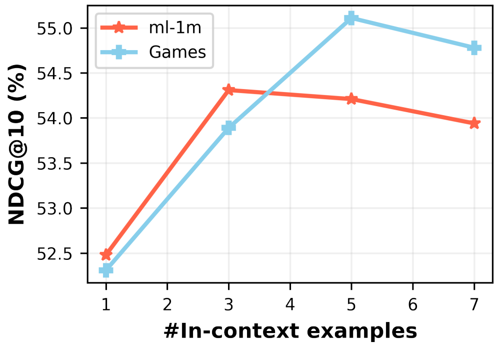


Figure 3.14: Impact of #in-context examples in PCL. Several more examples can improve performance.

Table 3.9: Case study of structure analysis in the historical interaction sequence.

Cluster 1: [Mad Max - PlayStation 4, Metal Gear Solid V: The Phantom Pain - PlayStation 4].
Cluster summary: Action games on PlayStation 4.
Cluster 2: [Star Wars: Battlefront - Standard Edition - PlayStation 4, Fallout 4 - PlayStation 4, Just Cause 3 - PlayStation 4, Far Cry Primal - PlayStation 4 Standard Edition].
Cluster summary: Open-world action games on PlayStation 4.
Cluster 3: [Tom Clancyś The Division - PlayStation 4, Uncharted 4: A Thiefś End - PlayStation 4, Homefront: The Revolution - PlayStation 4, Deus Ex: Mankind Divided - PlayStation 4].
Cluster summary: Action games with a focus on story and/or multiplayer on PlayStation 4.
Cluster 4: [Rise of the Tomb Raider: 20 Year Celebration - PlayStation 4, Dishonored 2 - PlayStation 4, Resident Evil 7: Biohazard - PS4 Digital Code, Horizon Zero Dawn - PlayStation 4, Tom Clancy's Ghost Recon Wildlands - PlayStation 4].
Cluster summary: Single-player action shooting games with a focus on exploration and/or stealth on PS4.
Target item: Prey - Pre-load - PS4 Digital Code First-person action-adventure shooting game

Ablation Study

Impact of history length. It has been reported in Hou et al. [83] that increasing the number of historical user behaviors does not improve the ranking performance, but even negatively impacts the ranking performance. To study the impact of history length on `Tempura`, we vary the history length $|\mathcal{H}|$ used for constructing the prompt from 15 to

50. We compare `Tempura` with the standard baseline Sequential and the best performing baseline ICL. Here history length $|\mathcal{H}|$ is the maximum allowed history length, the real history length could be shorter. We did not include the results on Games and Kindle since the user interaction history on these two datasets is short.

The results are reported in Figure 3.13. We observe that utilizing a longer history does not improve performance; in fact, it results in decreased performance on the ML-1M dataset. We hypothesize that the extensive history distracts LLMs, making it difficult for baselines to understand the evolution of user interests. By using temporal-aware prompts and the prompt ensemble strategy, `Tempura` demonstrates robust performance even with long historical interaction sequences.

Impact of the number of in-context examples. We utilize a user's historic items as in-context demonstrations to understand the temporal information in his / her behavior sequence. It is important to understand how many examples are needed. To this end, we study the performance with different number of examples in PCL. We keep the total length of the user's history as 15 and use the latest k items as examples, setting k to values in the set $[1, 3, 5, 7]$. We report the results on the ML-1M and Games datasets in Figure 3.14. We can observe more examples can boost the performance significantly than only one demonstration. As we analyzed in Section 3.4.3, LLMs learn to utilize temporal information by learning to predict a series of historical items. However, it is not always the case that more is better. It is observed that a slight performance drop with more examples. We speculate that longer prompts may cause distraction for LLMs.

Results on GPT-4. More advanced LLMs, like GPT-4 [1], demonstrate enhanced capabilities in knowledge, understanding, and reasoning. Therefore, we evaluate the sequential recommendation performance using GPT-4 to determine if `Tempura` can also augment GPT-4's capabilities. We present the results in Table 3.10. It has been observed that GPT-4 exhibits a robust capacity for sequential recommendation, even when employing the most standard prompting strategy, Sequential. Notably, the improvement is most significant on the Kindle dataset, leading to the hypothesis that GPT-4 possesses extensive knowledge about Kindle books. The performance improvement with GPT-4 shows its strong ability in understanding and utilizing temporal information. By applying `Tempura`, the performance can be further improved when the backbone LLM is more powerful.

Case Study

We present an example result from the cluster analysis conducted on the Games dataset. We employ `gpt-3.5-turbo` to cluster historic items using the prompt discussed in Section

Table 3.10: Performance with GPT-4 (NDCG@10). `Tempura` can further improve the performance when the backbone LLM is more powerful.

Method	ML-1M	Games	Kindle
Sequential	55.75	66.43	57.65
ICL	54.82	67.84	54.72
`Tempura`	58.39	68.13	58.59

3.4.3. The historic items was successfully clustered into 4 clusters, accompanied by a generated summary for each cluster. It can be easily observed that the user's most recent interest lies in action shooting games. With this analysis, the target item can be easily identified since it is a first-person action-adventure shooting game, aligning with the user's latest interest.

3.5 Conclusion

In this chapter, we investigate learning from interactive human feedback approaches. Firstly, in [38], we study an important but largely under-explored problem in conversational recommendation systems (CRS), i.e., reward function design. We present a principled solution for reward learning for CRS and formulate an online algorithm to learn intrinsic rewards via bi-level multi-objective optimization. The results on three CRS benchmarks demonstrated the effectiveness of learned intrinsic rewards.

Secondly, in [32, 37], we present meta-reinforcement-learning based solutions to handle cold-start policy learning for new users. In general, we learn a meta policy for generalization and fast adapt it on new users. Specifically, we have developed two strategies aimed at accelerating the exploration of user preferences, leveraging pre-trained user embeddings and clustering techniques to categorize user preferences.

Thirdly, we leverage the world knowledge and natural language understanding abilities of LLMs to interpret user preferences embedded in their sequential feedback. We introduce two kinds of prompting strategies: one to learn to track user preference changes via in-context learning and another to explicitly analyze the temporal structures in historical interaction sequences. Our study demonstrates that by incorporating specific prompting strategies, LLMs can significantly improve in capturing user preferences in the sequential human feedback.

Overall, we develop approaches to promptly and precisely comprehend and respond to human feedback from the perspectives of reward assignment, efficient personalized policy learning and feedback understanding. In conclusion, the advancements presented in this chapter contribute significantly toward the overarching aim of developing systems that engage users more effectively. By enhancing our understanding of user feedback and fostering more engaging conversations, these systems are better equipped to retain users and meet their needs.

Chapter 4

Conclusion and Future Work

In this dissertation, we explore the problem of Human-Feedback-driven Learning from both human and system perspectives. This research aims to deliver an in-depth examination of the characteristics of real-world human feedback within the context of machine learning, along with advanced methodologies to address the identified challenges.

4.1 Conclusion

In Chapter 2, we investigate advanced approaches for learning from noisy human feedback. To model the noise in human feedback, we propose a novel noise model distinguishes between common noise shared across annotators and the individual confusions specific to each annotator (Section 2.1 [34]). To address the challenge of sparse human feedback, we employ data augmentation through generative models to enrich the dataset with missing annotations (Section 2.2 [35]). Human feedback is essential to align Large Language Models (LLMs) with human values and needs. To mitigate the impact of noisy human feedback, we further design a fine-tuning framework to improve low-fidelity LLMs with the guidance from high-fidelity LLMs (Section 2.3).

In Chapter 3, we study the problem of learning from interactive human feedback. Our goal is to promptly and precisely comprehend and respond to human feedback. First, we introduce a multi-objective optimization framework designed to learn the assignment of credits to conversation policies, thereby enhancing the elicitation of human feedback (Section 3.1 [38]). Second, we address the problem of conversational policy learning for cold-start users via meta-reinforcement-learning frameworks (Section 3.2 [37] and 3.3 [32]). Third, we design a principled prompting framework Tempura to interpret complex user interests and intentions by leveraging the zero-shot reasoning capabilities of LLMs (Section 3.4).

Overall, this dissertation solves several key challenges in learning from real-world human feedback problems. The key contribution of the research lies in offering practical solutions that are specifically tailored to the unique characteristics of real-world human feedback scenarios, including learning from noisy human feedback and interactive human feedback. This is substantiated by rigorous theoretical analysis and extensive empirical evaluations conducted on public benchmarks, demonstrating the applicability and robustness of our research. This line of research will promote more robust and practical learning from human feedback solutions. Our research is dedicated to attaining precise alignment with human needs and values, with the expectation that such alignment will not only foster more human-centric systems but also enhance their utility for both humans and systems alike. More generally, our research has empowered a wide spectrum of important real-world applications such as recommender systems [35], crowdsourcing platforms [63, 169], online education [68], healthcare [157], ethical AI [200] and many more.

4.2 Future Work

Data privacy and security. It is important to consider potential privacy breaches in HFL-based systems, particularly those that collect users' personal information, such as medical records in smart healthcare systems. Differential privacy [52] is a promising technique to protect users' data privacy. For example, introducing randomness into human feedback to ensure that individual entries cannot be identified. Moreover, federated learning algorithms [203] can be utilized as they enable learning from human feedback without the need to centralize sensitive information, thus reducing privacy risks. Furthermore, secure multi-party computation (SMC) [50] could be useful in collaborative HFL systems where data come from a pool of users. By employing SMC, HFL systems can aggregate insights from distributed feedback without exposing the underlying data, further mitigating the risk of privacy breaches.

Interactive and multimodal feedback systems. Leveraging multiple modes of feedback (e.g., voice, video, touch signals) can provide richer information for machine learning models. For example, voice intonations can convey emotions more effectively than text, and video can provide contextual cues that are absent in other forms of feedback. Future research could focus on developing models that can integrate and interpret these diverse data types to gain a more nuanced understanding of user feedback. For example, multimodal large language models [199] are highly effective in integrating feedback from both images and text. Also, HFL is highly related to the Human-Computer Interaction (HCL) community, which employs design principles and usability studies to craft user interfaces capable of effectively eliciting and managing multimodal feedback. HCI techniques can be leveraged to develop HFL systems that are not only intuitive but also engaging, ensuring a seamless and user-friendly experience.

Bibliography

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab. "Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1313–1321.

[3] A. Antoniou, A. Storkey, and H. Edwards. "Data augmentation generative adversarial networks". In: *arXiv preprint arXiv:1711.04340* (2017).

[4] P. Auer. "Using confidence bounds for exploitation-exploration trade-offs". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.

[5] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. "Program synthesis with large language models". In: *arXiv preprint arXiv:2108.07732* (2021).

[6] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. "Stochastic Variational Video Prediction". In: *International Conference on Learning Representations*. 2018. URL: `https://openreview.net/forum?id=rk49Mg-CW`.

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

[8] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback". In: *arXiv preprint arXiv:2204.05862* (2022).

[9] K. Bao, J. Zhang, Y. Zhang, W. Wang, F. Feng, and X. He. "Tallrec: An effective and efficient tuning framework to align large language model with recommendation". In: *arXiv preprint arXiv:2305.00447* (2023).

[10] R. v. d. Berg, T. N. Kipf, and M. Welling. "Graph convolutional matrix completion". In: *arXiv preprint arXiv:1706.02263* (2017).

[11] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[12] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (2013).

[14] R. A. Bradley and M. E. Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[15] T. Buecheler, J. H. Sieg, R. M. Füchslin, and R. Pfeifer. "Crowdsourcing, open innovation and collective intelligence in the scientific method: a research agenda and operational framework". In: *The 12th International Conference on the Synthesis and Simulation of Living Systems, Odense, Denmark, 19-23 August 2010*. MIT Press. 2010, pp. 679–686.

[16] C. Burns, P. Izmailov, J. H. Kirchner, B. Baker, L. Gao, L. Aschenbrenner, Y. Chen, A. Ecoffet, M. Joglekar, J. Leike, et al. "Weak-to-strong generalization: Eliciting strong capabilities with weak supervision". In: *arXiv preprint arXiv:2312.09390* (2023).

[17] R. Cai, X. Bai, Z. Wang, Y. Shi, P. Sondhi, and H. Wang. "Modeling sequential online interactive behaviors with temporal point process". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 873–882.

[18] R. Cai, Q. Wang, C. Wang, and X. Liu. "Learning to structure long-term dependence for sequential recommendation". In: *arXiv preprint arXiv:2001.11369* (2020).

[19] R. Cai, J. Wu, A. San, C. Wang, and H. Wang. "Category-aware collaborative sequential recommendation". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 388–397.

[20] P. Cao, Y. Xu, Y. Kong, and Y. Wang. "Max-mig: an information theoretic approach for joint learning from crowds". In: *arXiv preprint arXiv:1905.13436* (2019).

[21] P. Cao, Y. Xu, Y. Kong, and Y. Wang. "Max-MIG: an Information Theoretic Approach for Joint Learning from Crowds". In: *ICLR*. 2019. URL: `https://openreview.net/forum?id=BJg9DoR9t7`.

[22] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J.-T. Lee. "Cfgan: A generic collaborative filtering framework based on generative adversarial networks". In: *Proceedings of the 27th ACM CIKM*. 2018, pp. 137–146.

[23] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, and Y. Li. "Sequential recommendation with graph neural networks". In: *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2021, pp. 378–387.

[24] D. Chen, L. Wu, S. Tang, X. Yun, B. Long, and Y. Zhuang. "Robust Meta-learning with Sampling Noise and Label Noise via Eigen-Reptile". In: *Proceedings of the 39th International Conference on Machine Learning* (2022).

[25] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. "Evaluating large language models trained on code". In: *arXiv preprint arXiv:2107.03374* (2021).

[26] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, and J. Tang. "Towards knowledge-based recommender dialog system". In: *arXiv preprint arXiv:1908.05391* (2019).

[27] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *arXiv preprint arXiv:1606.03657* (2016).

[28] Z. Chen. "PALR: Personalization Aware LLMs for Recommendation". In: *arXiv preprint arXiv:2305.07622* (2023).

[29] K. Christakopoulou, A. Beutel, R. Li, S. Jain, and E. H. Chi. "Q&R: A two-stage approach toward interactive recommendation". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 139–148.

[30] K. Christakopoulou, F. Radlinski, and K. Hofmann. "Towards conversational recommender systems". In: *Proceedings of the 22nd ACM SIGKDD*. 2016, pp. 815–824.

[31] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[32] Z. Chu, R. Cai, and H. Wang. "Meta-reinforcement learning via exploratory task clustering". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 10. 2024, pp. 11633–11641.

[33] Z. Chu, J. Ma, and H. Wang. "Learning from Crowds by Modeling Common Confusions". In: *arXiv preprint arXiv:2012.13052* (2020).

[34] Z. Chu, J. Ma, and H. Wang. "Learning from Crowds by Modeling Common Confusions." In: *AAAI*. 2021, pp. 5832–5840.

[35] Z. Chu and H. Wang. "Improve learning from crowds via generative augmentation". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 167–175.

[36] Z. Chu and H. Wang. "Meta-Reinforcement Learning via Exploratory Task Clustering". In: *arXiv preprint arXiv:2302.07958* (2023).

[37] Z. Chu, H. Wang, Y. Xiao, B. Long, and L. Wu. "Meta Policy Learning for Cold-Start Conversational Recommendation". In: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 2023, pp. 222–230.

[38] Z. Chu, N. Wang, and H. Wang. "Multi-Objective Intrinsic Reward Learning for Conversational Recommender Systems". In: *arXiv preprint arXiv:2310.20109* (2023).

[39] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[40] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. “Training verifiers to solve math word problems”. In: *arXiv preprint arXiv:2110.14168* (2021).

[41] P. Covington, J. Adams, and E. Sargin. “Deep neural networks for youtube recommendations”. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 191–198.

[42] D. Dai, Y. Sun, L. Dong, Y. Hao, Z. Sui, and F. Wei. “Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers”. In: *arXiv preprint arXiv:2212.10559* (2022).

[43] A. P. Dawid and A. M. Skene. “Maximum likelihood estimation of observer error-rates using the EM algorithm”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28.1 (1979), pp. 20–28.

[44] K. Deb and K. Deb. “Multi-objective optimization”. In: *Search methodologies: Introductory tutorials in optimization and decision support techniques*. Springer, 2013, pp. 403–449.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[46] J. Deng, J. Krause, and L. Fei-Fei. “Fine-grained crowdsourcing for fine-grained recognition”. In: *CVPR*. 2013, pp. 580–587.

[47] Y. Deng, Y. Li, F. Sun, B. Ding, and W. Lam. “Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning”. In: *arXiv preprint arXiv:2105.09710* (2021).

[48] J.-A. Désidéri. “Multiple-gradient descent algorithm (MGDA) for multiobjective optimization”. In: *Comptes Rendus Mathematique* 350.5-6 (2012), pp. 313–318.

[49] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. “Deep unsupervised clustering with gaussian mixture variational autoencoders”. In: *arXiv preprint arXiv:1611.02648* (2016).

[50] W. Du and M. J. Atallah. “Secure multi-party computation problems and their applications: a review and open problems”. In: *Proceedings of the 2001 workshop on New security paradigms*. 2001, pp. 13–22.

[51] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. “Rl2: Fast reinforcement learning via slow reinforcement learning”. In: *arXiv preprint arXiv:1611.02779* (2016).

[52] C. Dwork. “Differential privacy: A survey of results”. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[53] W. Fan, Z. Zhao, J. Li, Y. Liu, X. Mei, Y. Wang, J. Tang, and Q. Li. “Recommender systems in the era of large language models (llms)”. In: *arXiv preprint arXiv:2307.02046* (2023).

[54] C. Finn, P. Abbeel, and S. Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *ICML*. PMLR. 2017, pp. 1126–1135.

[55] G. Fischer. “User modeling in human–computer interaction”. In: *User modeling and user-adapted interaction* 11 (2001), pp. 65–86.

[56] G. French, M. Mackiewicz, and M. Fisher. “Self-ensembling for visual domain adaptation”. In: *arXiv preprint arXiv:1706.05208* (2017).

[57] A. Garivier, T. Lattimore, and E. Kaufmann. “On explore-then-commit strategies”. In: *Advances in Neural Information Processing Systems* 29 (2016).

[58] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. “Offline a/b testing for recommender systems”. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pp. 198–206.

[59] A. Glaese, N. McAleese, M. Trebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, et al. “Improving alignment of dialogue agents via targeted human judgements”. In: *arXiv preprint arXiv:2209.14375* (2022).

[60] J. Goldberger and E. Ben-Reuven. “Training deep neural-networks using a noise adaptation layer”. In: (2016).

[61] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[62] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial networks”. In: *arXiv preprint arXiv:1406.2661* (2014).

[63] M. L. Gordon, K. Zhou, K. Patel, T. Hashimoto, and M. S. Bernstein. “The disagreement deconvolution: Bringing machine learning performance metrics in line with reality”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–14.

[64] J. Gou, B. Yu, S. J. Maybank, and D. Tao. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[65] T. D. Griffiths, C. Büchel, R. S. Frackowiak, and R. D. Patterson. “Analysis of temporal structure in sound by the human brain”. In: *Nature neuroscience* 1.5 (1998), pp. 422–427.

[66] M. Y. Guan, V. Gulshan, A. M. Dai, and G. E. Hinton. “Who said what: Modeling individual labelers improves classification”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[67] J. Guiver and E. Snelson. “Bayesian inference for Plackett-Luce ranking models”. In: *proceedings of the 26th annual international conference on machine learning*. 2009, pp. 377–384.

[68] L. Guo, Y. Jin, G. Liu, F. Hao, M. Ren, and V. Loia. “Type diversity maximization aware coursewares crowdcollection with limited budget in MOOCs”. In: *Information Sciences* 649 (2023), p. 119663.

[69] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. “Meta-reinforcement learning of structured exploration strategies”. In: *Advances in neural information processing systems* 31 (2018).

[70] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[71] F. M. Harper and J. A. Konstan. “The movielens datasets: History and context”. In: *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[72] J. Harte, W. Zorgdrager, P. Louridas, A. Katsifodimos, D. Jannach, and M. Fragkoulis. “Leveraging large language models for sequential recommendation”. In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1096–1102.

[73] H. Harutyunyan, K. Reing, G. Ver Steeg, and A. Galstyan. “Improving generalization by controlling label-noise information in neural network weights”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4071–4081.

[74] R. He and J. McAuley. “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering”. In: *proceedings of the 25th international conference on world wide web*. 2016, pp. 507–517.

[75] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. “Neural collaborative filtering”. In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[76] Z. He, Z. Xie, R. Jha, H. Steck, D. Liang, Y. Feng, B. P. Majumder, N. Kallus, and J. McAuley. “Large Language Models as Zero-Shot Conversational Recommenders”. In: *arXiv preprint arXiv:2308.10053* (2023).

[77] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. “Session-based recommendations with recurrent neural networks”. In: *arXiv preprint arXiv:1511.06939* (2015).

[78] G. Hinton, O. Vinyals, and J. Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[79] J. Hong, N. Lee, and J. Thorne. “Reference-free Monolithic Preference Optimization with Odds Ratio”. In: *arXiv preprint arXiv:2403.07691* (2024).

[80] J. J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[81] Y. Hou, Z. He, J. McAuley, and W. X. Zhao. “Learning vector-quantized item representation for transferable sequential recommenders”. In: *Proceedings of the ACM Web Conference 2023*. 2023, pp. 1162–1171.

[82] Y. Hou, S. Mu, W. X. Zhao, Y. Li, B. Ding, and J.-R. Wen. “Towards universal sequence representation learning for recommender systems”. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 585–593.

[83] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao. “Large language models are zero-shot rankers for recommender systems”. In: *arXiv preprint arXiv:2305.08845* (2023).

[84] M. Huai, J. Sun, R. Cai, L. Yao, and A. Zhang. “Malicious attacks against deep reinforcement learning interpretations”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 472–482.

[85] J. Humplik, A. Galashov, L. Hasenclever, P. A. Ortega, Y. W. Teh, and N. Heess. “Meta reinforcement learning as task inference”. In: *arXiv preprint arXiv:1905.06424* (2019).

[86] H. Imamura, I. Sato, and M. Sugiyama. “Analysis of minimax error rate for crowdsourcing and its application to worker clustering model”. In: *arXiv preprint arXiv:1802.04551* (2018).

[87] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[88] A. A. Irissappane, H. Yu, Y. Shen, A. Agrawal, and G. Stanton. “Leveraging GPT-2 for Classifying Spam Reviews with Limited Labeled Data via Adversarial Training”. In: *arXiv preprint arXiv:2012.13400* (2020).

[89] T. Joachims, A. Swaminathan, and M. de Rijke. “Deep learning with logged bandit feedback”. In: *ICLR*. 2018.

[90] E. Kamar, A. Kapoor, and E. Horvitz. “Identifying and accounting for task-dependent bias in crowdsourcing”. In: *Third AAAI Conference on Human Computation and Crowdsourcing*. Citeseer. 2015.

[91] P.-A. Kamienny, M. Pirotta, A. Lazaric, T. Lavril, N. Usunier, and L. Denoyer. “Learning adaptive exploration strategies in dynamic environments through informed policy regularization”. In: *arXiv preprint arXiv:2005.02934* (2020).

[92] W.-C. Kang and J. McAuley. “Self-attentive sequential recommendation”. In: *2018 IEEE international conference on data mining (ICDM)*. IEEE. 2018, pp. 197–206.

[93] W.-C. Kang, J. Ni, N. Mehta, M. Sathiamoorthy, L. Hong, E. Chi, and D. Z. Cheng. “Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction”. In: *arXiv preprint arXiv:2305.06474* (2023).

[94] A. Khetan and S. Oh. “Achieving budget-optimality with adaptive schemes in crowdsourcing”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4844–4852.

[95] M. Kim, H. Song, Y. Shin, D. Park, K. Shin, and J.-G. Lee. “Meta-Learning for Online Update of Recommender Systems”. In: (2022).

[96] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[97] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[98] T. N. Kipf and M. Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[99] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[100] S. Laine and T. Aila. “Temporal ensembling for semi-supervised learning”. In: *arXiv preprint arXiv:1610.02242* (2016).

[101] J. Lanchantin, A. Sekhon, and Y. Qi. “Neural message passing for multi-label classification”. In: *ECML-PKDD*. Springer. 2019, pp. 138–163.

[102] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[103] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung. “Melu: Meta-learned user preference estimator for cold-start recommendation”. In: *Proceedings of the 25th ACM SIGKDD*. 2019, pp. 1073–1082.

[104] Y. Lee and S. Choi. “Gradient-based meta-learning with learned layerwise metric and subspace”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2927–2936.

[105] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, and T.-S. Chua. “Estimation-action-reflection: Towards deep interaction between conversational and recommender systems”. In: *Proceedings of the 13th WSDM Conference*. 2020, pp. 304–312.

[106] W. Lei, G. Zhang, X. He, Y. Miao, X. Wang, L. Chen, and T.-S. Chua. “Interactive path reasoning on graph for conversational recommendation”. In: *Proceedings of the 26th ACM SIGKDD*. 2020, pp. 2073–2083.

[107] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. “Crowdsourced data management: A survey”. In: *IEEE TKDE* 28.9 (2016), pp. 2296–2319.

[108] J. Li, M. Wang, J. Li, J. Fu, X. Shen, J. Shang, and J. McAuley. “Text Is All You Need: Learning Language Representations for Sequential Recommendation”. In: *arXiv preprint arXiv:2305.13731* (2023).

[109] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. “Neural attentive session-based recommendation”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 1419–1428.

[110] L. Li, Y. Zhang, D. Liu, and L. Chen. “Large Language Models for Generative Recommendation: A Survey and Visionary Discussions”. In: *arXiv preprint arXiv:2309.01157* (2023).

[111] L. Li, J. Y. Kim, and I. Zitouni. “Toward predicting the outcome of an A/B experiment for search relevance”. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 2015, pp. 37–46.

[112] R. Li, S. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal. “Towards deep conversational recommendations”. In: *Proceedings of the 32nd NeurIPS Conference*. 2018, pp. 9748–9758.

[113] S. Li, W. Lei, Q. Wu, X. He, P. Jiang, and T.-S. Chua. “Seamlessly unifying attributes and items: Conversational recommendation for cold-start users”. In: *ACM TOIS* 39.4 (2021), pp. 1–29.

[114] Y. Li, B. Rubinstein, and T. Cohn. “Exploiting worker correlation for label aggregation in crowdsourcing”. In: *International Conference on Machine Learning*. 2019, pp. 3886–3895.

[115] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn. “Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices”. In: *ICML*. PMLR. 2021, pp. 6925–6935.

[116] J. Liu, C. Liu, R. Lv, K. Zhou, and Y. Zhang. “Is chatgpt a good recommender? a preliminary study”. In: *arXiv preprint arXiv:2304.10149* (2023).

[117] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. “Lost in the middle: How language models use long contexts”. In: *arXiv preprint arXiv:2307.03172* (2023).

[118] R. Liu, F. Bai, Y. Du, and Y. Yang. “Meta-Reward-Net: Implicitly Differentiable Reward Learning for Preference-based Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 22270–22284.

[119] B. D. Lund and T. Wang. “Chatting about ChatGPT: how may AI and GPT impact academia and libraries?” In: *Library Hi Tech News* 40.3 (2023), pp. 26–29.

[120] J. McAuley. *Personalized machine learning*. Cambridge University Press, 2022.

[121] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. “Rethinking the role of demonstrations: What makes in-context learning work?” In: *arXiv preprint arXiv:2202.12837* (2022).

[122] A. Y. Ng, D. Harada, and S. Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: *Icml*. Vol. 99. Citeseer. 1999, pp. 278–287.

[123] J. Ni, J. Li, and J. McAuley. “Justifying recommendations using distantly-labeled reviews and fine-grained aspects”. In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2019, pp. 188–197.

[124] A. Nichol, J. Achiam, and J. Schulman. “On first-order meta-learning algorithms”. In: *arXiv preprint arXiv:1803.02999* (2018).

[125] A. C. Nobre and F. Van Ede. “Anticipated moments: temporal structure in attention”. In: *Nature Reviews Neuroscience* 19.1 (2018), pp. 34–48.

[126] A. Odena. “Semi-supervised learning with generative adversarial networks”. In: *arXiv preprint arXiv:1606.01583* (2016).

[127] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[128] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. “Curiosity-driven exploration by self-supervised prediction”. In: *International conference on machine learning*. PMLR. 2017, pp. 2778–2787.

[129] J. C. Peterson, R. M. Battleday, T. L. Griffiths, and O. Russakovsky. “Human uncertainty makes classification more robust”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9617–9626.

[130] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. “Direct preference optimization: Your language model is secretly a reward model”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[131] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. “Efficient off-policy meta-reinforcement learning via probabilistic context variables”. In: *International conference on machine learning*. PMLR. 2019, pp. 5331–5340.

[132] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell. “Continual unsupervised representation learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[133] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. “Learning from crowds”. In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1297–1322.

[134] S. Rendle. “Factorization machines”. In: *2010 IEEE International conference on data mining*. IEEE. 2010, pp. 995–1000.

[135] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. “Factorizing personalized markov chains for next-basket recommendation”. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 811–820.

[136] S. Robertson, H. Zaragoza, et al. “The probabilistic relevance framework: BM25 and beyond”. In: *Foundations and Trends® in Information Retrieval* 3.4 (2009), pp. 333–389.

[137] F. Rodrigues, F. Pereira, and B. Ribeiro. “Gaussian process classification and active learning with multiple annotators”. In: *International conference on machine learning*. 2014, pp. 433–441.

[138] F. Rodrigues and F. C. Pereira. “Deep learning from crowds”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[139] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel. “Promp: Proximal meta-policy search”. In: *arXiv preprint arXiv:1810.06784* (2018).

[140] M. A. Runco. *Divergent thinking*. Ablex Publishing Corporation Norwood, NJ, 1991.

[141] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. “LabelMe: a database and web-based tool for image annotation”. In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.

[142] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[143] T. Schnabel, P. N. Bennett, S. T. Dumais, and T. Joachims. “Short-term satisfaction and long-term coverage: Understanding how users tolerate algorithmic exploration”. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pp. 513–521.

[144] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[145] O. Sener and V. Koltun. “Multi-task learning as multi-objective optimization”. In: *Advances in neural information processing systems* 31 (2018).

[146] B. Settles. “Active learning literature survey”. In: (2009).

[147] N. B. Shah, S. Balakrishnan, and M. J. Wainwright. “A permutation-based model for crowd labeling: Optimal estimation and robustness”. In: *arXiv preprint arXiv:1606.09632* (2016).

[148] V. S. Sheng and J. Zhang. “Machine learning with crowdsourcing: A brief summary of the past research and future directions”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 9837–9843.

[149] R. Shu, H. H. Bui, H. Narui, and S. Ermon. “A dirt-t approach to unsupervised domain adaptation”. In: *arXiv preprint arXiv:1802.08735* (2018).

[150] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. “Intrinsically motivated reinforcement learning: An evolutionary perspective”. In: *IEEE Transactions on Autonomous Mental Development* 2.2 (2010), pp. 70–82.

[151] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl, et al. “Large language models encode clinical knowledge”. In: *Nature* 620.7972 (2023), pp. 172–180.

[152] A. Singla, I. Bogunovic, G. Bartók, A. Karbasi, and A. Krause. “Near-optimally teaching the crowd to classify”. In: *ICML*. PMLR. 2014, pp. 154–162.

[153] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. “Learning from noisy labels with deep neural networks: A survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[154] J. T. Springenberg. “Unsupervised and semi-supervised learning with categorical generative adversarial networks”. In: *arXiv preprint arXiv:1511.06390* (2015).

[155] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever. “Some Considerations on Learning to Explore via Meta-Reinforcement Learning”. In: *ArXiv* abs/1803.01118 (2018).

[156] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. "Learning to summarize with human feedback". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3008–3021.

[157] D. Stutz, A. T. Cemgil, A. G. Roy, T. Matejovicova, M. Barsbey, P. Strachan, M. Schaekermann, J. Freyberg, R. Rikhye, B. Freeman, et al. "Evaluating AI systems under uncertain ground truth: a case study in dermatology". In: *arXiv preprint arXiv:2307.02191* (2023).

[158] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer". In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[159] Y. Sun and Y. Zhang. "Conversational recommender system". In: *The 41st ACM SIGIR*. 2018, pp. 235–244.

[160] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[161] A. Swaminathan and T. Joachims. "Batch learning from logged bandit feedback through counterfactual risk minimization". In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 1731–1755.

[162] J. Tang and K. Wang. "Personalized top-n sequential recommendation via convolutional sequence embedding". In: *Proceedings of the eleventh ACM international conference on web search and data mining*. 2018, pp. 565–573.

[163] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman. "Learning from noisy labels by regularized estimation of annotator confusion". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11244–11253.

[164] A. Tarvainen and H. Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems* 30 (2017).

[165] F. Tétard and M. Collan. "Lazy user theory: A dynamic model to understand user selection of products and services". In: *2009 42nd Hawaii International Conference on System Sciences*. IEEE. 2009, pp. 1–9.

[166] E. Todorov, T. Erez, and Y. Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.

[167] R. S. Tsay. *Analysis of financial time series*. John wiley & sons, 2005.

[168] L. Tunstall, E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, L. von Werra, C. Fourrier, N. Habib, et al. "Zephyr: Direct distillation of lm alignment". In: *arXiv preprint arXiv:2310.16944* (2023).

[169] D. Ustalov, N. Pavlichenko, and B. Tseitlin. "Learning from Crowds with Crowd-Kit". In: *arXiv preprint arXiv:2109.08584* (2021).

[170] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle. "A meta-learning perspective on cold-start recommendations for items". In: (2017).

[171] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[172] J. W. Vaughan. "Making better use of the crowd: How crowdsourcing can advance machine learning research". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 7026–7071.

[173] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. "Community-based bayesian aggregation models for crowdsourcing". In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 155–164.

[174] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim. "Multimodal model-agnostic meta-learning via task-aware modulation". In: *arXiv preprint arXiv:1910.13616* (2019).

[175] C. Wang, X. Liu, and D. Song. "Language models are open knowledge graphs". In: *arXiv preprint arXiv:2010.11967* (2020).

[176] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo. "Graphgan: Graph representation learning with generative adversarial nets". In: *AAAI*. Vol. 32. 1. 2018.

[177] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. "Learning to reinforcement learn". In: *arXiv preprint arXiv:1611.05763* (2016).

[178] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. "Irgan: A minimax game for unifying generative and discriminative information retrieval models". In: *Proceedings of the 40th International ACM SIGIR conference*. 2017, pp. 515–524.

[179] L. Wang and E.-P. Lim. "Zero-Shot Next-Item Recommendation using Large Pre-trained Language Models". In: *arXiv preprint arXiv:2304.03153* (2023).

[180] Q. Wang, H. Yin, H. Wang, Q. V. H. Nguyen, Z. Huang, and L. Cui. "Enhancing collaborative filtering with generative augmentation". In: *Proceedings of the 25th ACM SIGKDD conference*. 2019, pp. 548–556.

[181] Y. Wang, Z. Jiang, Z. Chen, F. Yang, Y. Zhou, E. Cho, X. Fan, X. Huang, Y. Lu, and Y. Yang. "RecMind: Large Language Model Powered Agent For Recommendation". In: *arXiv preprint arXiv:2308.14296* (2023).

[182] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. "Finetuned language models are zero-shot learners". In: *arXiv preprint arXiv:2109.01652* (2021).

[183] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[184] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. “The multidimensional wisdom of crowds”. In: *Advances in neural information processing systems*. 2010, pp. 2424–2432.

[185] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. “Whose vote should count more: Optimal integration of labels from labelers of unknown expertise”. In: *Advances in neural information processing systems*. 2009, pp. 2035–2043.

[186] R. J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3 (1992), pp. 229–256.

[187] J. Wu, R. Cai, and H. Wang. “Déjà vu: A contextualized temporal attention mechanism for sequential recommendation”. In: *Proceedings of The Web Conference 2020*. 2020, pp. 2199–2209.

[188] J. Wu, C. Zhao, T. Yu, J. Li, and S. Li. “Clustering of Conversational Bandits for User Preference Learning and Elicitation”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2129–2139.

[189] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang. “Autogen: Enabling next-gen llm applications via multi-agent conversation framework”. In: *arXiv preprint arXiv:2308.08155* (2023).

[190] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan. “Session-based recommendation with graph neural networks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 346–353.

[191] K. Xu, J. Yang, J. Xu, S. Gao, J. Guo, and J.-R. Wen. “Adapting User Preference to Online Feedback in Multi-round Conversational Recommendation”. In: *Proceedings of the 14th ACM WSDM*. 2021, pp. 364–372.

[192] L. Xu, J. Zhang, B. Li, J. Wang, M. Cai, W. X. Zhao, and J.-R. Wen. “Prompting Large Language Models for Recommender Systems: A Comprehensive Framework and Empirical Analysis”. In: *arXiv preprint arXiv:2401.04997* (2024).

[193] Y. Xu, P. Cao, Y. Kong, and Y. Wang. “L_DMI: A Novel Information-theoretic Loss Function for Training Deep Nets Robust to Label Noise.” In: *NeurIPS*. 2019, pp. 6222–6233.

[194] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy. “Learning from multiple annotators with varying expertise”. In: *Machine learning* 95.3 (2014), pp. 291–327.

[195] F. Yao, R. Cai, and H. Wang. “Reversible action design for combinatorial optimization with reinforcement learning”. In: *arXiv preprint arXiv:2102.07210* (2021).

[196] H. Yao, Y. Wei, J. Huang, and Z. Li. “Hierarchically structured meta-learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7045–7054.

[197] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. “Tree of thoughts: Deliberate problem solving with large language models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[198] L. Yin, J. Han, W. Zhang, and Y. Yu. “Aggregating crowd wisdoms with label-aware autoencoders”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 1325–1331.

[199] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen. “A survey on multimodal large language models”. In: *arXiv preprint arXiv:2306.13549* (2023).

[200] W. Yin, V. Agarwal, A. Jiang, A. Zubiaga, and N. Sastry. “Annobert: Effectively representing multiple annotators’ label choices to improve hate speech detection”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 17. 2023, pp. 902–913.

[201] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning”. In: *Conference on robot learning*. PMLR. 2020, pp. 1094–1100.

[202] M.-C. Yuen, I. King, and K.-S. Leung. “A survey of crowdsourcing systems”. In: *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*. IEEE. 2011, pp. 766–773.

[203] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao. “A survey on federated learning”. In: *Knowledge-Based Systems* 216 (2021), p. 106775.

[204] J. Zhang, J. Wang, H. Hu, T. Chen, Y. Chen, C. Fan, and C. Zhang. “Metacure: Meta reinforcement learning with empowerment-driven exploration”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12600–12610.

[205] T. Zhang, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, X. Zhou, et al. “Feature-level Deeper Self-Attention Network for Sequential Recommendation.” In: *IJCAI*. 2019, pp. 4320–4326.

[206] X. Zhang, H. Xie, H. Li, and J. CS Lui. “Conversational contextual bandit: Algorithm and application”. In: *Proceedings of The Web Conference 2020*. 2020, pp. 662–672.

[207] Y. Zhang, L. Wu, Q. Shen, Y. Pang, Z. Wei, F. Xu, B. Long, and J. Pei. “Multi-Choice Questions based Multi-Interest Policy Learning for Conversational Recommendation”. In: *arXiv preprint arXiv:2112.11775* (2021).

[208] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft. “Towards conversational search and recommendation: System ask, user respond”. In: *Proceedings of the 27th acm international conference on information and knowledge management*. 2018, pp. 177–186.

[209] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. “Spectral methods meet EM: A provably optimal algorithm for crowdsourcing”. In: *Advances in neural information processing systems*. 2014, pp. 1260–1268.

[210] C. Zhao, T. Yu, Z. Xie, and S. Li. “Knowledge-aware Conversational Preference Elicitation with Bandit Feedback”. In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 483–492.

[211] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin. “Recommendations with negative feedback via pairwise deep reinforcement learning”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 1040–1048.

[212] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh. “What can learned intrinsic rewards capture?” In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11436–11446.

[213] Z. Zheng, J. Oh, and S. Singh. “On learning intrinsic rewards for policy gradient methods”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[214] D. Zhou, S. Basu, Y. Mao, and J. C. Platt. “Learning from the wisdom of crowds by minimax entropy”. In: *Advances in neural information processing systems*. 2012, pp. 2195–2203.

[215] K. Zhou, Y. Zhou, W. X. Zhao, X. Wang, and J.-R. Wen. “Towards Topic-Guided Conversational Recommender System”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. 2020, pp. 4128–4139.

[216] B. Zhu, J. Jiao, and M. I. Jordan. “Principled Reinforcement Learning with Human Feedback from Pairwise or K-wise Comparisons”. In: *arXiv preprint arXiv:2301.11270* (2023).

[217] L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. “VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=Hkl9JlBYvr.

[218] L. Zou, L. Xia, Y. Gu, X. Zhao, W. Liu, J. X. Huang, and D. Yin. “Neural interactive collaborative filtering”. In: *Proceedings of the 43rd ACM SIGIR*. 2020, pp. 749–758.