Machine Learning-based Approaches to Data Quality Improvement in Mobile Crowdsensing and Crowdsourcing

by

Jinghan Jiang
B.Sc., Huazhong University of Science and Technology, 2016

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

© Jinghan Jiang, 2021
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

Machine Learning-based Approaches to Data Quality Improvement in Mobile Crowdsensing and Crowdsourcing

by

Jinghan Jiang
B.Sc., Huazhong University of Science and Technology, 2016

Supervisory Committee

Dr. Kui Wu, Supervisor
(Department of Computer Science)

Dr. Sean Chester, Department Member
(Department of Computer Science)

Dr. Issa Traore, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Kui Wu, Supervisor
(Department of Computer Science)

Dr. Sean Chester, Department Member
(Department of Computer Science)

Dr. Issa Traore, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

With the wide popularity of smart devices such as smartphones, smartwatches, and smart cameras, Mobile Crowdsensing (MCS) and Crowdsourcing (CS) have been broadly applied for collecting data from a large group of ordinary participants. The quality of participants' contributed data, however, is hard to guarantee, and as such it is critical to develop efficient and effective methods to automatically improve data quality over MCS/CS platforms. In this thesis, we propose three machine learning-based solutions for data quality enhancement in different participatory MCS/CS scenarios. Our solutions aim at the data extraction phase as well as the data collection phase of participatory MCS/CS, including: (1) trustworthy information extraction from conflicting data, (2) recognition of learning patterns, and (3) worker recruitment based on interactive training and learning pattern extraction. The first one is designed for the data extraction phase and the other two for the data collection phase.

First, to derive reliable data from diverse or even conflicting labels from the crowd, we design a mechanism to infuse knowledge from domain experts into the labels from the crowd to automatically make correct decisions on classification-based MCS tasks. Our solution, named EFusion, utilizes a probabilistic graphical model and the expectation maximization (EM) algorithm to infer the most likely expertise level of each crowd worker, the difficulty level of tasks, and the ground truth answers. Furthermore, we introduce a method to extend EFusion from solving binary classification problems to handling multi-class classification problems. We evaluate EFusion using real-world case studies as well as simulations. Evaluation results demonstrate that EFusion can return more accurate and stable classification results than the majority voting method and state-of-the-art methods.

Second, we propose Goldilocks, an interactive learning pattern recognition framework that can identify suitable participants whose performance follows desired learning patterns. To accurately extract a participant's learning pattern, we first estimate the impact of previous training questions on the participant before she answers a new question. After the participant answers each new question, we adjust the estimation of her capability by considering a quantitative measure of the impact of

previous questions and her answer to the new question. Based on the extracted learning curve of each participant, we recruit the candidates, who have showed good learning capability and desired learning patterns, for the formal MCS/CS task. We further develop a web service over Amazon Web Services (AWS) that automatically adjusts questions to maximize individual participants' learning performance. This website also profiles the participants' learning patterns, which can be used for task assignment in MCS/CS.

Third, we present HybrTraining, a hybrid deep learning framework that captures each candidate's capability from a long-term perspective and excludes the undesired candidates in the early stage of the training phase. Using two collaborative deep learning networks, HybrTraining can dynamically match participants and MCS/CS tasks. In detail, we build a deep Q-network (DQN) to match the candidates and training batches in the training phase, and develop a long short-term memory (LSTM) model that extracts the learning patterns of different candidates and helps the DQN make better worker-task matching decisions. We build HyberTraining on Compute Canada and evaluate it over two scientific datasets. For each dataset, the learning data of candidates is collected with a Python-based Django website over Amazon Elastic Compute Cloud (Amazon EC2). Evaluation results show that HybrTraining can increase data collection efficiency and improve data quality in MCS/CS.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor, Dr. Kui Wu, for continuously mentoring me, always being supportive and patient during my Ph.D. study.

my family and friends, for their consistent and unconditional support and care.

my collaborators, for the meaningful and fun time to work together.

my committee members, Dr. Sean Chester and Dr. Issa Traore, for the precious review and comments on my thesis.

Dr. Thomas Kunz, for helping me as the external examiner.

DEDICATION

To the lonely moments in these past five years, that helped me grow up.

Chapter 1

Introduction

With the popularity of smart devices such as smartphones, smartwatches, and smart cameras, it is more and more simple for people to employ their mobile devices to upload the local data to the Internet. Inspired by such popularity of Internet and smart devices, the concept of Mobile Crowdsensing (MCS) and Crowdsensing (CS) has been widely used to solve many real-world problems, because a task (e.g., labelling images) can be performed in parallel by a large crowd through with MCS/CS platforms (e.g. Amazon Mechanical Turk [92]). Although MCS/CS can work for data collection and analysis problems [7] in a flexible, economical and efficient way, critical and novel challenges are posed in terms of quality control since the crowd is typically composed of people with unknown and diverse capabilities, interests, and personal preference. Currently, MCS/CS platforms are leveraging advanced machine learning methods for quality control in many tedious and complicated real life problems.

1.1 Mobile Crowdsensing (MCS) and Crowdsourcing (CS)

The concept of CS is proposed by Jeff Howe and Mark Robinson in 2006 [37]. The core idea is to divide a complicated task into many "microtasks" and employ an open group of participants to work on them in parallel to achieve a cumulative result. Derived from the concept of CS, MCS specifically focuses on the type of crowdsourcing tasks where the sensor data comes from mobile devices. In other words, the ubiquitous mobile smart devices serve as the end sensors to collect and upload various data.

A typical MCS/CS system consists of three main components: requester, crowd workers, and MCS/CS platform. The typical workflow of MCS/CS system is as follow: The requester submits an MCS/CS job (Step 1) to the MCS/CS platform. The platform then employs a task dispatcher to assign the task to mobile users, called crowd workers or simply workers of MCS/CS (Step 2). After being assigned the task, the crowd workers finish the task and submit their feedback to the MCS/CS platform (Step 3). Last, the MCS/CS platform returns the collected data from crowd workers to the requester (Step 4). This workflow is shown in Fig. 1.1. Normally, different solutions are applied in Step 2 and Step 4 to improve the quality of MCS/CS data.

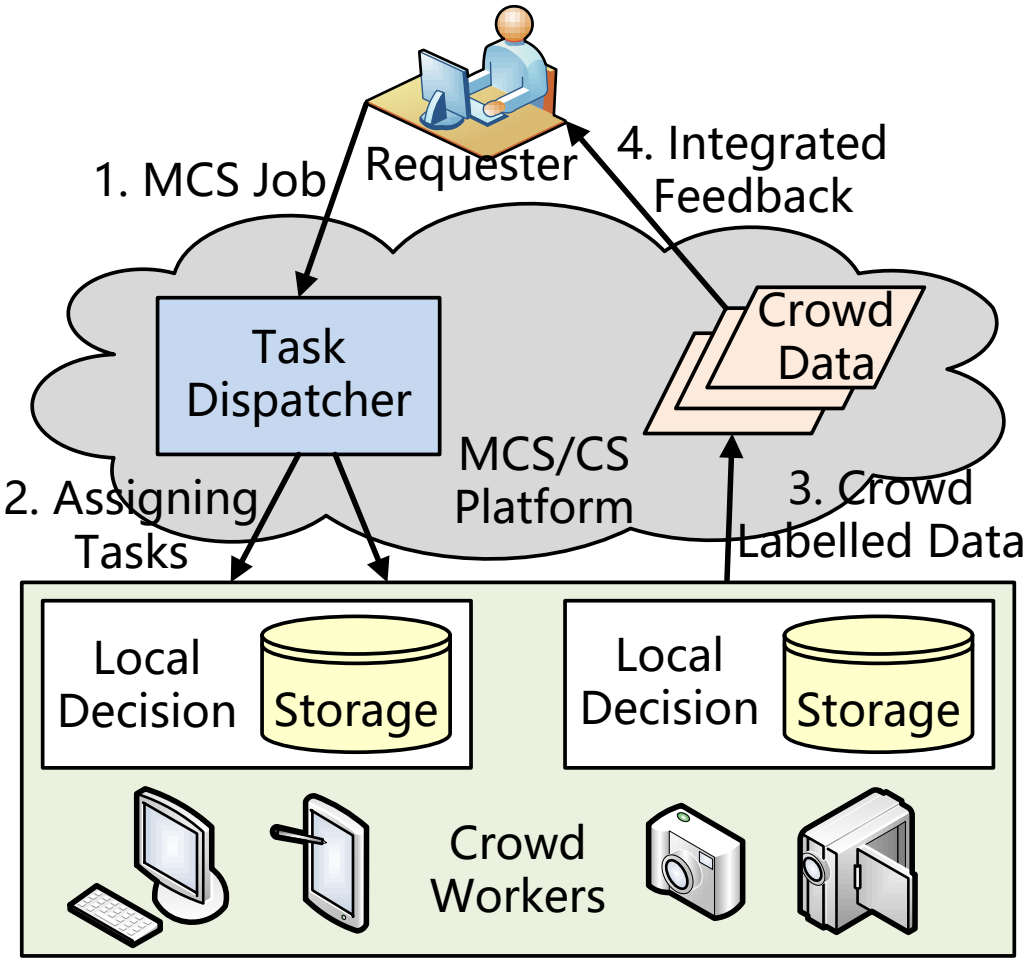


Figure 1.1: Typical work flow of MCS/CS.

1.2 Data Quality Optimization in MCS/CS

Reliable data is essential to ensure the effectiveness of MCS/CS because the final result returned by the MCS/CS platform is extracted based on the analysis of the collected data. However, the quality of data contributed by different workers is hard to guarantee in practice due to various reasons, e.g. the conflicting information from different workers, the diverse capabilities and preferences of crowd workers, and the insufficiency of training data.

Data quality optimization was traditionally investigated in the context of enterprise data management systems as well as the outcome of web searches [49]. Using mathematical models, e.g., maximum likelihood estimation (MLE) [71], Maximum A Posteriori (MAP) Estimation [68], and matrix factorization and regularization [60], this kind of study has been extended to more areas including information classification, information identification, and information extraction in MCS/CS.

Despite substantial development efforts to enhance the data quality in MCS/CS, we observe that the following problems have not been thoroughly addressed:

- For an MCS/CS task without ground truth, the data from domain expert is usually more reliable than that from the common workers. Nevertheless, it is too expensive to recruit many experts for labelling the large amount of data, and such a high cost may defeat the original purpose of MCS/CS: using cheaper crowd workers for tedious tasks.
- The training process of crowd workers is usually time consuming, and only worker's static capability (e.g., total number of correct answers during the training) instead of their learning pattern has been considered in the worker recruitment strategy.
- The MCS/CS platform usually lacks a mechanism to dynamically train crowd workers[1] and quickly match MCS/CS tasks to appropriate workers.

1.3 Research Objectives and Contributions

Motivated by the above observations, we focus on addressing three critical problems in this thesis, covering the research domains of conflicting data integration, worker learning pattern detection, and dynamic worker recruitment in MCS/CS. These three problems along with the corresponding MCS/CS stages are shown in Fig. 1.2.

Our research can benefit the task requester, MCS/CS platforms, as well as crowd workers. For the task requester, high quality data from the crowd can be collected through our solution of the first problem. By solving the other two problems before the task assignment step shown in Fig. 1.2, both efficiency of data collection and the quality of collected data can be improved. For MCS/CS platforms, the improvement of efficiency and effectiveness in task assignment and/or worker estimation can not only lower operating cost but also enhance service quality, where higher service quality is also essential to attract more users. In addition, the capability level and/or learning pattern of crowd workers identified by our solutions can be used by MCS/CS platforms for better user management. For crowd workers, they can improve their profile data for the future task selection. One more

[1] Dynamical training means adjusting training questions based on the trainee's performance over time.

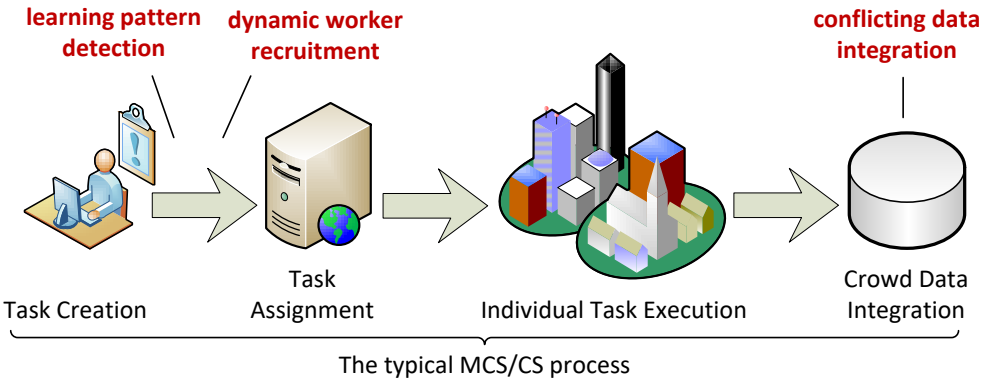


Figure 1.2: The three research problems with corresponding MCS/CS stages.

benefit for crowd workers is that they can get personalized training during an interactive training process so it is easier for them to master new skill/knowledge.

1.3.1 Automatic Expert Knowledge Infusion

In a typical MCS/CS workflow, the quality of answers from workers depends on many aspects, e.g., the domain knowledge of different workers, and the computational power and storage of smart devices. Therefore, MCS/CS needs a "quality assurance" mechanism to verify the results from crowd workers. Clearly, depending on the domain experts to monitor and check all results from crowd workers is prohibitive and defeats the original purpose of using cheap crowd workers. Therefore, a question that needs to be answered is: *how can we infuse the knowledge learned with a small amount of reliable answers from the domain experts to automatically correct crowd workers' labels?*

To solve this question, we propose a new algorithm, called EFusion, which infuses the knowledge from experts into the unreliable data from the crowd to automatically identify and eliminate incorrectly labelled data. Since it is unrealistic to ask experts to answer all questions due to the higher cost (e.g., higher hourly rate) in hiring professionals, EFusion takes advantage of both experts and crowd workers. In this regard, we have made the following contributions:

- We design a new algorithm, EFusion, which utilizes the knowledge from domain experts to gauge answers from unreliable distributed smart devices and workers. By taking advantage of both types of contributors, EFusion greatly improves the likelihood in discovering the ground truth from MCS/CS.
- We solve the non-trivial inference problem in EFusion, which infers the ground truth labels, the expertise level of each contributing smart device/crowd worker, and the difficulty level of questions.
- We perform a comprehensive evaluation of EFusion using real-world case studies as well as simulations. Evaluation results demonstrate that EFusion outperforms other popular methods, such as majority voting, the DS method proposed by dawid and skene [16], the method for

Conflict Resolution on Heterogeneous Data (CRH) [53], and the Generative model of Labels, Abilities, Difficulties (GLAD) [91].

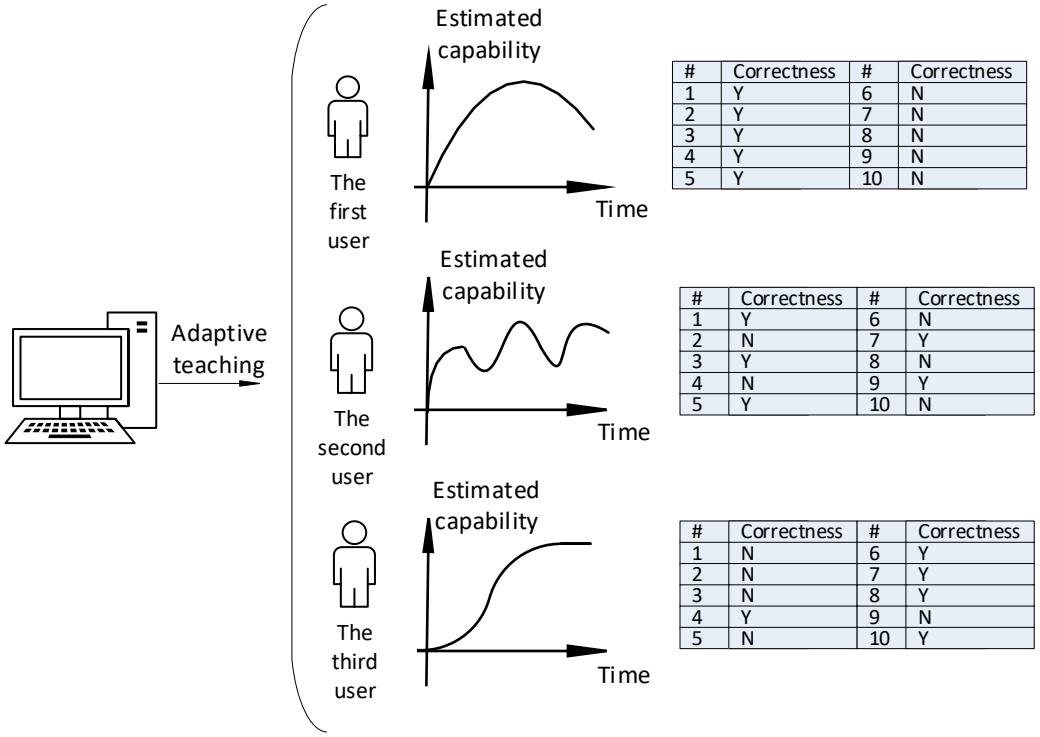
1.3.2 Interactive Learning Pattern Recognition

In order to enhance data quality in MCS/CS, many approaches [26, 75] have been proposed to identify the qualification of crowd workers. A widely-used method is (randomly) inserting questions with known answers during their participation. The samples with known ground truth are gold instances [15], which are used to evaluate crowd workers' domain knowledge and accordingly take proper actions on their answers. This approach, however, has a well-known pitfall: embedding gold instances in every crowd worker's annotation process may incur a high and sometimes unnecessary cost.

Another type of broadly-used method is training the crowd workers before assigning them to a given MCS task [69, 78]. This type of methods overcome the problem of embedding gold instances in the whole annotation process of crowd workers because a crowd worker without enough domain knowledge can get trained and crowd workers not performing well after the training can be excluded earlier. Nevertheless, in most existing solutions in this category, a fixed training set is used to train the crowd workers and the final selection of the crowd workers is mainly based on the number of questions they correctly answered in the training phase. We, via the following illustrative example, argue that the number of correctly answered questions during the training process may not be a good indicator for the final selection of crowd workers, and we can do much better if we consider each individual user's learning pattern.

Motivating example. Consider three different learning patterns in the training phase, as shown in Fig 1.3. The horizontal axis represents the time (or rounds of teaching), while the vertical axis denotes the capability estimation on the basis of user's correctly answered questions over time. If we only consider the number of correctly answer questions, the three users have identical performance w.r.t. the total number of correctly-answered questions during the training phase (e.g. 5 in the example) and they should be treated equally in the final selection of crowd workers. Nevertheless, they exhibit sharply different learning patterns: The first user correctly labels five questions in the beginning, but then makes mistakes in the follow-up questions; the performance of the second user varies all the time during the training process; for the third user, the number of correctly-labelled questions begins to increase after she learns a few examples and then her capability remains stable at a good level. It is worth noting that for the three users, although their learning curves are different, the corresponding areas under the curves are actually identical for they have the same number of correctly answered questions.

The reasons for different patterns could be many. For example, the first user may get confused or get tired quickly with more teaching instances; the second user may not learn effectively at all and consistently makes mistakes over time; the third user can quickly learn new knowledge and remain stable at a good capability level. Anyway, the hard-to-validate conjecture is irrelevant, since we are only interested in the observed learning patterns during the teaching phase. The point here is that the learning pattern can comprehensively reflect a user's capability of accepting, memorizing and applying new knowledge, which cannot be captured by the oversimplified metric: the total number



#	Correctness	#	Correctness
1	Y	6	N
2	Y	7	N
3	Y	8	N
4	Y	9	N
5	Y	10	N

#	Correctness	#	Correctness
1	Y	6	N
2	N	7	Y
3	Y	8	N
4	N	9	Y
5	Y	10	N

#	Correctness	#	Correctness
1	N	6	Y
2	N	7	Y
3	N	8	Y
4	Y	9	N
5	N	10	Y

Figure 1.3: Example learning patterns in the training phase.

of correctly answered questions. Clearly, users whose learning pattern in the training stage conforms to the third type in Fig 1.3 should be a better choice than users whose learning pattern follows the other two curves, particularly when we only have limited teaching samples and cannot afford training the users for a long time.

Motivated by the above observation, we propose Goldilocks[2], an interactive capability assessment framework for MCS, which adopts judicious user selection strategy to eliminate unsuitable users from the current MCS job at the early stage. Our goal is to select the participants whose performance follows the desired learning pattern to maximize the overall performance in the later formal MCS task. Being different from previous work that uses a fixed training set to train the candidates and selects them based on the number of correct answers [69], Goldilocks adopts an adaptive teaching model to filter the candidates with stronger ability to generalize new learned knowledge in the given tasks. The adaptive teaching strategy is tailored for each individual and is designed to exert the greatest learning potential of the candidate. Our philosophy is that if the "ultimate" capability of one candidate is still not satisfactory after the tailored teaching, we have enough evidence to believe that the candidate may not be qualified for the task and thus should be excluded from task assignment at the early stage.

We make the following contributions in this problem domain:

- We propose a new framework, Goldilocks, for adaptive learning pattern recognition in participatory MCS. This framework is largely different from previous works that use either gold instance in the whole process or simplified criteria (e.g. the total number of correct answers) in the participant selection.

- Goldilocks integrates the teaching and selection phases in a unified framework, and selects the qualified users as early as possible without assigning all the questions to each user. In this way, it significantly saves time and cost for a given MCS task.

- Based on the work in [44], we develop a new website over Amazon Web Services (AWS) [3], which uses deep learning to obtain the model parameters and automatically adjusts questions to maximize individual participants' learning performance. The website profiles the participants' learning pattern, which can be used for task assignment over MCS. Experiments on real-world datasets show that Goldilocks outperforms the baseline methods in both user profiling and the final participant selection.

1.3.3 Dynamic Worker Recruitment on MCS/CS Platforms

Crowdsourcing is an effective and efficient way to utilize human intelligence to solve real-world problems. Several crowdsourcing platforms, such as Appen [1] and MTurk [65], are gaining popularity due to their ease of use and more importantly their tools to support training. It is well known that not all crowd workers are qualified for given tasks, and as such training is needed before they undertake assigned tasks. Training crowd workers can actually lead to great savings by securing more reliable, high-quality results from crowdsourcing. Therefore, most crowdsourcing platforms

[2]The term is meant to emphasize that our ultimate goal is to identify "the right people for right tasks" rather than teaching and evaluating people's learning capability.

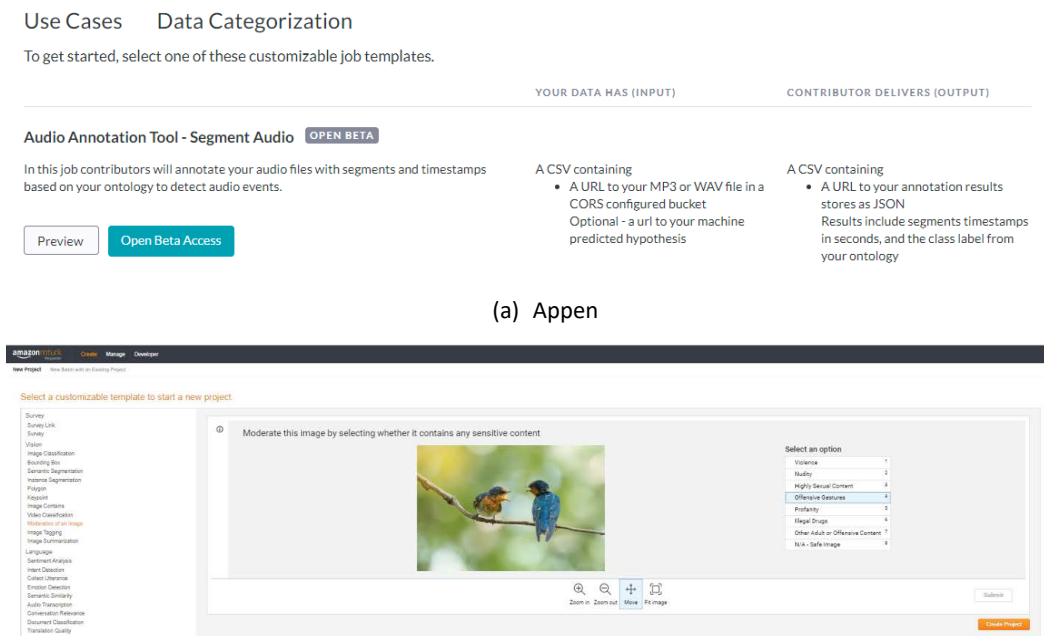


Figure 1.4: Example training templates on different crowdsourcing platforms.

support some forms of training. For instance, Appen and Mturk provide simple training templates to help requesters to build training samples, as shown in Fig. 1.4.

"Teach-before-use" can be realized with two main strategies: (a) static training and (b) dynamic training. In the first category, "static" means that these methods estimate different crowdsourcing candidates based on their final performance on the training set. In other words, the behavior of candidates during the training process is not considered in static methods. In [28], the candidates who give the most correct answers on the training set will be recruited. In [58], different models are built for the candidates according to their final performances on the training set, and the candidates will then be matched to the tasks that are most suitable for their capability.

In the second category, "dynamic" means that not only candidates' final performance but also their behavior in the training process will be taken into account when estimating their capability. In the dynamic methods, candidates' behavior (e.g., mouse movements, key presses) or responses on training sample are used to determine if a candidate should be assigned to the formal tasks [27, 47]. Compared to static methods, dynamic methods can be affected by more factors in the final recruitment result. The state-of-the-art method [44] in this category not only tracks a candidate's behavior at each step, but also selects personalized teaching samples based on the candidate's progress, with the hope of training a better worker at the end of the training process. However, since all the candidates are asked to finish the entire training process, such methods may result in a long training time and a waste of training resources.

The main drawback of static training is that the total number of correct answers alone may not accurately capture a candidate's capability of performing the crowdsourcing tasks. For example, consider two candidates who have the same number of correct answers in training. If one candidate

makes some errors in the beginning but no errors at the end of training and the other candidate persistently makes errors throughout the training process, the first candidate should be a better choice for the final task assignment because she clearly makes progress during the training.

While dynamic training overcomes the aforementioned deficiency in static training, existing dynamic training methods suffer from two problems: (a) nearsightedness, and (b) inefficient training. Regarding (a), existing methods [27, 47] assess candidates based on their behaviour or responses in *each step*[3] of training. However, since the final goal is to recruit qualified workers for the formal task, candidates should be evaluated holistically. For example, one candidate might give a wrong answer on the current training sample, but the knowledge she gains from this step may make her a strong candidate. In other words, we should assess a candidate in a longer time window rather than step by step in the training. Regarding (b), existing methods [44] require all candidates to complete the entire training process. In practice, as training goes on, some candidates may learn well while others may have little hope to become qualified for the formal task. If we can identify the latter type of workers early and remove them from further training, training time can be reduced, leading to higher efficiency in the recruitment of crowd workers.

Our Solution: We develop a hybrid deep learning framework for dynamic training, called *HybrTraining*, to address the above problems. Specifically, to avoid nearsightedness, we train a long short-term memory (LSTM) network for each candidate to extract her learning pattern from an interactive training process. Since LSTM accounts for the order of history inputs and can naturally model a candidate's learning pattern, we can infer how well the candidate learns from a batch of training samples based on the output of the LSTM model. To address the training efficiency issue, we train a Deep Q-Network (DQN), a widely-used reinforcement learning model, to train candidates with appropriate training questions. As a result, the total training time could be reduced without sacrificing training effectiveness.

Although LSTM is appropriate to identify different learning patterns and DQN is suitable to estimate the current expertise of candidates, we need to combine the outputs of these two networks appropriately when deciding if a candidate should be presented with a certain training sample. Simply assigning different weights to their results does not work, because doing so fails to adjust the dynamic training process according to the changing decision-making environment over time.

In our proposed solution, we take the output of LSTM as a part of the input to the DQN. In this way, both the learning pattern and the current expertise level of a crowd worker are considered as the training process goes on. In addition, we apply a feature extractor to extract the feature of each training batch and concatenate it with the feature of each candidate in each episode. In this way, the DQN can take actions based on both sources of information. At the same time, the two features are also included in the input of LSTM. In this way, the two networks can be trained together and cooperate with each other efficiently.

we make the following contributions in this research problem:

- We develop HybrTraining, a framework that integrates two distinctive deep neural networks: LSTM and DQN, to optimize recruitment processes on crowdsourcing platforms. Different from traditional dynamic training approaches, HybrTraining considers candidates' long-term

[3]Training is an iterative process and in each iteration, the candidate answers one or a batch of questions.

learning patterns and selects appropriate training samples during training.

- We consider candidates' learning pattern in the training and recruitment of crowdsourcing and design a novel state representation to integrate the extracted learning pattern into worker-sample matching.
- We evaluate HybrTraining with two real-world datasets. Experimental results show that HybrTraining outperforms several baseline methods in both candidate training and candidate selection efficiency. Furthermore, transfer learning from trained HybrTraining framework can greatly improve the training efficiency of the two deep neural networks in new application cases.

1.4 Thesis Outline

Using various machine learning methods, this thesis solves three critical data quality optimization problems in MCS/CS systems: automatic expert knowledge infusion to crowd answers, interactive learning pattern recognition of crowd workers, and dynamic worker recruitment. The rest of the thesis is organized as follow.

In Chapter 2, we analyze the generality and specificity among CS and different paradigms of MCS. Specifically, the inherent properties of CS, participatory MCS, and opportunistic MCS are investigated and compared.

In Chapter 3, we formulate the automatic expert knowledge infusion problem based on a probabilistic graphical model and then solve it with an expectation maximization (EM) based algorithm, named EFusion. EFusion can infer the ground truth of each question, the difficulty level of each question, and the expertise level of each candidate. The technical content of this chapter has been published in [43].

In Chapter 4, we design an interactive learning pattern recognition framework. We also optimize the process of both candidates training and candidates selection so that the final data quality of participatory MCS can be guaranteed in a high-efficient way. The technical content of this chapter has been published in [42].

In Chapter 5, we introduce a hybrid deep learning framework, which integrates two deep neural networks, LSTM and DQN, to optimize the worker recruitment process on crowdsourcing platforms. In this chapter, we also evaluate the transferability of the framework across different problem settings.

In Chapter 6, we compare the two models in Chapter 4 and Chapter 5.

In Chapter 7, we conclude the thesis and discuss future research.

Chapter 2

Comparison of CS and MCS Paradigms

2.1 Overview

With the fast development and wide deployment of various CS/MCS applications, there are more and more overlaps between the application scenarios of CS and MCS. Although CS and MCS share common properties in many aspects, they have their specific inherent features. Characterizing the application scenario is critical to build an efficient mathematical model for the problem and then solve it by appropriate methods. In this chapter, we analyze, compare, and conclude the inherent properties of CS, participatory MCS, and opportunistic MCS. This background is much needed to understand the context of the MCS/CS data quality optimization problems studied in this thesis.

2.2 Crowdsourcing (CS)

2.2.1 Inherent Properties of CS

When Jeff Howe and Mark Robinson firstly proposed the concept of CS [37], they defined it as: *the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call.* However, when CS is applied in different practical scenarios, this definition has expanded its scope in different perspectives, including deployment structure [7], problem resolution [19], or innovation application [8, 9].

From the various definitions of CS, we summarize the following inherent properties, which can be used to describe any given CS activity:

- Crowdsourcing is a type of participatory online activity in which an individual, institution, or organization publishes a task to a group of individuals via a flexible open call.
- To finish the task, the crowd workers are required to contribute their judgement, knowledge work, or experience.

Figure 2.1: Examples of sensors on current smart mobile devices. (a) iPhone X. (b) Phree.

- The requester will receive the satisfaction of a given type of need, be it economic, social recognition, or massive data, while the crowd workers can receive the compensation in the form of individual skill development, money, or spiritual satisfaction.

2.2.2 Representative CS Application Scenarios

The examples of successfully adopting CS can be easily found in everyday life. There are several famous and representative CS platforms, including:

- *InnoCentive [40].* InnoCentive is an online crowdsourcing innovation marketplace where money is offered to a global network problem solvers in exchange for the solution of various challenges from different organizations.
- *Amazon Mechanical Turk (AMT) [65].* On the AMT platform, an individual can choose to be either requester or crowd worker in different tasks. In other words, they can request the data or solution of a task by offering money to the crowd workers online, or they can also get paid by finishing the tasks published on this platform by the other users.
- *iStockphoto [41].* iStockphoto is an international online image sale platform where millions of users worldwide contribute photos, videos, and audio tracks in return for royalties.

2.3 Participatory MCS

MCS is a specific paradigm of CS where the contributors contribute sensing data collected with various mobile devices. Nowadays, more and more smart devices have embedded sensors and use wireless communication to upload sensing data to the Internet. For example, Fig. 2.1 shows different mobile devices and their equipped sensors.

2.3.1 Property Comparison

Specifically, participatory MCS requires the active involvement of individuals (e.g., taking a picture, reporting a phenomenon). For this reason, user interactivity is always involved in participatory MCS. Compared to CS, participatory MCS has the following same properties:

- The workflow of participatory MCS is the same as that of CS. Both of them need a requester to publish a clearly-defined crowd task by an open call. The task will then be finished by a group of crowd workers, who will then get the pre-defined task reward.
- The implementation of both CS and participatory MCS relies on the Internet; As a result, the network conditions in different areas may impact the quality of sensing data from different devices. Because of this reason, controlling the network load during data collection to avoid network congestion should be considered during the implementation of CS and participatory MCS.
- Both CS and participatory MCS require a relatively high user participative level. In other words, not only are crowd workers asked to upload the sensing data from their smart devices, but also they need to use their knowledge, judgement, or even learn necessary skills during the sensing data collection process to finish the tasks.

Based on the above-mentioned properties, we conclude that the diversity and unreliability of networks and participants make the data quality in CS and participatory MCS hard to guarantee. In addition, participatory MCS is more challenging than CS due to specific properties in participatory MCS:

- Compared to CS, participatory MCS task requests more specific type of data, i.e., the data collected by the sensors of smart mobile devices.
- The different requirements of uploaded data consequently lead to the different recruitment strategies between participatory MCS and CS. In the recruitment process of participatory MCS, the spatio-temporal information and the efficiency of workers, as well as the performance of mobile devices are emphasized more than in CS.

2.3.2 Participatory MCS Application Scenarios

As examples, we introduce three participatory MCS applications which are employed in different practical scenarios and require different kinds of sensing data.

- *CarTel [38].* CarTel is a participatory MCS system, in which the crowd devices equipped with a set of sensors are asked to dynamically summarize, filter, and prioritize the sensing data before uploading it. This system is used to monitor city traffic condition, analyze WiFi deployments of an area, or/and diagnose vehicle conditions.
- *CreekWatch [48].* CreekWatch is a water management platform developed by IBM Almaden Research Center. It asks crowd workers to collect information along creek in varied ways (e.g. taking photos, texting trash amount) and then upload the data to the platform. CreekWatch conducts evaluations of both its data contributors and data requesters to maintain the reliability of collected data.
- *DietSense [74].* In DietSense, users are asked to upload the photos of what they eat. Such information is documented and can be used to recommend dietary choices for other users.

2.4 Opportunistic MCS

Besides participatory MCS, opportunistic MCS is another paradigm of MCS that does not require a high level of user involvement. For example, the task asking for continuous location information without the explicit action of users belongs to opportunistic MCS [29].

2.4.1 Property Comparison

Although opportunistic MCS has the same workflow as CS, opportunistic MCS differs from CS in the following ways:

- Opportunistic MCS asks for only sensing data from mobile devices, while CS may require much richer information beyond sensing data.
- Users' involvement level in opportunistic MCS is much lower than that in CS. As a result, data optimization in opportunistic MCS always happens in the data extraction/analyzing phase instead of data collection phase.

Since participatory MCS and opportunistic MCS both fall in the category of MCS, both involve sensing data from various kinds of mobile devices. The main difference between the two is the different level of user involvement, which leads to different worker recruitment strategies. Specifically, worker recruitment for opportunistic MCS relies more on the spatial-temporal information of users or the prediction of user mobility, while worker recruitment for participatory MCS relies more on user-related factors, such as the knowledge, capability, or/and preference of a user.

2.4.2 Opportunistic MCS Application Scenarios

- *Nericell [64].* Nericell is a opportunistic MCS platform that uses various sensors on individuals' smartphone to collect real-time road condition data, so the traffic delays, honking levels, and potholes on roads can be detected and monitored in an efficient and low-cost way.
- *BikeNet [25].* BikeNet leverages opportunistic sensor network to collect the real-time data of road and environment (e.g., bumpiness of road and air pollution level) from the sensors equipped on users' bicycles. In this way, it can recommend riding routes to other users.
- *AirSense [21].* AirSense is an opportunistic MCS platform for air quality monitoring. The sensing data from crowd workers will be analyzed and aggregated to form air pollution heat maps, which will be sent back to end users.

2.5 Conclusion

In this chapter, we introduce the inherent properties of CS, participatory MCS, and opportunistic MCS. On the top of that, we compare those properties, discuss their difference in real-world applications, and then present representative application scenarios of CS, participatory MCS, and opportunistic MCS. In the following three chapters, we will show how to define and solve different MCS/CS data quality optimization problems based on these inherent properties.

Chapter 3

Automatic Expert Knowledge Infusion to Crowd Labels

3.1 Overview

With technological advances in mobile smart devices, MCS recruits not only crowd of mobile users but also the sensing crowd consists of "smart" devices/programs that possess machine intelligence to capture data of interest and input their own judgment (i.e., intelligence) to facilitate the processing of big data. One example of the machine intelligence is the smart cameras that can recognize human faces or detect urgent events such as a car collision. Another example is WeChat mini program pet recognition, which can tell, with a level of confidence, the breeds of dogs or cats from the pictures taken with phones. Those smart devices/programs have a certain level of intelligence, and as such they can be treated as another source of information critical to MCS applications. In the MCS work flow under this situation, a core challenge is that the labelled data from the crowd may be error prone. The correctness of answers from humans is subject to their domain knowledge; the accuracy of answers from smart devices/programs is limited by their lower computational power and storage. Therefore, the accuracy of their answers is generally inferior than that obtained in cloud data centers.

In this chapter, following the idea that "answers from experts are assumed to be more accurate than those from crowd workers, and the knowledge from experts could help us make better decisions", we propose a new quality assurance mechanism, called EFusion, to automatically infuse the knowledge from experts into the uncertain data from the crowd. For the labelled data from crowd workers with unknown expertise levels, we assign a small portion of data to experts and ask experts to judge the labels. The answers from experts are explored to infer the most likely expertise level for each crowd worker, as well as the ground-truth answers. Since the number of questions that experts need to answer is small, the extra cost of using experts can be well controlled.

3.2 Related Work

Existing work can be roughly divided into two categories: discovering truth in MCS and utilizing expert knowledge to improve quality of answers from crowd sourcing.

3.2.1 Truth Discovery in MCS

In this category, the goal of proposed solutions is to handle the situation where the ground truth is unknown and data contributed from multiple workers in MCS are inconsistent or even contradictory. Wang et al. [87] used the EM algorithm to determine whether one user's answer can be accepted as the truth. Peng et al. [70] extended the work in [87] by establishing a connection between the quality of user's sensing data and their reward. Liu et al. [56] estimated the truth based on the estimation of data quality from different users in an online manner. To get the accurate estimation of user's reliability, a model which combines multiple properties is also proposed in [53]. In a more specific setting where there are correlations among monitored entities, Meng et al. [59] formulated an optimization problem to find truth. Under the same assumption, Wang et al. [89] provided a scalable approach that exploits dependencies between observed variables to improve fact-finding accuracy of social sensing data. Research in the first category also includes the works aiming to protect users' privacy which may be compromised by truth discovery methods. Cheng et al. [61] proposed a cloud-enabled privacy-preserving truth discovery (PPTD) framework for crowdsensing. Miao et al. [62] further extended the work in [61] by designing L-PPTD and L^2-PPTD models, which incur less overhead to crowd workers.

All the above research is mainly focused on obtaining high quality sensing data, with little attention paid to improving the quality of final results by infusing professional knowledge to the initial answers.

3.2.2 Professional Knowledge Infusion in Crowd Data Quality Enhancement

In this category, the problem of infusing professional knowledge into crowd workers has been investigated in [39,66]. The key idea is to combine answers from different groups (workers and experts). This concept has widely used in active learning to obtain the effective model on truth discovery [4,66]. Tang et al. [82] proposed a semi-supervised method to combine the labels from experts and workers so that the consensus labels can be inferred. Sheshadri and Lease [77] provided an open source shared task framework to compare the performance of various statistical consensus methods. Both solutions assumed that experts always know the ground truth so that their answers provide labels to a subset of tasks for the workers to learn. In addition to these works, Aroyo and Welty [2] aggregated the labels from experts and workers by k-score to train a model for semantic interpretation of sentences. Their work, however, mainly focused on the truth estimation for semantic recognition, which may not be applicable to general cases.

EFusion belongs to this category because it can infuse expert knowledge into crowd workers' unreliable answers. However, the purpose and usage of expert labels in EFusion are different from those in existing methods. In active learning, the purpose of incorporating expert opinions is to

select the most informative question for participants to label, so that a better classification model could be trained. In contrast, the expert's opinion is used to directly infer more truthful labels in EFusion. Furthermore, EFusion can *automatically* infer the ground truth and expertise level of workers in MCS. At last, the expert opinion in our framework does not represent ground truth as in other existing methods.

3.3 Details of EFusion

3.3.1 Problem Formulation

Let **Expert**, be a group of professionals who have expert domain knowledge. Here we do not distinguish individual experts in this chapter, and use Expert to denote them as a whole. We consider a batch of m classification tasks in MCS to be labeled. To improve the quality of answers from workers, we try to gain some expertise knowledge on the batch by asking Expert to label a (small) subset of k instances, where $k \leq m$. Let the total number of crowd workers be n.

Denote the answer of crowd worker i for question j by $l_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$. Denote Expert's answers as $e_l, 1 \leq l \leq k$. For ease of presentation, we assume that l_{ij} and e_l are binary values, while the model can be easily extended to multi-class classification tasks, as discussed later in Section 3.3.5. Note that $l_{ij}(1 \leq i \leq n, 1 \leq j \leq m)$ and $e_l(1 \leq l \leq k)$ are inputs to EFusion.

Given the batch of questions, we assume that Expert has a higher chance of giving correct answers (answers are closer to the ground truth) than crowd workers. We thus introduce the concept of *expertise level*: the higher the expertise level, the higher the probability of returning a correct answer. Associated with a crowd worker i is the expertise level $\alpha_i \in (-\infty, +\infty)$, and associated with Expert is the expertise level $\alpha_E \in (0, +\infty)$, where $+\infty$ means that the labeler always answers correctly and $-\infty$ means that the labeler always answers incorrectly. A positive/negative expertise value implies the labeler is more likely to return a correct/incorrect answer for a specific question. We assume *a priori distribution* for α_i and assume a priori value for α_E, which is larger than the prior mean of α_i. Since the difficulty level of a question may impact the probability that a correct answer can be obtained, we introduce a positive parameter $\beta_j(1 \leq j \leq m)$ to denote the difficulty level of question j. We assume *a priori distribution* for β_j. $\beta_j \in (0, +\infty)$, where a small β_j means that it is easier to answer the question correctly by the same worker. Note that the difficulty of a question is inherent to the question and independent of the crowd workers. The likelihood of a truthful answer from worker i to question j is jointly decided by i's expertise level and j's difficulty level. This is represented using a probabilistic graphic model illustrated Fig. 3.1. The notations used in the chapter are listed in Table 3.1.

Problem 1. ***The goal of EFusion:*** *Given the graphical model in Figure 3.1 and the observed values $l_{ij}(1 \leq i \leq n, 1 \leq j \leq m)$ and $e_l(1 \leq l \leq k)$, what are the ground-truth answers $Z_j(1 \leq j \leq m)$? what are the posteriori estimates of α_i? what are the posteriori estimates of β_j?*

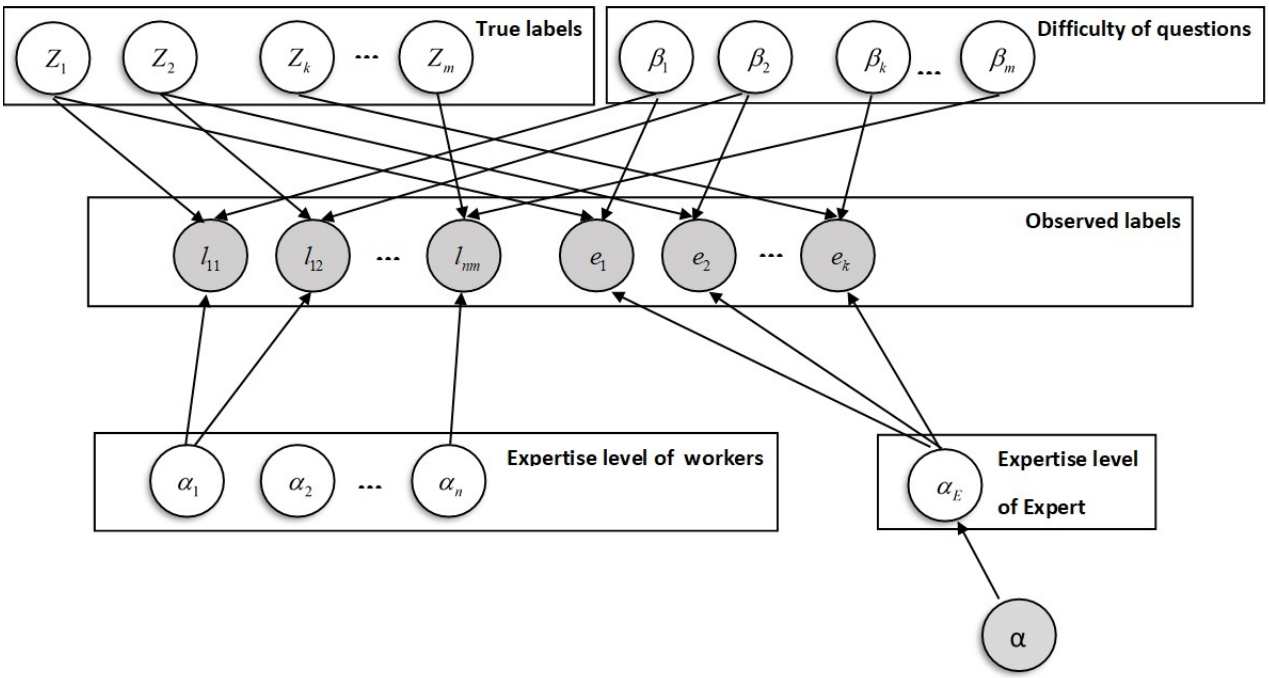


Figure 3.1: Graphical model of Z_j, α_i, α_E, l_{ij}, β_j and e_j. Note that only the shaded part are observed.

Table 3.1: Summary of main notations

Notation	*Description*
α_i	the expertise level of worker i
l_{ij}	the answer worker i gives for question j
α_E	the expertise level of Expert
β_j	the difficulty of question j
n	the number of workers
m	the number of questions in the batch
k	the number of questions answered by Expert
Z_j	the ground truth answer for question j
e_l	the answer of Expert for question l

3.3.2 Probability of Correct Answers

In general, the more difficult a question, the lower the probability that a correct answer can be obtained; and the higher the expertise level, the higher the probability that a correct answer can be obtained. Based on this intuition, we propose to model the probability that a correct answer is returned using a logistic function. It is worth noting that the logistic function and its variants have been widely used in financial domain for calculating probability of correct prediction [14] as well as in similar problem settings [91].

In particular, the probability that Expert returns the correct answer to question j is modeled as:

$$p(e_j = Z_j | Z_j, \alpha_E, \beta_j) = \frac{1}{1 + e^{-\alpha_E/\beta_j}}, j \in \{1, \ldots, k\}. \tag{3.1}$$

The probability that crowd worker i returns the correct answer to question j is modeled as:

$$\begin{aligned} p(l_{ij} = & Z_j | Z_j, \alpha_i, \beta_j) = \frac{1}{1 + e^{-\alpha_i/\beta_j}}, \\ & i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}. \end{aligned} \tag{3.2}$$

3.3.3 Main Algorithm

We use the Expectation Maximization (EM) algorithm to solve Problem 1. To simplify notation, we denote $\bar{\boldsymbol{\alpha}}_W = \{\alpha_1, \ldots, \alpha_n\}$, $\bar{\boldsymbol{\alpha}} = \bar{\boldsymbol{\alpha}}_W \cup \{\alpha_E\}$. Starting with initial prior values of α_1, ..., α_n, α_E, β_1, ..., β_m, the EM algorithm iterates through the following two basic steps: the E step and the M step.

The E Step

Given observed values of $\{l_{11}, \ldots, l_{nm}\}$ and $\{e_1, \ldots, e_k\}$, we first calculate the posterior probabilities of all true answers Z_j using the estimated $\bar{\boldsymbol{\alpha}}$ and β_j's from the last M step. Denote $l_{\cdot j} = \{l_{1j}, \ldots, l_{nj}\}$. Without loss of generality, we assume the first k questions are answered by Expert as well as crowd

workers. For each $j \leq k$, we have:

$$\begin{aligned} & p(Z_j|l_{.j}, e_j, \bar{\boldsymbol{\alpha}}, \beta_j) \\ & \propto p(Z_j)p(e_j|Z_j, \alpha_E, \beta_j) \prod_i^n p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j) \end{aligned} \tag{3.3}$$

where $p(Z_j|\bar{\boldsymbol{\alpha}}, \beta_j) = p(Z_j)$ because of the conditional independence assumptions from the graphical model. Similarly, for each question $j(k < j \leq m)$ answered only by crowd workers, we have:

$$\begin{aligned} & p(Z_j|l_{.j}, \bar{\boldsymbol{\alpha}}, \beta_j) \\ & \propto p(Z_j) \prod_i^n p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j) \end{aligned} \tag{3.4}$$

The M Step

The auxiliary function Q is defined as the expectation of joint log likelihood of the observed and hidden variables ($l_{.j}$,Z_j), given the parameters ($\bar{\boldsymbol{\alpha}}_W$,$\beta_j$).

$$\begin{aligned} Q(\bar{\boldsymbol{\alpha}}_W, \beta_j) =& E[\ln \prod_{j=1}^{k} p(e_j, l_{.j}, Z_j|\bar{\boldsymbol{\alpha}}, \beta_j) \prod_{j=k+1}^{m} p(l_{.j}, Z_j|\bar{\boldsymbol{\alpha}}, \beta_j)] \\ =& \sum_{j=k+1}^{m} E[\ln p(Z_j)] + \sum_{i=1,j=1}^{i=n,j=k} E[\ln p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j)] + \\ & \sum_{j=1}^{k} E[\ln p(Z_j)p(e_j|Z_j, \alpha_E, \beta_j)] + \sum_{i=1,j=k+1}^{i=n,j=m} E[\ln p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j], \end{aligned} \tag{3.5}$$

where the expectation is computed from the posterior probabilities in the E step. Let α^p, β^p denote the α and β estimated by the previous iteration.

The first part of Equation (3.5) can be expanded to Equation (3.6) as follows:

$$\begin{aligned} & \sum_{j=k+1}^{m} E[\ln p(Z_j)] \\ =& \sum_{j=k+1}^{m} (p(Z_j = 0|\mathbf{L}, \alpha^p, \beta^p) \ln p(Z_j = 0) + \\ & p(Z_j = 1|\mathbf{L}, \alpha^p, \beta^p) \ln p(Z_j = 1)), \end{aligned} \tag{3.6}$$

in which $\mathbf{L}$ means all the labels given for the $(k+1)$-th to m-th questions.

The second part and the last part of Equation (3.5) have similar form (i.e., the only difference

is on the range of j value) and thus can be expanded in similar form shown in Equation (3.7):

$$\begin{aligned}&\sum_{ij} E[\ln p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j]\\ =&\sum_{ij}(p(Z_j = 0|\mathbf{L}, \alpha^p, \beta^p) \ln p(l_{ij}|Z_j = 0, \alpha_i, \beta_j) +\\ &p(Z_j = 1|\mathbf{L}, \alpha^p, \beta^p) \ln p(l_{ij}|Z_j = 1, \alpha_i, \beta_j)),\end{aligned} \tag{3.7}$$

where $p(l_{ij}|Z_j = 0, \alpha_i, \beta_j)$ and $p(l_{ij}|Z_j = 1, \alpha_i, \beta_j)$ can be attained with Equation (3.1). Without causing confusion, we here slightly abuse the notation by using $\mathbf{L}$ to denote all the labels given for the questions in the same range as that of j.

The third part of Equation (3.5) can be expanded to Equation (3.8):

$$\begin{aligned}&\sum_{j=1}^{k} E[\ln p(Z_j)p(e_j|Z_j, \alpha_E, \beta_j)]\\ =&\sum_{j=1}^{k}(p(Z_j = 0|\mathbf{L}, \alpha^p, \beta^p) \ln p(e_j|Z_j = 0, \alpha_E, \beta_j) +\\ &p(Z_j = 1|\mathbf{L}, \alpha^p, \beta^p) \ln p(e_j|Z_j = 1, \alpha_E, \beta_j)),\end{aligned} \tag{3.8}$$

where Equation (3.1) is used to calculate $p(e_j|Z_j = 0, \alpha_E, \beta_j)$ and $p(e_j|Z_j = 1, \alpha_E, \beta_j)$.

Then we use gradient descent to find the values of $\bar{\boldsymbol{\alpha}}_W, \beta_j$ to maximize the function Q. Note that we assume the value of α_E is known and thus we do not update α_E in the EM algorithm. The algorithm iterates through the E step and the M step until convergence. Here, convergence means that either the number of iterations reaches a given maximum threshold or the difference in learned parameters between consecutive iterations falls within a given small threshold.

The posterior probabilities of Z_j values are obtained after the **last** (i.e., the one before the algorithm stops) E step. After the last E step, for each question $j(1 \leq j \leq m)$, we use Equation (3.1) and Equation (3.2) to calculate the probability of getting correct labels from workers and Expert, respectively, and then choose the label with the highest overall probability as the final label for this question.

3.3.4 Priors on Parameters

Both $\bar{\boldsymbol{\alpha}}_W$ and β_j are continuous random variables in EFusion. According to probability theory, the distribution of such variables generally conforms to the normal distribution. So in the implementation we used Gaussian priors on $\bar{\boldsymbol{\alpha}}_W$, and truncated Gaussian priors on β_j's such that all β_j values are positive. In particular, we assume that the expertise level of each worker follows a normal distribution $\mathcal{N}(\mu_1, \sigma_1^2)$, and the difficulty of each question follows a truncated normal distribution $\mathcal{N}(\mu_2, \sigma_2^2)$. The expertise level of Expert is assumed to be known in advance, and is denoted as $\alpha_E = \alpha$. We set $\alpha >> \mu_1$ since Expert has much more expertise than crowd workers on average. The value of prior probabilities of each class (i.e., $p(Z_j)$) is also influential on the performance of EFusion. So the task publishers who have some domain knowledge on the assigned tasks can acquire better estimated parameters by changing $p(Z_j)$ in E-step.

3.3.5 Further Discussion

EFusion can be easily extended to handle scenarios involving multi-class classification. In this section, we discuss how this can be done.

Suppose that answers are grouped into S categories. Given a question j, let its correct answer be Z_j. The probabilities that Expert and worker i gives the correct answer can still be calculated with Equation (3.1) and Equation (3.2), respectively. We assume incorrect answers are equally probable here. This is reasonable because when we do not have enough knowledge, the best that people can do is to follow the "principle of insufficient reason [24]". In other words, we have:

$$p(e_j = s' | Z_j, \alpha_E, \beta_j) = \frac{e^{-\alpha_E/\beta_j}}{(S-1)(1+e^{-\alpha_E/\beta_j})}, \quad j \in \{1, \ldots, k\}, s' \neq Z_j. \tag{3.9}$$

$$p(l_{ij} = s' | Z_j, \alpha_i, \beta_j) = \frac{e^{-\alpha_i/\beta_j}}{(S-1)(1+e^{-\alpha_i/\beta_j})}, \quad i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}, ... \tag{3.10}$$

Consequently, in the M step of EFusion, Equation (3.9) and Equation (3.10) should be used to calculate $p(e_j|Z_j, \alpha_E, \beta_j)$ and $p(l_{ij}|Z_j, \bar{\boldsymbol{\alpha}}_W, \beta_j)$ in Equation (3.5), and the Q value in each iteration. The rest of EFusion remains unchanged.

3.4 Performance Evaluation

In this section, we evaluate the performance of EFusion using two case studies as well as comprehensive simulations. We also compare its performance with other baseline methods described below. During the process of collecting experimental data for evaluation, (1) all the participants are anonymous and cannot be identified, (2) no staging or manipulating is involved, and (3) tasks are made public online so anyone who has Internet can participate.

3.4.1 Baseline Methods

Although the works in Section 3.2.1 are all designed for truth discovery in CS/MCS, the model in [87] is focused on finding truth from individuals instead of from online groups. The method in [87] establishes a connection between the quality of sensing data and user's reward during collecting data. However, our EFusion is designed for the data integration step after the data collection, and no user's reward is involved. Therefore, this method is not comparable. For the similar reason, other works discussed in Section 3.2.1 are designed for the scenarios that ask for contextual information, such as the place that users collect the sensing data [56], users' historical skills [59], correlation among sensing questions [89], or users' encrypted data [61, 62]. This information is not accessible in our scenario. In Section 3.2.2, expert knowledge is infused to crowd data for semantic interpretation of sentences in [2], or expert knowledge is infused as ground truth. In EFusion, there is no ground truth and expert's opinion is used to find it.

Based on the above analysis, for comparison, we implemented the following baseline methods, which are designed for truth discovery in MCS/CS in the similar scenario as EFusion, and no expert knowledge is taken as ground truth.

- Majority voting (MV): The final answer to a question is the answer that appears the most of times among all crowd workers. And all contributors are treated equally, so there is no "Expert".
- The DS method [16]: It uses full confusion matrices to denote the expertise of each contributor, the EM algorithm is used to obtain maximum likelihood estimates of ground truth of polytomous classes problem under medical background.
- Conflict Resolution on Heterogeneous Data (CRH) [53]: This is a general model for truth discovery from multiple sources that might have different data types. It uses an optimization framework where truths and source reliability are defined as two sets of unknown variables, with the objective to minimize the overall weighted deviation between the truths and the multi-source observations where each source is weighted by its reliability.
- Generative model of Labels, Abilities, Difficulties (GLAD) [91]: GLAD makes decisions regarding ground truth, difficult level of questions, and expertise level of workers, using a similar graphical model as in EFusion but without any inputs from Expert.

3.4.2 Performance Metrics

In the datasets used in the case studies and simulations, the ground-truth answer of each question is given. This allows us to compute exactly the estimate errors of EFusion and other baseline methods. We adopt the following measures to evaluate the performance of different methods.

- Accuracy: It is defined as the ratio of correct answers from different methods over the total number of questions in the batches.
- F-measure: It is the harmonic mean of precision and recall, where precision is the proportion of predicted positive labels and real positives labels, and recall is the proportion of real positive labels that are correctly predicted positive [72]. To disclose more details of F-measure, we also use tables to list the values of precision and recall.

3.4.3 Case Studies

We perform two case studies in different participatory MCS application domains.

Case 1: Detection of Distracted Driving Distracted driving is a main cause of accidents in our daily transportation. Even if many cities have law enforcement, most distracted drivers remain uncaught due to the high cost in detecting distracted driving. One solution is to launch a MCS campaign by recruiting volunteers or using smart cameras on streets to report distracted drivers and upload images during a certain time period.

To *emulate* a MCS campaign, we use a public dataset [17] which consists of 606 photos, each recording a potential distracted driver, as examples shown in Figure 3.2. The images and the ground

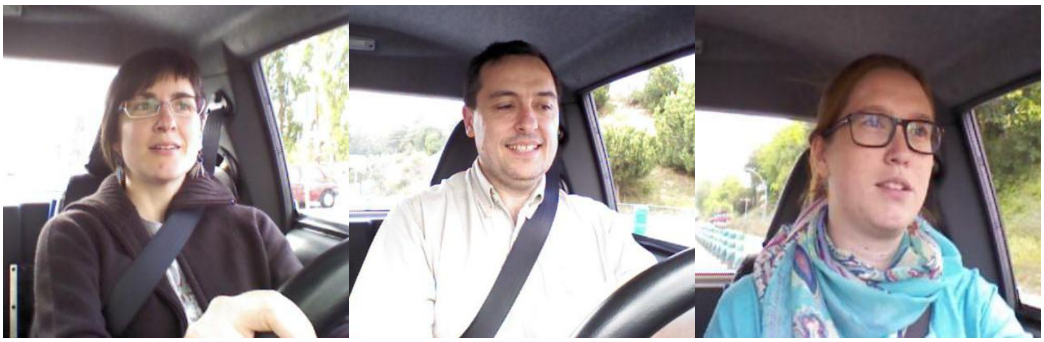


Figure 3.2: Example images of distracted driving.

truth labels can be found from [17]. To emulate the action of workers, we posted the labelling task with a fee in CrowdFlower [1] and asked crowd workers to determine whether or not the driver in each of the 606 photos is fatigue or distracted[1]. In this way, we collected 15150 answers in total from 25 workers, and apply different methods to determine the final answers.

Case 2: Mushroom Classification There are thousands of mushroom poisoning cases across the United States each year, and diagnosis and management of mushroom poisoning is a challenging problem for physicians [90]. Participatory MCS can be used to monitor and report poisonous mushrooms in the wilderness. In order to emulate this application, we selected 150 pictures of different mushrooms, known to be either edible or poisonous. The pictures are then dispatched to workers through CrowdFlower. Each worker was asked to answer whether or not the mushrooms in the 150 pictures are poisonous. We instructed on CrowdFlower that the workers should answer the question only based on their own experience and should not use Internet/literature search. In addition, we offered a very low payment (i.e., 2 cents per judgment) to workers so that they are not incentivized to spend much time on finding the correct answers. We collected answers from 25 different workers.

Note that in both case studies, ground truths are only used to evaluate the performance of different methods and are unavailable to the workers. In the case studies, we set $p(Z_j)$ to 0.5, β_j to 1 for $j = 1, \ldots, m$, where m is 606 and 150 for the first and second studies, respectively. And set $\alpha_E = 3$ so that Expert has around 90% chance to give a correct answer for each question. The number of questions answered by the Expert has an impact on the performance of EFusion. Since we know the ground-truth labels, answers from Expert are simulated by returning the correct answers with 0.9 probability.

The results for the first case study are summarized in Fig. 3.3, where Fig. 3.3a and Fig. 3.3b demonstrate the performance of different methods with different numbers of workers in terms of accuracy and F-measure. In all the figures, EFusion is shortened as EF. Note that we only draw the performance of EFusion when Expert answers 40% questions, and use an error bar to present the accuracy and F-measure achieved by EFusion when the percentage of questions answered by Expert changes from 20% to 80%. In that way, we can observe the differences in the performance between EFusion and other baseline methods as the Expert coverage rate changes. The precision and recall

[1] An image labeled as *false* by a worker implies that in the real-world MCS scenario the worker does not upload the image.

of different methods are summarized in Table 3.2.

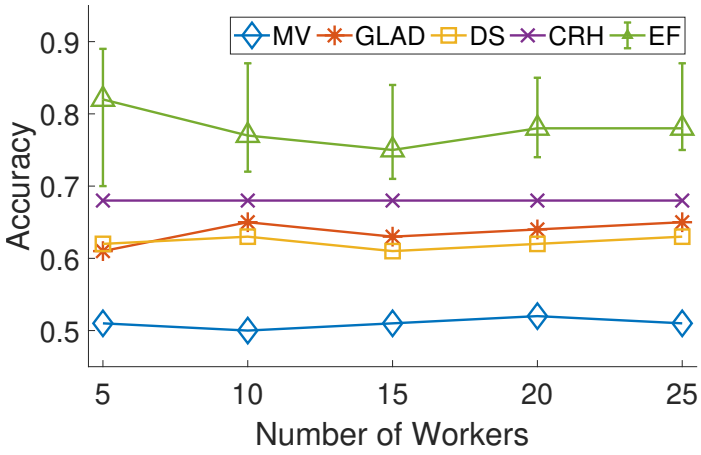
Table 3.2: Performance Comparison on Precision and Recall Under Different Number of Workers on Detection of Distracted Driving

		#Workers=5		**#Workers=10**		**#Workers=15**		**#Workers=20**		**#Workers=25**	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Method	*MV*	0.4528	0.5455	0.4364	0.5455	0.4561	0.5909	0.4727	0.5909	0.4510	0.5227
	GLAD	0.5099	0.5359	0.4864	0.5957	0.6462	0.5725	0.6591	0.5800	0.6364	0.5957
	DS	0.5524	0.5318	0.6818	0.5556	0.7045	0.5741	0.7273	0.5714	0.6364	0.5714
	CRH	0.6818	0.6250	0.6818	0.6250	0.6818	0.6250	0.6818	0.6250	0.6818	0.6250
	EF(20%)	0.7273	0.6400	0.7273	0.6667	0.7027	0.5909	0.7143	0.6818	0.7209	0.7045
	EF(40%)	0.8409	0.7708	0.7442	0.7273	0.7727	0.6939	0.7955	0.7292	0.7727	0.7391
	EF(60%)	0.8864	**0.7959**	0.8409	0.7400	0.7805	0.7273	**0.8636**	0.7451	**0.8864**	0.7500
	EF(80%)	**0.9459**	0.7955	**0.9091**	**0.8163**	**0.8500**	**0.7727**	0.8409	**0.8222**	0.8444	**0.8636**

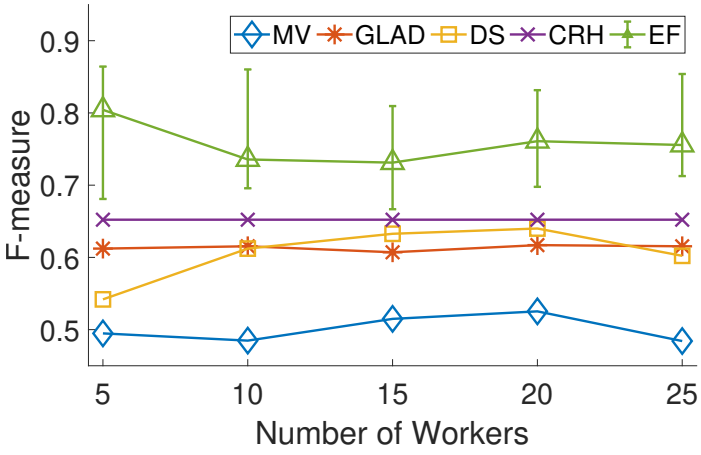
Note: the value in the parentheses after "EF" denotes the coverage rate of Expert in EFusion.

As shown in Figure 3.3a and Figure 3.3b, EFusion outperforms all the baseline methods for different numbers of workers. On average, EFusion has about 10% improvement over the best baseline CRH in terms of accuracy and F-measure. Majority voting performs the worst among all the methods in this case study. Notably, both the accuracy and F-measure of CRH remain the same as the number of workers changes. This is because the performance of CRH is mainly subject to the data heterogeneity, while in this case only one type of categorical data is involved. According to CRH frame, the algorithm can infer a part of reliable workers, and then detect truth only based on them. It is notable that having more workers does not help for all schemes in this case, we make the following explanation for this phenomenon: Since the accuracy is around fifty-fifty, it appears roughly half of the workers regardless the total number of workers give the correct results. Therefore, increasing the number of workers does not help much in all schemes. For EFusion, we are interested in studying the trade-off between accuracy and knowledge infusion. As such, we introduce *coverage rate of Expert*, defined as the percentage of questions answered by Expert, to capture knowledge infusion. The results are presented in Fig. 3.3c. We can see that as expected, the more questions answered by Expert, the more accurate the final results. With more questions answered by Expert (e.g., higher than 70%), the improvement in final accuracy diminishes. In other words, the marginal utility of infusing expert knowledge decreases. Such a relationship can be used to guide decisions on how much expertise knowledge is needed. Furthermore, EFusion has indeed utilized the knowledge from crowd workers as the accuracy is consistently better than that relying on Expert alone (e.g., 0.9 x coverage rage).

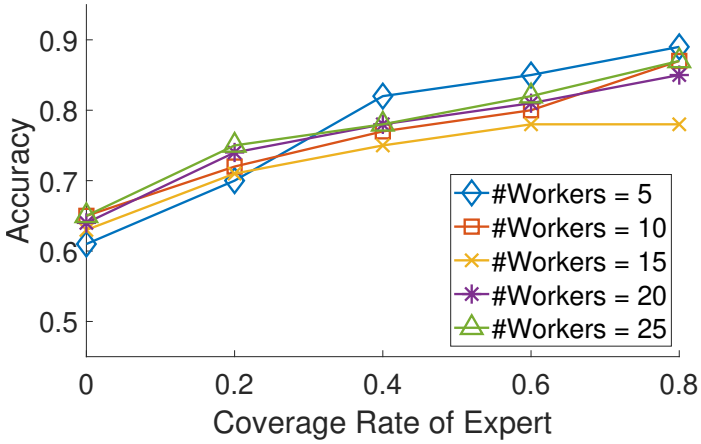
Figs. 3.4a and 3.4b summarize the accuracy and F-measure of EFusion and baseline methods in the second case study. We can observe that all the methods perform much better than majority voting. Among the five methods, EFusion achieves the highest accuracy and F-measure. In contrast to the first case study, where crowd workers appear to be equally uncertainty about the answers, a slight majority of the crowd workers know the correct answers in this case. As a result, with the growth of the number of workers, both GLAD and EFusion show improvement in accuracy and F-measure. Fig. 3.4c demonstrates EFusion's performance when varying the coverage rate of Expert from 20% to 80% in the second case study. We observe similar trend as in Fig. 3.3c where increasing of Expert's coverage rate improves the accuracy. Expert in the second case study, however, brings less benefit than in the first case study. This may be attributed to the fact that labeling mushrooms



(a) Accuracy of different methods on detection of distracted driving.



(b) F-measure of different methods on detection of distracted driving.



(c) Performance of EFusion on detection of distracted driving when varying the Expert coverage rate.

Figure 3.3: The performance and comparison of different methods on detection of distracted driving when $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$, $\alpha_E = 5$.

is relatively easier than labeling distracted drivers. Thus, more reliable answers are available from the crowd in the second case study. Table 3.3 shows the precision and recall of different methods on mushroom classification when the number of workers changes.

Table 3.3: Performance Comparison on Precision and Recall on Mushroom Classification

		#Workers=5		**#Workers=10**		**#Workers=15**		**#Workers=20**		**#Workers=25**	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Method	*MV*	0.4286	0.5745	0.5472	0.6170	0.4545	0.5319	0.4902	0.5319	0.5019	0.5957
	GLAD	0.7447	0.6604	0.7447	0.7000	0.7174	0.7021	0.8085	0.7755	0.8511	0.7273
	DS	0.7347	0.7660	0.6667	0.7660	0.7021	0.7500	0.7056	0.7660	0.7500	0.8298
	CRH	0.8298	0.7647	0.8298	0.7647	0.8298	0.7647	0.8298	0.7647	0.8298	0.7647
	EF(20%)	0.7955	0.7447	0.7447	0.7143	0.766	0.75	0.8511	0.7692	0.8511	0.8000
	EF(40%)	0.8298	0.8125	0.8511	0.8163	0.8837	0.8085	0.8936	**0.8571**	0.9149	0.8269
	EF(60%)	0.8222	0.7872	0.8936	0.8400	0.9149	**0.8431**	0.9070	0.8298	0.9524	0.8511
	EF(80%)	**0.9574**	**0.8824**	**0.9149**	**0.8776**	**0.9362**	0.8302	**0.9091**	0.8511	**0.9574**	**0.9184**

Note: the value in the parentheses after "EF" denotes the coverage rate of Expert in EFusion.

Since in the empirical studies we do not know the ground truth about workers' expertise and the difficulty levels of questions, we cannot evaluate the performance of EFusion with respect to the accuracy of inferred workers' expertise and questions' difficulty levels. Next, we perform controlled simulations to evaluate these two aspects.

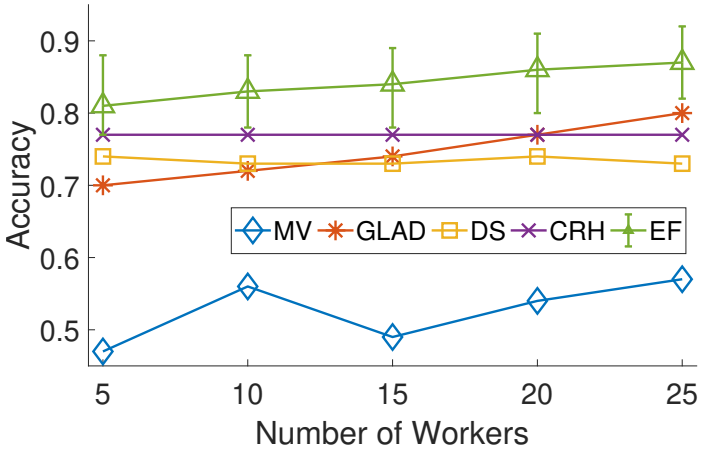
3.4.4 Simulation

In the simulation, 2000 questions are generated in total, each having a binary answer. For ground-truth label of each question, we set its value to 0 or 1 randomly with equal probability. A total of 25 workers were simulated.

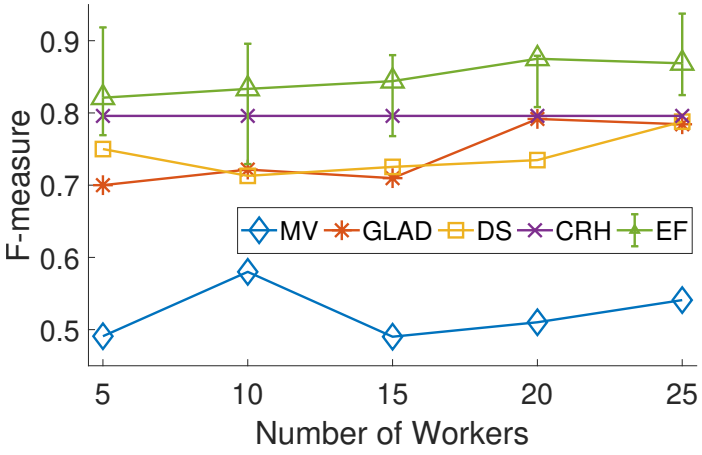
We first simulate a base case where majority voting can achieve a reasonably good accuracy. For this purpose, we set the ground-truth values of workers' expertise α_W following normal distribution $\mathcal{N}(\mu = 1, \sigma = 0.2)$ and set the value of Expert expertise α_E to $5(\gg \bar{\boldsymbol{\alpha}}_W)$. The difficulty levels of the 2000 questions in the batch are drawn from an independent truncated normal distribution $\mathcal{N}(\mu = 5, \sigma = 1)$. For each parameter setting, the simulation is repeated 50 times to smooth out the variability among trails. In each trail, we computed the accuracy, F-measure for all methods, as well as the correlation between the estimated expertise level, the estimated question difficulty and their ground truth. The results in all the figures of this section reflect the mean values from the 50 trails.

The accuracy and F-measure with 5 to 25 workers and the accuracy under different Expert's coverage rates are summarized in Fig. 3.5. The simulation results further confirm that: 1) For the accuracy and F-measure of the five methods, EFusion outperforms all the baselines under different numbers of workers. 2) As the coverage rate of Expert increases, EFusion achieves higher accuracy with different number of workers. More details of the precision and recall of the simulations are given in Table 3.4.

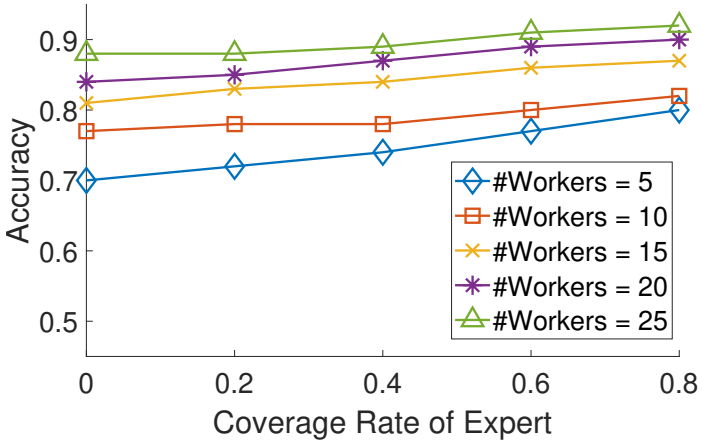
To evaluate the accuracy in estimating worker's expertise and question difficulty, we compute the root-mean-square error (RMSE) between the worker's expertise (/question's difficulty levels) estimated by EFusion and the corresponding true values. Fig. 3.6 shows the RMSE with different numbers of workers. It can be seen that both the estimated workers' expertise and the estimated



(a) Accuracy of different methods on mushroom classification.



(b) F-measure of different methods on mushroom classification.



(c) Performance of EFusion on mushroom classification when varying the Expert coverage rate.

Figure 3.4: The performance and comparison of different methods on mushroom classification when $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$, $\alpha_E = 5$.

question difficulty are closer to the corresponding true values as the number of workers increases. When the number of workers is 25, the RMSE for expertise level is 0.0467, and the RMSE for the difficulty levels is 0.0694.

Table 3.4: Performance Comparison on Precision and Recall Under Different Number of Workers in Simulation

		#Workers=5		**#Workers=10**		**#Workers=15**		**#Workers=20**		**#Workers=25**	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Method	*MV*	0.5049	0.532	0.501	0.5322	0.5127	0.5209	0.544	0.5616	0.5225	0.5308
	GLAD	0.6385	0.636	0.6117	0.5949	0.636	0.6262	0.643	0.6024	0.6008	0.6356
	DS	0.6193	0.6145	0.6227	0.6008	0.6189	0.6164	0.6385	0.591	0.6182	0.6106
	CRH	0.5996	0.5949	0.5996	0.5949	0.5996	0.5949	0.5996	0.5949	0.5996	0.5949
	EF(20%)	0.6945	0.6673	0.6693	0.6667	0.6792	0.6712	0.7179	0.7123	0.7188	0.7104
	EF(40%)	0.7996	0.728	0.7081	0.7025	0.7636	0.7397	0.8016	0.7828	0.8116	0.7926
	EF(60%)	0.7867	0.7836	0.7415	0.7241	0.7652	0.7505	0.8216	0.8023	0.8415	0.8285
	EF(80%)	**0.8024**	**0.8043**	**0.7893**	**0.7769**	**0.8063**	**0.8031**	**0.8415**	**0.8285**	**0.8632**	**0.8395**

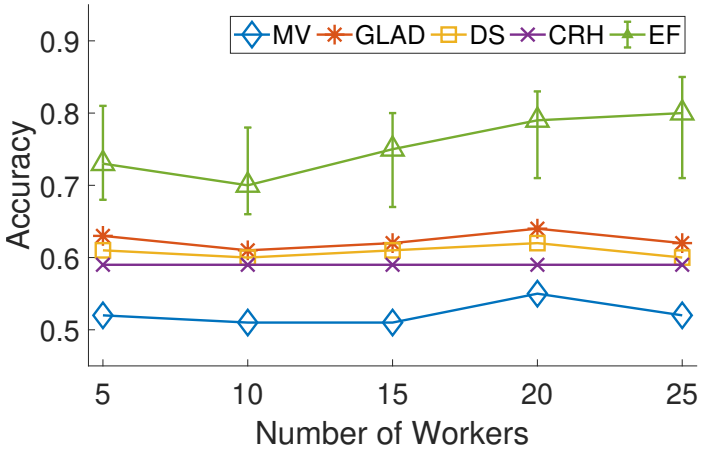
Note: the value in the parentheses after "EF" denotes the coverage rate of Expert in EFusion.

To investigate the stability of EFusion in different scenarios, we simulate some more difficult settings where distributions of α_W and β_j vary. On the basis of original distribution, we increase the variance of α_W, increase the variance of β_j, decrease the mean value of α_W to negative, and increase the mean value of β_j in turn. Since those settings will bring noise and difficulty to the detection of true labels.

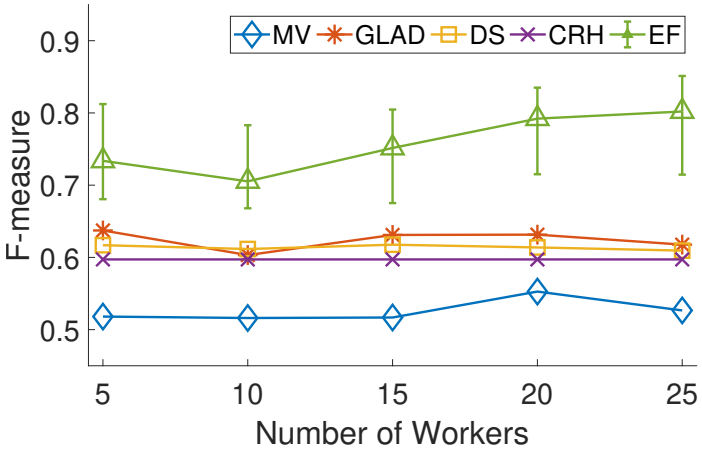
Fig. 3.7 shows the performance of baseline methods and EFusion under different α_W, β_j settings, using 40% coverage rate of Expert. Fig. 3.7a and Fig. 3.7b show the results when the numbers of workers are 5 and 25, respectively. From the figure, it can be observed that EFusion gets higher accuracy than the baseline methods under different settings when the number of workers varies from 5 to 25. The performance of EFusion remains stable even if α_W and/or β_j have a high variance. For instance, EFusion can achieve a high accuracy when the workers' expertise level has a high variance in Setting 1. This is because when workers' expertise varies largely, some workers' expertise is close to Expert. The answers returned from those workers are very close to those answered by Expert. This is equivalent to that we have more "experts" and their answers were implicitly given higher weights in the decision process of EFusion.

3.5 Conclusions

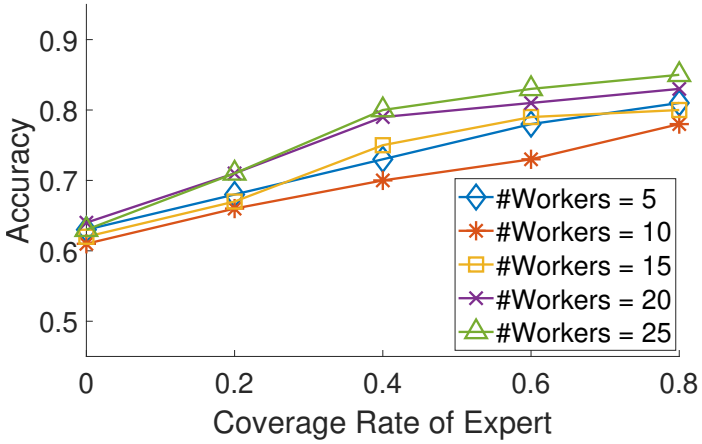
Crowdsensing that uses the information from crowd workers in finding answers from various sensing data has been applied into more and more areas. Observing that the expensive cloud resource can use its higher computation capability to obtain the knowledge from domain expert and return more reliable answers compared to the crowd, we design a model named EFusion, which uses a small amount of computation capability of the cloud to infuse the domain knowledge into crowd's answers, so that the false answers could be corrected *automatically*. Meanwhile, the expertise levels of crowd workers and the difficulty level of questions can also be inferred. The results demonstrate that in addition to the high robustness and good performance in estimating parameters, EFusion outperforms all the baseline methods in terms of accuracy and f-measure.



(a) Accuracy of different methods in the simulation.



(b) F-measure of different methods in the simulation.



(c) Performance comparison of EFusion in the simulation when varying the Expert coverage rate.

Figure 3.5: The performance of different methods in the simulation when $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$, $\alpha_E = 5$.

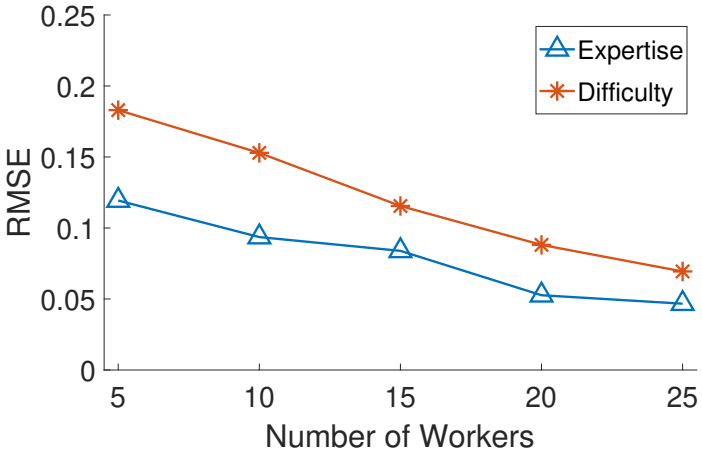
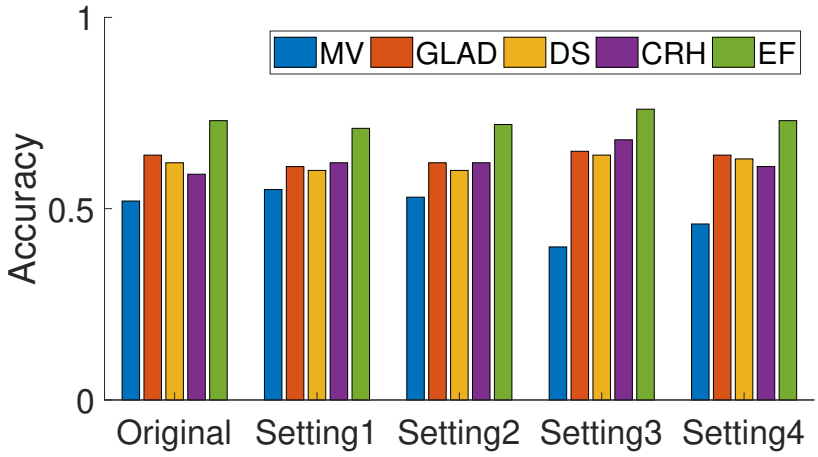
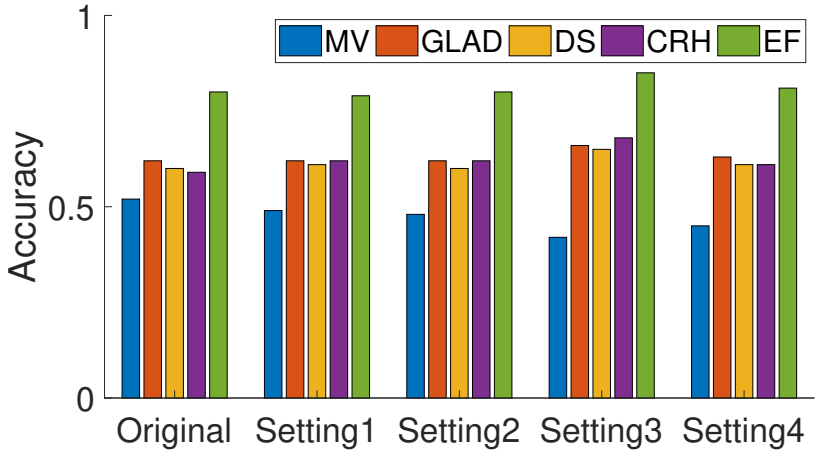


Figure 3.6: RMSE between estimated parameters and the corresponding true values when varying the number of workers under $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$, $\alpha_E = 5$.



(a) Number of workers is 5.



(b) Number of workers is 25.

Figure 3.7: Performance of different methods in various settings:
Original: $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$
Setting 1: $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$
Setting 2: $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 5)$
Setting 3: $\alpha_W \sim \mathcal{N}(\mu = -1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 5, \sigma = 1)$
Setting 4: $\alpha_W \sim \mathcal{N}(\mu = 1, \sigma = 0.2)$, $\beta_j \sim \mathcal{N}(\mu = 10, \sigma = 1)$.

Chapter 4

Interactive Learning Pattern Recognition

4.1 Overview

One of the universal applications of MCS involves asking mobile sensors for real-time sensing data (e.g. photos, noises) of some particular objects or some particular scenarios for further research. One example is environmental monitoring, where the participants are required to take photos of some endangered species or specific objects (e.g. poisonous mushrooms). Such MCS tasks normally involve both mobile users' spatial-temporal information as well as their capability of securing correct, high-quality sensing data. As a result, whether or not the people who are equipped with the mobile sensors can correctly recognize the objects becomes a critical problem, especially in the scenarios where the majority of users do not have the prior knowledge. The cost of this kind of participatory MCS is usually high, because the crowd workers may need to be rewarded, and even if they are purely volunteers, it would require substantial efforts for the decision maker to clean incorrect data if the crowd workers are not competent for the given tasks. In summary, "finding right people for right tasks" has become one of the most fundamental challenges in MCS.

To identify crowds who have sufficient knowledge for a certain MCS task, limited truth-labeled samples are usually used to test candidates or help them getting familiar with the formal MCS task. In this process, different people make different learning progresses due to their background and their capability in learning. To ensure a MCS task is completed with high quality, we need to identify competent workers with limited training samples before assigning them the formal task.

In this chapter, we propose an interactive learning pattern recognition framework: Goldilocks, that can filter users by recognizing their learning patterns based on the behavioral parameters and the question features. To achieve this goal, we adopt adaptive teaching strategy. We evaluate Goldilocks on two challenging real-world datasets. Evaluation results demonstrate that compared to the baseline methods, Goldilocks can improve the accuracy and stability of multi-categories classification problems as well as saving time and transmission cost.

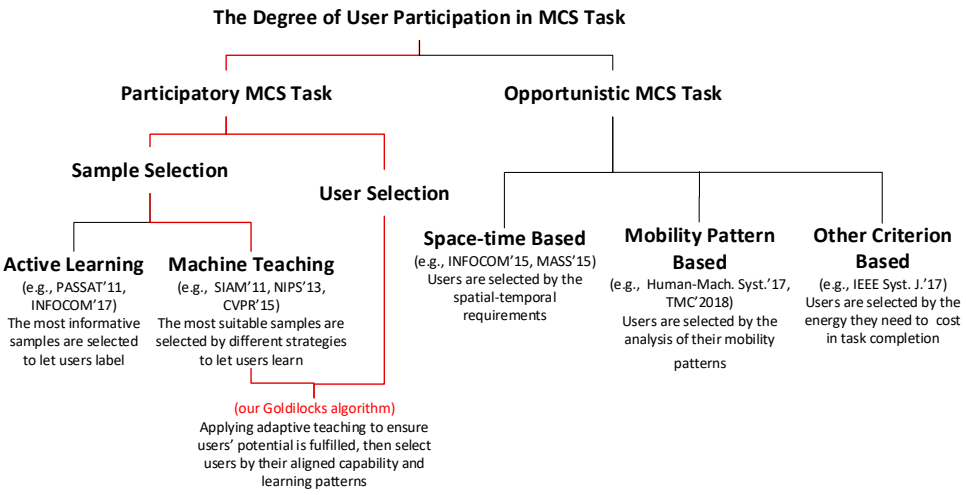


Figure 4.1: Classification of MCS participant-cloud interaction patterns.

4.2 Related Work

As an extended sensing paradigm of the traditional CS, MCS uses the connection between devices equipped with sensors and the cloud-based network to obtain real-time sensing data. As discussed in Section 2, the MCS tasks released by different platforms can be roughly classified into two categories based on the degree of human participation: participatory sensing and opportunistic sensing. From these two perspectives, much research has been done to improve the collected sensing data quality. The taxonomic status of Goldilocks is illustrated in Fig. 4.1.

4.2.1 Data Quality Enhancement for Opportunistic MCS

[46] proposes a recruitment strategy in opportunistic networking to generate the required space-time paths across the network for collecting data from a set of fixed locations. [52] presents a spatial-temporal model for participant recruitment, but it considers a more complex situation where the tasks come in real time and have different spatial-temporal requirements. In [31], the strategy of multitask worker selection is made on the basis of workers' movement patterns and the task's sensitivity to time. The recruitment strategy proposed in [88] is based on probability that a user moves to a destination. In addition to the objective spatial-time position and user's mobility, there is other research on user selection in opportunistic MCS tasks. For example, [54] considers the user's energy needed to complete tasks and uses a participant sampling behavior model in the participant selection. [94] presents a personalized task recommend system which recommends tasks to users based on their preference and reliability to different tasks. However, these works study the optimal worker selection problem without considering user's capability change over time in the task completion process.

4.2.2 Data Quality Enhancement for Participatory MCS

Since participatory MCS requires the active involvement of individuals, active learning and machine teaching are frequently used to interact with users in their labeling process, so the progress of users can be observed in real time.

The authors in [98] introduce a method to automatically identify the most valuable unlabeled instances as well as the samples that might benefit from relabeling. [97] suggests to combine both labeled and unlabeled instances when interacting with users. It proposes a bidirectional active learning algorithm by querying user both the informative unlabeled samples and the unreliable labeled instances simultaneously in a two-way process. The distributed active learning framework in [93] takes the upload and query cost into consideration in each round when it interacts with different users, with the goal of minimizing the prediction errors for classification-based MCS tasks. In addition to the above active learning frameworks, diverse studies in machine teaching also provide methods to deal with the interaction problem in the labeling process with users. [20] presents a teaching strategy to achieve more effective learning, which builds a probabilistic model based on users' answers in each round. [102] employs Bayesian models to find the optimal teaching set for individuals. The model in [44] is an interactive machine teaching algorithm that enables a computer to teach challenging visual concepts to a user by probabilistically modeling the user's ability based on their correct and incorrect answers. However, none of these works consider fine-grained user learning pattern other than their answers in the interactive process.

4.2.3 Adaptive Teaching

Since many MCS tasks require specific domain knowledge, users usually need training before they are assigned tasks. In order to select right people for a given task, we should consider two problems: (a) how to teach people so that they can generalize the learned knowledge as soon as possible? (b) how to profile users' learning pattern so that better candidates for the MCS task can be determined?

Regarding the first problem, adaptive teaching [45] is a proper way to help people learn more effectively by posing training questions adaptively chosen based on their existing performance. In [20], the next teaching image for a user is the one that the user's answer is predicted to be the farthest from the ground truth. An offline Bayesian model is applied to select adaptive teaching samples in [78]. The teaching strategy in [44] chooses a teaching sample that has the greatest reduction on the future error over the rest of unlabeled samples. While the goal of adaptive teaching is to stimulate a user's learning potential as much as possible, each person's learning progress and final abilities may be different. After adaptive teaching, it is more reasonable to eliminate the users whose performance is not suitable for the MCS task.

Regarding the second problem, we need to extract users' learning styles through their training progress. Although learning pattern recognition algorithms are studied before [5, 30], they normally assume that abundant labeling data in a given domain are available and then apply different machine learning classification models on the historical learning data to extract learning patterns. This does not match our scenario where learning profile should be built on the fly as a new question is asked and answered. In addition, note that the learning pattern recognition methods in existing work are more focused on predicting the accuracy of new labels based on whether or not a user's past

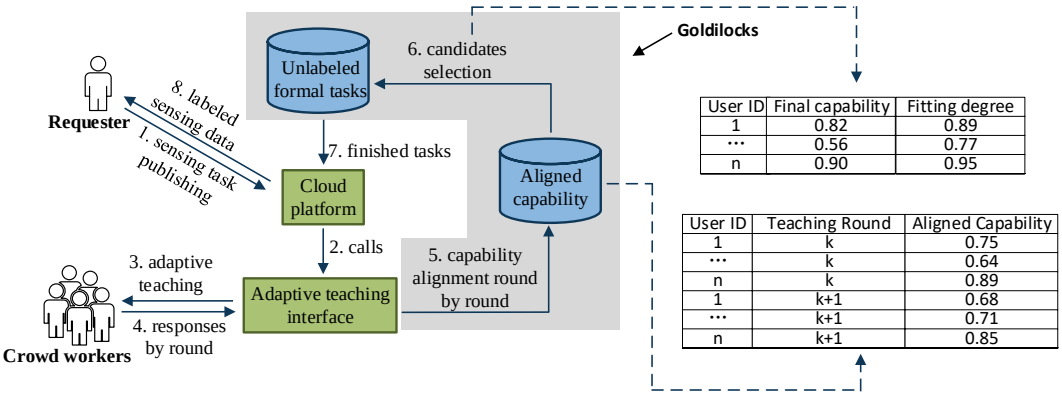


Figure 4.2: The role of Goldilocks in the MCS platform; solid lines denote the sensing data requesting procedure; dash lines denote the intermediate results obtained by Goldilocks; the grey box denotes the main functions of Goldilocks.

questions are labeled correctly. From the traditional research of learning theory [6], the past learning experience can make different influence on different individuals. At the same time, even for the same person, whether a question is labelled right or wrong can bring different stimuli that affect her follow-up study. Therefore, to conduct worker-task matching in participatory MCS, the detailed learning trend of a user has to be captured.

In this thesis, we consider the above two problems and propose an interactive user selection framework for participatory MCS platforms. Note that *we build on the existing solution [44] for adaptive teaching but develop a new method for profiling users' learning patterns.* Unlike other works, we focus on selecting users with adaptive capability acquisition round by round[1] for each worker. To match the right user to right task, we firstly select the most appropriate teaching samples for each user to ensure they can learn as much as possible according to their own learning patterns. Then we select users by fitting their learning pattern to the non-linear (sigmoid) [95] curves.

4.3 Overview of Goldilocks

The role of Goldilocks in the workflow of MCS is illustrated in Fig. 4.2: an organization or individual requests sensing data from the cloud platform (Step 1), then an adaptive teaching interface is called to teach crowd workers according to their performances (Steps 2, 3, 4). Here the "adaptive" means the interface will select the questions that can maximize the user's probability of correctly answering the questions in the future round by round. After each round, a comparison will be taken between the previous estimation of the user based on her learning performance so far and her actual performance in labeling the new question (Step 5). Steps 3, 4, and 5 will *repeat until the specified number of teaching samples are taught* to each user. This number is same to everyone. Then we adopt a two-stage candidate selection strategy based on the acquired performance indicators returned by Step 5 to select the optimized user subset for the formal tasks (Step 6). For users who are not selected,

[1] A round means that the user answers a question and then is told whether her answer is correct or not as well as what the ground truth is.

we will not continue to teach or hire them. Finally, the formal questions without ground truths will be completed by *the selected users* under a fixed budget, and the sensing data from this user subset will be returned to the cloud platform, which will then be provided to the original requester (Steps 7 and 8). The above procedure is denoted by the solid lines in Fig. 4.2.

The core component of Goldilocks is the design of Step 5 and Step 6, denoted in the grey box in Fig. 4.2. They can describe the learning pattern of each user by adjusting a user's estimated capability on the basis of her true performance on the teaching sample in each new round. The dash lines refer to the data stored in the "Round-by-round performance indicator" database, and the high-level indicators of user's learning pattern calculated by the "Round-by-round performance indicator" database.

The two core steps are denoted by the grey box in Fig. 4.2: "capability adjustment" and "candidate selection," which will be introduced in Sections 4.4 and 4.5, respectively. The notations used in this chapter are listed in Table 4.1.

Table 4.1: Summary of Main Notations

Notation	*Description*
i	the question number of labeled questions ($0 < i < j$)
j	the question number of the current question
m	the number of extracted keypoint feature vector of a question
n	the number of questions in the teaching set
$\mathbf{x_{iw}^k}$	the wth ($0 < w < m$) keypoint feature vector of the ith question labeled by user k
$\mathbf{A_i^k}$	the vector set consists of the m keypoint feature vectors of the question i labeled by user k
d_{ij}^k	the value of $j - i$ of user k
s_{ij}^k	the similarity between question i and question j answered by user k
a_i^k	binary value, $a_i^k = 1$ if the ith question is correctly labeled by user k, else $a_i^k = 0$
t_i^k	the time of labeling the ith question by user k
I_i^k	the ID of the ith question labeled by user k
δ_j^k	the offset on user k's capability after the jth question is labeled
α_{ij}^k	the impact of the user k's previously answered question i on her current to be answered question j
ϕ_j^k	the value of $\sum_i \alpha_{ij}^k$
M_j^k	the accumulated capability of user k after the jth question is labeled

4.4 Capability Adjustment in Goldilocks

We adjust the estimation of user's capability round by round. If we measure the user's capability with a *performance indicator*, its value should be adjusted over time, based on the questions that

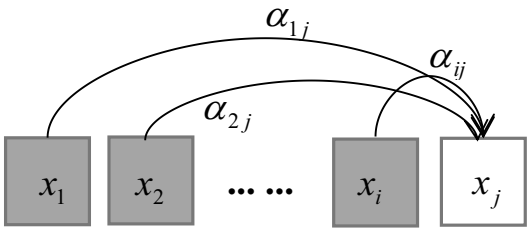


Figure 4.3: The impacts of the previous learned questions on the selection of current question. Note that the shaded parts are samples that have been taught to the user.

the user have answered and the current question at the current round. Using M_j^k to indicate user k's capability estimated after she answers the j-th question, we need to design an algorithm that calculates $\{M_1^k, M_2^k, \cdots, M_n^k\}$, where n is the total number of training questions for the user. We use superscript k to denote user k to emphasize that the sequences of teaching questions for different users are different.

Intuitively, when a user is trained with a question i, the knowledge learned from this question may have an impact on the probability that the user correctly answers a later question. This impact comes from various reasons, e.g., (1) the questions may be similar, and (2) adaptive teaching [45] raises the next question based on the user's historical answers to previous questions. In addition, the impact may become smaller as time goes. As such, we need a method to estimate the impact that all previous questions on the user's performance of answering current question (Section 4.4.1) and consider this impact when we adjust the user's performance indicator value (Section 4.4.2).

4.4.1 Impact of Previous Questions

We describe the teaching set for user k in the teaching phase as $D^k = \{(1, \mathbf{A_1^k}, t_1^k, a_1^k), ..., (n, \mathbf{A_n^k}, t_n^k, a_n^k)\}$, where $\mathbf{A_i^k}$ is an extracted high-dimensional vector set to describe the features of ith question. Note that the similarity between different teaching samples and their order should be used in our model. t_i^k denotes the time that user k spends on labeling question i and a_i^k is a binary variable to record whether her answer is correct ($a_i^k = 1$ if question is correctly labeled, otherwise $a_i^k = 0$).

Although the size of the teaching set is same for all the users, each user actually faces different teaching samples in each round because the adaptive teaching method [45] shows different next question to each user based on her historical answers. As shown in Fig. 4.3, for each user in the teaching phase, each previously answered question has an impact on the current selection of question j to be raised to the user. We denote this influence value as α_{ij}^k.

In the estimation of this sigmoid, we need to consider parameters a_i^k and t_i^k on each previously learned question i. If question i is correctly labeled, a shorter time spent on the question implies a better skill on such type of questions, and thus she is more likely to use the knowledge related to question i when answering the current question j. If question i is given a wrong label, the larger t_i^k implies that user k has worked harder on this question. While the reasons could vary, it is reasonable to assume that a larger t_i^k would have a more positive impact on the user's later

performance compared to the situation that she spends a little time on question i with a wrong answer returned. Based on the above consideration, we suggest the following equation to calculate α_{ij}^k:

$$\alpha_{ij}^k = \frac{1}{1 + e^{-\left[\left(\frac{s_{ij}^k}{d_{ij}^k}\right)\left(1+\frac{(-1)^{a_i^k+1}}{t_i^k}\right)\right]}}, \tag{4.1}$$

where $d_{ij}^k = j - i$ is the distance between questions i and j, s_{ij}^k is the similarity between question i and question j. The calculation of s_{ij}^k depends on specific applications using different vector extraction methods (e.g. image, audio, and text should use different algorithms to calculate similarity). Following the human memory curve [22], a larger d_{ij} implies a smaller likelihood that the user correctly labels question j.

When we estimate the capability of a user before she answers a current question j, we should consider the impact of *all* previous questions. As such, we calculate $\sum_i \alpha_{ij}^k$, which is denoted as ϕ_j^k for simplicity:

$$\phi_j^k = \sum_{i=0}^{j} \alpha_{ij}^k. \tag{4.2}$$

ϕ_j^k could be considered as a quantitative measure of *the impact of previous questions on the ability that the user can correctly answer the current question*, since ϕ_j^k encodes all the impact of previous questions before j. ϕ_j^k also implicitly reflects the knowledge that the user has gained so far and thus is useful for later adjustment of her capability estimation after we see her true answer.

Equation (4.2) has two advantageous properties: (a) It considers the relationship of different teaching questions. And in such context, it utilizes the user's correctness and their time spent on those questions. (b) As the number of samples learned by the user increases, ϕ_j^k is increased by a number in $(0, 1)$ each time. Compared to other user profiling methods, this process enables us to focus on the user's capability of applying the new knowledge she just learned, rather than simply counting the number of correctly answered questions.

Remark. *Equation (4.2) is just one way, among potentially many others, of estimating ϕ_j^k. While there is no theoretical guarantee on its accuracy, it can be empirically shown to be effective in our later experimental studies.*

4.4.2 Adjusting Capability Estimation

After the user answers the current question j, we then use ϕ_j^k and her answer to adjust the estimation of her capability. As mentioned before, we use M_j^k to indicate user k's capability estimated after she answers the j-th question, and we record $M_1^k, M_2^k, \cdots$. Essentially, we need to find the adjustment value δ_j^k to calculate $M_j^k = M_{j-1}^k + \delta_j^k$.

When the user answers the current question correctly

We propose the following formula to calculate δ_j^k when a user answers the current question j correctly (i.e. $a_j^k = 1$):

$$\delta_j^k = \frac{\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(\phi_j^k - \mu)^2}{2\sigma^2}}}{t_j^k} \tag{4.3}$$

The above formula is proposed due to the following considerations: (1) Assuming that ϕ_j^k (i.e., the knowledge gained by a user) follows normal distribution, the numerator is the normal distribution's probability density function (pdf) with mean μ and variance σ^2, where μ and σ are empirical values[2]. (2) Assuming that the longer the time that the user spends on the question, the lower the increments on her capability estimation, we divide the probability value by t_j^k. It is worth noting that in our method it is the shape (i.e., the bell shape when t_j^k is fixed) rather than the absolute values that matter, and as such we believe other possible functions of the similar shape would also work well.

When the user answers the current question incorrectly

In this case, the more likely that she should answer the question correctly from her historical estimation, the higher reduction that should be posed to adjust her capability estimation. Due to this reason, we use a monotone decreasing function w.r.t. ϕ_j^k and t_j^k to obtain δ_j^k:

$$\delta_j^k = \frac{-\nu\left(\phi_j^k\right)^2 + c}{t_j^k} \tag{4.4}$$

where ν and c are parameters set with experimental results[3].

Overall, the 3-D functional image of a_j^k, t_j^k, ϕ_j^k and δ_j^k is shown in Fig. 4.4.

4.4.3 Pseudocode

To summarize, the main idea of user's capability estimation includes: (a) estimating the impact of previous learned questions and (b) based on the estimation and the latest user performance to adjust the current capability estimation. The pseudocode is outlined in Algorithm 1.

In this workflow, EER($\cdot$) (i.e., Expected Error Reduction) is the interactive teaching algorithm [44] that we adopt to determine the next sample which can stimulate user's learning potential in the greatest extend. To be more specific, the next teaching sample is the one that, if labeled correctly, can reduce the probability of total error in the greatest level over the samples that are still not selected into the teaching set. This selection problem is formulated based on the conditional probability function and solved by a graph-based semi-supervised method named Gaussian Random Field (GRF) [103]. $\Psi(\cdot)$ represents the feature detection algorithm to obtain the similarity between any two questions. U($\cdot$) is the impact measure function of how the previous learning questions act on a user k, which have been introduced in Section 4.4.1. $\prod_0(\cdot)$ and $\prod_1(\cdot)$ stand for the different ca-

[2]From our later experimental results, $\mu = 2.69, \sigma = 1$.

[3]In our later experiments, to make Equation (4.4) and Equation (4.3) have the same range values, we set $\nu = 0.011$ and $c = -0.018$.

pability adjustments when the user gives a correct or wrong answer to a new question j, respectively, as discussed in Section 4.4.2.

Algorithm 1 Capability Adjustment Workflow

Input: The number of question n, and the first teaching sample labeled by the kth user $T^k = \{(1, I_1^k, t_1^k, a_1^k)\}$, where I_1^k is the ID of the first question labeled by the kth user. t_1^k is the time the kth user spend on the first question. $a_1^k = 1$ if the first question is correctly labeled by user k, else $a_1^k = 0$

Output: The user's performance indicator after each round and the final accumulated capability indicator of the user $M^k = \{M_1^k, M_2^k, \cdots, M_n^k\}$

$M_1^k \leftarrow 0$
for $j = 2 \rightarrow n$ **do**
  $(j, I_j^k, t_j^k, a_j^k) \leftarrow \text{EER}(j-1, I_{j-1}^k, a_{j-1}^k)$
  $T^k \leftarrow T^k \cup (j, I_j^k, t_j^k, a_j^k)$
  $\alpha_{ij}^k \leftarrow 0$
  for $i = 1 \rightarrow j-1$ **do**
    $s_{ij}^k \leftarrow \Psi(I_i^k, I_j^k)$
    $d_{ij}^k \leftarrow j - i$
    $\alpha_{ij}^k \leftarrow \alpha_{ij}^k + \text{U}(s_{ij}^k, d_{ij}^k, a_i^k, t_i^k)$
  $\delta_j^k \leftarrow 0$
  if $a_j^k = 0$ **then**
    $\delta_j^k \leftarrow \prod_0(\alpha_{ij}^k, t_j^k)$
  else
    $\delta_j^k \leftarrow \prod_1(\alpha_{ij}^k, t_j^k)$
  $M_j^k \leftarrow M_{j-1}^k + \delta_j^k$
return $M^k = \{M_1^k, M_2^k, \cdots, M_n^k\}$

4.5 Candidate Selection in Goldilocks

To accurately select the high-quality users, we divide the selection process into two stages based on the learning curve (drawn by M^k returned by Algorithm 1) of each candidate.

- Firstly, we keep the a percentage ($p\%$) of candidates according to their final capability (i.e., M_n^k) and exclude the others, where p is a tunable parameter. We adopt this step because the final capability is a reflection of a user's final learning result, and thus we need filter out the ones whose final capability is below a certain threshold.

- Next, we fit the learning curve of each remaining candidate with our desired curve. We order these candidates according to the *degree of match* and eliminate the bottom candidates until the desired number of candidates are left.

Regarding the calculation of *degree of match*, after the first step, we only care about the shape of each remaining candidate's learning curve. For this reason, we avoid using a static, fixed curve as the benchmark. Instead, we use the shape of logistic regression (sigmoid) curve[4] $y = \frac{b}{1+ae^{-kx}}$

[4]This function is just one possible candidate, and people can adopt other desired shape here.

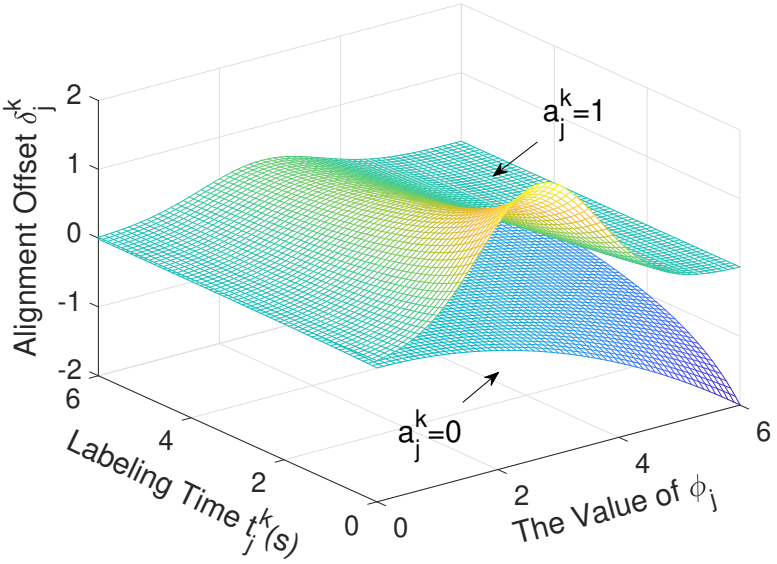


Figure 4.4: The 3-D functional image of a_j^k, t_j^k, ϕ_j^k and δ_j^k, where the upper half represents the relationship of t_j^k, ϕ_j^k and δ_j^k when $a_j^k = 1$, and the other part denotes the relationship of them when $a_j^k = 0$.

(where a, b, k are positive parameters) as the desired shape, and evaluate different measures on the degree of match in Section 4.6.

4.6 Experimental Evaluation

4.6.1 Implementation

To evaluate the performance of Goldilocks, we need real-world data that reflect people's learning patterns. For this, we deploy a Python-based Django [18] website over Amazon Elastic Compute Cloud (Amazon EC2) [23], which interacts with various users and collects the corresponding real-time data. Based on the work [44], we modify its database structure to collect more information includes a user's time spent on each question and the user's exact annotation for each question in teaching and testing phases. The background information of participants such as gender and age is not required in our experiments since uncontrolled participation is one of of the main features of MCS/CS and exactly the problem that we want to solve. In other words, Goldilocks is resilient to participants' backgrounds. If they are identified as qualified based on their performance during interactive training phase, they will be selected. Otherwise, they will be excluded from the final task.

In our experiment, the user will firstly be presented a fixed number of teaching samples one by one. In the teaching phase, the user will be provided the ground truth after she submits the answer of each question, and then our website chooses the next teaching sample to her using the adaptive learning algorithm introduced in [44]. In the testing phase (i.e., formal task), no ground truth will be provided for each question and the questions are selected randomly to each user. The setups of our two experiment datasets are shown in Table 4.2.

To avoid cheating behaviors like participants who are good at a certain task register multiple accounts to join in the same task for multiple times, at first the MCS/CS platform has e-mail and text validation mechanism to make sure each email address/phone number can only register for one account, which effectively increase the cost of multiple registration. Furthermore, we offered a tiny monetary reward of \$0.1 for answering each question in the task, which is unlikely to motivate users to cheat.

Table 4.2: Summary of the experiment datasets

Information \ Dataset	Butterfly	Oriole
#Teaching samples	20	16
#Testing samples	10	8
#Classes	5	4
#Samples per class	300	60

It is worth mentioning that in our experiments, we found that users often answered questions quickly, basically less than 2 seconds. Small time values can lead to a steep increase in the function defined in (4.1) and quick fluctuations on user capability estimated with Equation (4.2). This

problem can be easily avoided by using a "virtual" time system, e.g., using 2000 ms instead of 2 seconds or increasing the real response time by a constant. The purpose of using virtual time is to smooth the fluctuation in a way that we can easily identify a user's learning pattern. Since too-large virtual times (e.g. thousands) will take a heavy toll on using Equation (4.2) to distinguish different learning patterns, in our experiment the virtual time is determined by adding a small constant (2 seconds) to real response time.

4.6.2 Data

- **Question sets:** We select two scientific image sets in which the images are not common in everyday life to evaluate the performance of Goldilocks. The first dataset, called *Butterfly* dataset, consists of a total number of 1500 butterfly images in five categories from a museum collection [44]. The second image set, called *Oriole* dataset, includes 240 oriole images in four different species from a public dataset [86]. Correctly labeling these images requires domain knowledge, which normal users might not have in advance.

- **Data processing:** The scale-invariant feature transform (SIFT) algorithm [57] is an effective algorithm for image feature detection. We apply SIFT to calculate the keypoint feature vectors of each teaching sample, and then acquire the similarity between any two teaching samples by embedding a Heaviside step function:

$$f(\rho) = \sum_{i=1}^{m} \mathbb{1}_{\{\rho_i > \theta\}} \tag{4.5}$$

  where $\mathbb{1}_{\{\rho_i > \theta\}}$ is an indicator function:

$$\mathbb{1}_{\{\rho_i > \theta\}} = \begin{cases} 0 & \text{if } \rho_i > \theta \\ 1 & \text{otherwise} \end{cases}. \tag{4.6}$$

  Here function $\rho_i = \left\| \mathbf{x}^{\mathbf{k}}_{\mathbf{jw}} - \mathbf{x}^{\mathbf{k}}_{\mathbf{j'w}} \right\|_2^2$ is the Euclidean distance between the w-th ($0 < w < m$) keypoint feature vectors of teaching images j and j', and θ is a self-defined similarity threshold between two different keypoint feature vectors.

  We implement the SIFT algorithm in C++. To make the labeling task non-trivial, we set the image blur-adjustment parameter λ of the Gaussian pyramid to 1.1, which introduces some difference to intra-category images and some similarity to inter-category images [57].

- **Data collection:** We share the Python-based Django website with 100 participants, we collect their performance data in the teaching phase and the testing phase, respectively, for both the *Butterfly* and *Oriole* image sets. In this work we collected data from 100 users since (1) we asked at most 35 workers in the formal task after training, and (2) the final performance we expected in the formal task is not very high (only not less than 75% accuracy). Therefore 100 is an appropriate number for our experiment in terms of experiment precision requirement, experiment budget, and experiment timeline.

While we have collected the data for the participants in both the teaching and testing phases, to compare Goldilocks and other baseline methods in terms of the quality of final returned sensing data, we can *pretend* that after the teaching phase, only X number of qualified users out of the 100 participants are selected in the testing phase, where X is a variable determined by the participant selection criteria. In this way, we have the ground truth regarding the data quality with and without those filtered users in the testing phase.

4.6.3 Baseline Algorithms

Goldilocks is designed to optimize the recruitment in participatory MCS/CS, so though the recruitment problem of opportunistic MCS works in Section. 4.2 is related, they are not comparable here because the problems they try to solve are unlikely to involve user's capability, skill, or preference, which can be the main recruitment basis in participatory MCS/CS. For some works in Section. 4.2, their implementation requires correlation among answers from a group of users [97] or the upload and query cost during interactions with different users [93]. However, Goldilocks is aimed to solve the uncontrolled participation problem in MCS/CS without knowing much historical performance and cost in advance.

For the other related works [20,44,102], they are probabilistic methods that deal with the training optimization problem without worker selection. As a result, their outputs are not comparable with the output of Goldilocks. However, we take the method in [44] to optimize the training process in Goldilocks to safely exclude the candidates who still have unsatisfactory performance after the optimized training phase. We select the work in [44] here because the other two works are compared in [44] and the result shows the method proposed in [44] is the state of the art.

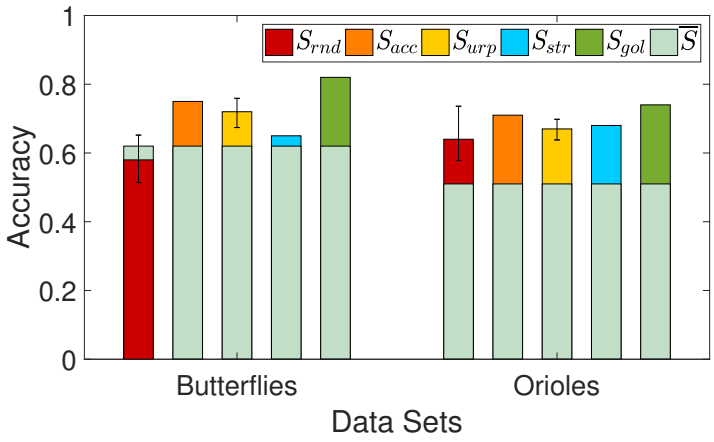
According to the above analysis, we compare our method with the following baseline methods:

- $\mathcal{S}_{acc}$: This method filters the candidates based on their accuracy in the teaching phase. The candidates with more correctly-labeled questions are chosen to participate in the formal tasks.
- $\mathcal{S}_{urp}$: This method [94] does not have an explicit teaching phase but it estimates users' reliability using the following *equivalent* method: After a user labels all the questions, we randomly select a small number of questions, and estimate the user's reliability based on whether or not she has correctly answered the chosen questions. In our experiment, we ask the users to label the 30 images from *Butterfly* dataset and randomly select 5 images to estimate the users' reliability. For the *Oriole* dataset, we ask the users to label the 20 images and randomly select 4 images to estimate the users' reliability. Users are ranked based on their reliability from high to low, and we select candidates based on their reliability.
- $\mathcal{S}_{str}$: STRICT [79] is an optimized algorithm that selects all the teaching samples for a user before the training, based on the use's prior. In other words, the order of teaching samples is computed offline and *fixed* in the user's training process. It finally selects the users who have the best performances on the fixed teaching set.
- $\mathcal{S}_{rnd}$: It randomly selects the required number of candidates without any teaching. To reduce the high variation due to the randomness, we take the average result over 50 random selections in all of our following experiments.

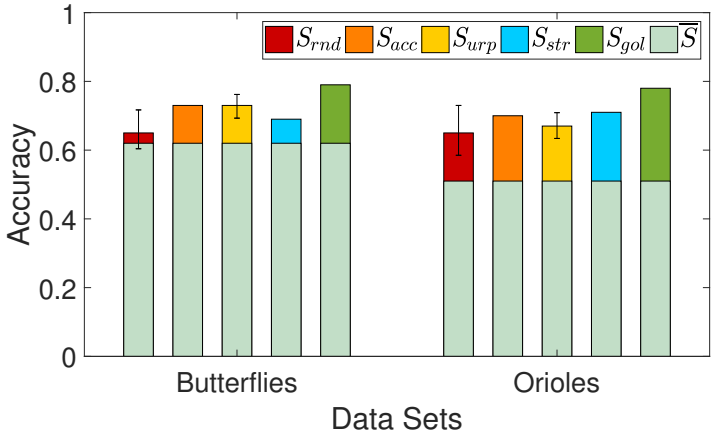
4.6.4 Evaluation Results and Analysis

Accuracy and F-measure

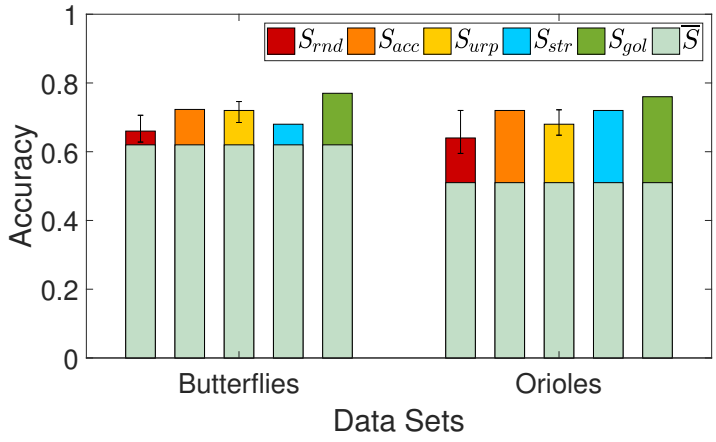
Fig. 4.5 shows the average labeling accuracy by users selected with different participant selection methods in the testing phase. For comparison, we set the number of selected users (i.e., ϵ), after the teaching phase, from 20 to 35 with the increment of 5. From the figure, we can see that on both datasets, all the methods perform better than the random selection. It is notable that when $\epsilon = 20$, the performance of random selected users is even worse than that of all the users. This is because the random selection may choose more unqualified users than qualified ones. We also observe that there is no fixed "second-best" across different methods and among different datasets. However, our Goldilocks achieves the highest accuracy among all the five methods in all the ϵ settings on both datasets.



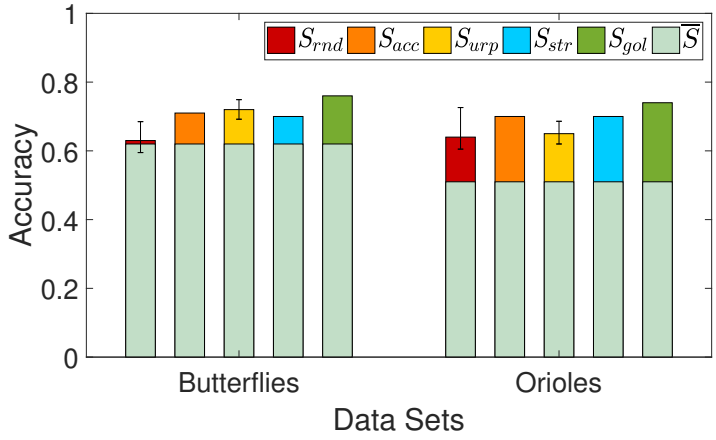
(a) $\epsilon = 20$.



(b) $\epsilon = 25$.



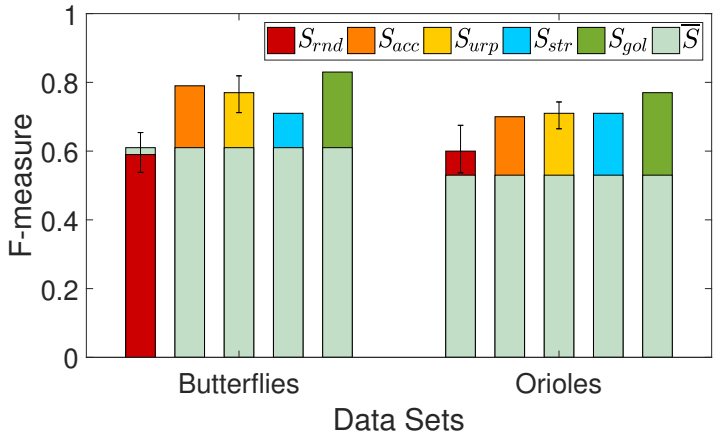
(c) $\epsilon = 30$.



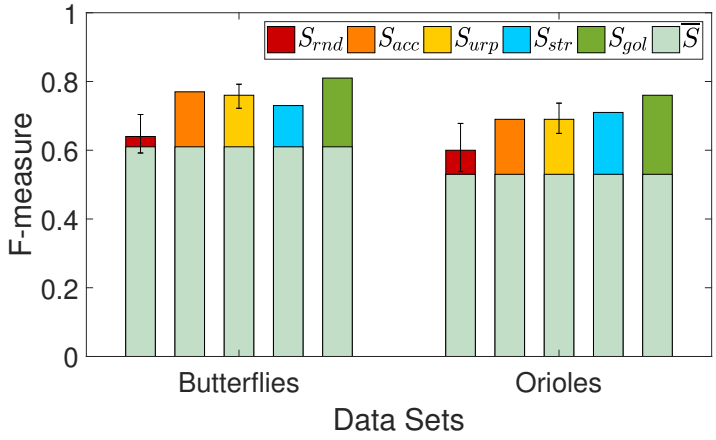
(d) $\epsilon = 35$.

Figure 4.5: Accuracy of different methods on different datasets when the number of selected users for the testing phase is different. $\mathcal{S}$ denotes the performance that all users are selected for the testing phase.

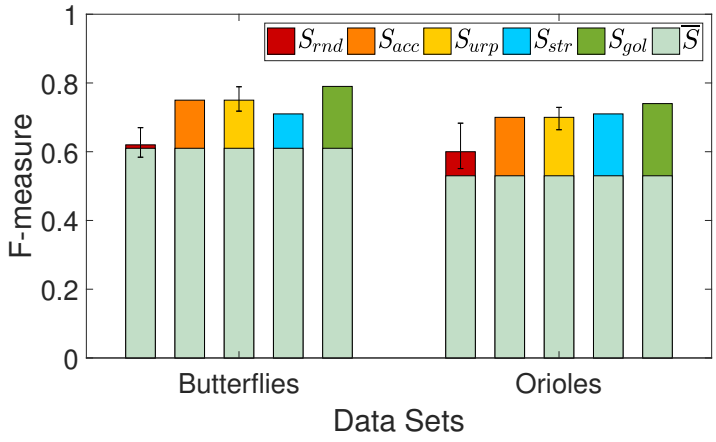
F-measure [72] is a harmonic mean of precise and recall that ranges from 0 to 1. In addition to accuracy, this metric also reflects the stability of a model. Fig. 4.6 shows the f-measure value of different methods under different number of the finally selected users on the two experiment datasets. On both image sets, $\mathcal{S}_{rnd}$ has the worst f-measure performance among all the methods. Meanwhile, $\mathcal{S}_{gol}$ outperforms all the baselines across different numbers of selected users. The maximum f-measure value of Goldilocks can reach 0.83 when $\epsilon = 20$ on the *Butterfly* dataset. Also, no baseline works consistently the second best w.r.t. f-measure. From Fig. 4.6, we can also observe that the f-measure performance is generally poorer on the *Oriole* dataset than on the *Butterfly* dataset. This is because image samples of orioles are more difficult to classify, possibly because the environmental background, as shown in Fig. 4.7, might distract users' attention.



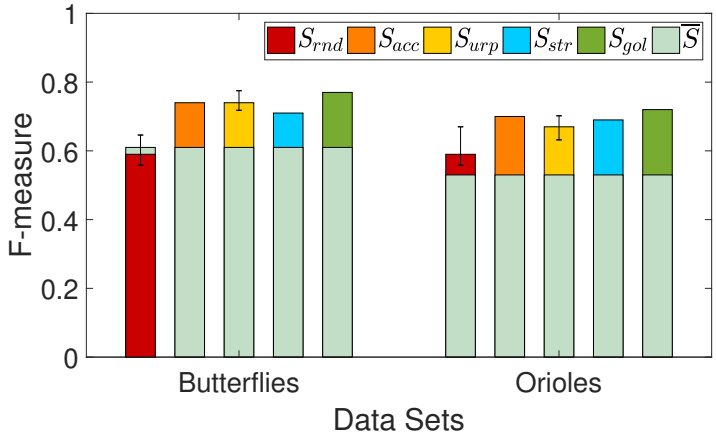
(a) $\epsilon = 20$.



(b) $\epsilon = 25$.



(c) $\epsilon = 30$.



(d) $\epsilon = 35$.

Figure 4.6: F-measure of different methods on different datasets when the number of selected users for the testing phase is different. $\mathcal{S}$ denotes the performance that all users are selected for the testing phase.

Response Time

If a user can answer questions quickly with a high accuracy, the user must have gained good knowledge for the task. We therefore analyze the response time of those users who are chosen for the testing phase. Here the response time is defined as from the time when a selected user is shown the first image to the time when she submits the answer of the last image during the testing phase.

Table 4.3 summarizes the average response time of users selected by different methods on two different datasets. we can observe that the users selected by Goldilocks tend to respond more quickly compared to other baselines. Considering Goldilocks' good performance in accuracy and F-measure, we have enough confidence to conclude that the users selected by Goldilocks have a better grasp of the knowledge. Although it can be seen that $\mathcal{S}_{rnd}$ and $\mathcal{S}_{urp}$ also perform well on the average response time in some ϵ settings, users' poorer performance on the accuracy and f-measure implies

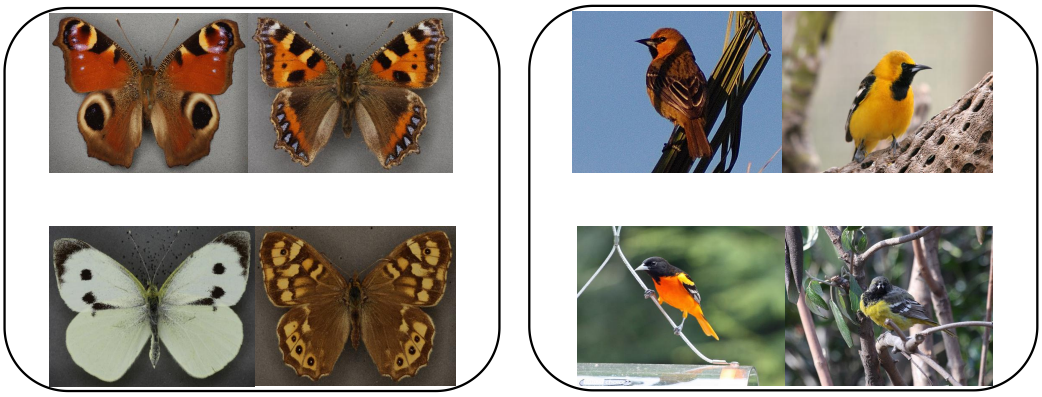


Figure 4.7: Example image samples in our experiments.

that the selected users may not be reliable.

Table 4.3: Average Response Time in the Testing Phase

		$\epsilon = 20$		$\epsilon = 25$		$\epsilon = 30$		$\epsilon = 35$	
		Butterfly	Oriole	Butterfly	Oriole	Butterfly	Oriole	Butterfly	Oriole
Method	$\mathcal{S}_{rnd}$	**41.35**	28.05	**44.32**	28.18	47.50	27.96	47.00	28.41
	$\mathcal{S}_{acc}$	47.60	34.90	46.60	35.44	45.00	34.13	47.26	33.60
	$\mathcal{S}_{urp}$	46.80	28.60	46.36	27.48	43.50	**26.80**	45.26	**26.66**
	$\mathcal{S}_{str}$	49.45	25.69	54.35	27.88	52.30	30.65	48.37	31.89
	$\mathcal{S}_{gol}$	42.96	**21.25**	45.30	**23.12**	**42.53**	27.40	**45.08**	28.03

Note: ϵ denotes the number of selected users for test

Teaching Curve

In Goldilocks, it is important to make sure that each candidate can get appropriate training through the adaptive teaching method in [44]. In this way, the candidates who still have unsatisfactory performance after the teaching phase can be safely excluded for the formal task. Since both Goldilocks and STRICT include teaching phase, we evaluate the effect of their teaching over all the users.

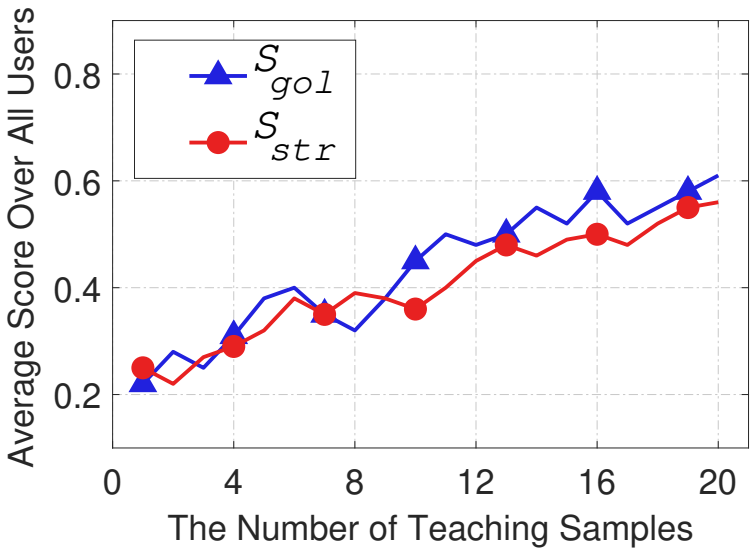
Fig. 4.8 shows the average accuracy over all the users when each new teaching sample is finished during the teaching phase on *Butterfly* dataset and *Oriole* dataset. We can see that for different users, the new released teaching samples are different because adaptive teaching actively selects new teaching sample for each user based on her performance so far.

From Fig. 4.8, we can see that as the teaching process goes on, though the teaching phase of Goldilocks and that of STRICT can both improve users' general performance on different datasets, candidates who are trained with the adaptive teaching strategy in Goldilocks outperform the candidates who are trained by STRICT. In other words, employing the adaptive teaching in Goldilocks can help exclude the unqualified users for the formal task.

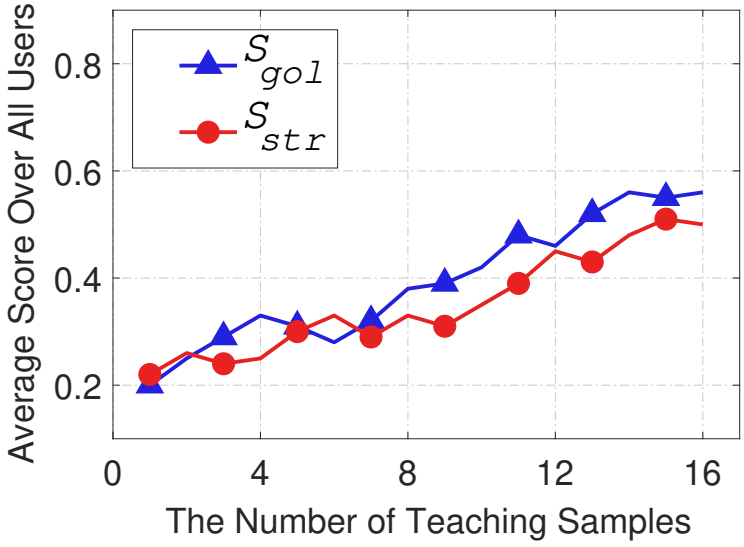
When a task requester requests a task in MCS/CS workflow with Goldilocks, her cost is decided by (1) the size of training set and (2) the size of candidate set and the number of selected candidates for the formal task. Fig. 4.8 shows that more training samples lead to higher average final capability of candidates (even with different learning patterns), while a larger training set will also increase total training time. From Fig. 4.5, we can see that as the number of selected users for the testing phase increases, the average accuracy of selected candidates decreases. To get higher accuracy with more selected candidates, the requester needs to enlarge the candidate set so that more people will have a chance to be well trained. Although this action causes no increment on monetary cost because only selected candidates who finish the formal task will get paid, it causes time cost because more candidates need to finish the training phase. Based on this analysis, the task requester needs to make a balance between their budget (time and monetary) and final data quality based on the feature (e.g. difficulty level, size of training set, etc) of their MCS/CS task.

Fitting Degree of Learning Curve

Goldilocks uses the logistic function introduced in Section 4.5 to benchmark a user's learning curve. Note that a user's learning curve is obtained from the round-by-round capability adjustment (Algo-



(a) *Butterfly* dataset.



(b) *Oriole* dataset.

Figure 4.8: Average accuracy over all the users in the teaching phase on different datasets.

Table 4.4: Fitting Performance of Top 5 Learning Curves and the Corresponding Accuracy in the testing phase on the Butterfly dataset

User	Metrics			Accuracy
	SSE	RMSE	Adj-rsquare	
13	0.0244	0.0379	0.9638	90%
54	0.0621	0.0605	0.9702	90%
55	0.0399	0.0485	**0.9837**	**100%**
62	0.0226	0.0364	0.9721	90%
98	0.0304	0.0423	0.9715	90%

rithm 1). To ease illustration, we normalize the capability values with the maximum possible value. The learning curves of the selected top five users on the two datasets are shown in Fig. 4.9a and Fig. 4.9b, respectively. We can see that all the learning curves have a similar shape of the logistic regression function, indicating the suitability of logistic regression.

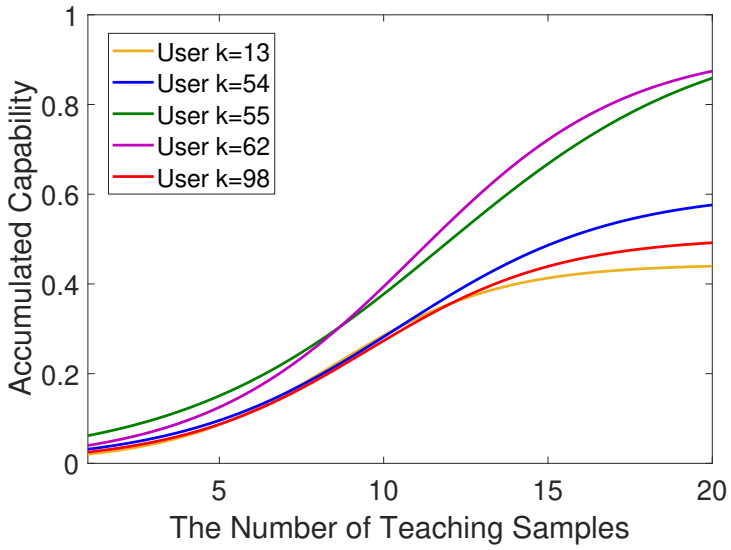
We, however, need a quantitative evaluation on the fitting degree of a learning curve towards the logistic function. Note that we only care about the shape (i.e., the trend) and thus we do not suggest a fixed logistic function. Instead, with logistic regression, different learning curves may fit to different logistic functions (i.e., the parameters in the fitted logistic functions may be different). To evaluate the fitting degree, we test the following metrics:

- *SSE*: The sum of squared errors (SSE) between the learning curve and the fitted logistic function (at discrete range values). The smaller the SSE, the better the fitting.
- *RMSE*: Root Mean Square Error. The smaller the RMSE, the better the fitting.
- *Adj-rsquare*: the adjusted R-square value according to the freedom degree of errors, where R-square is the square of the correlation coefficient between the measured data and the data obtained by the fitting curve. The higher the Adj-rsquare, the better the fitting. .

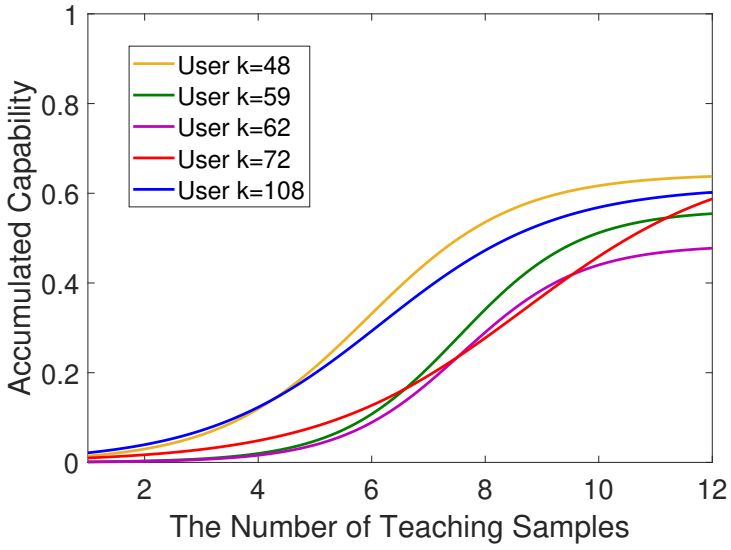
Table 4.4 shows that in the *Butterfly* dataset, both SSE and RMSE are less than 6.5% for all the 5 users' learning curves. And the values of their Adj-rsquare are all beyond 96%. Considering the high accuracy of these 5 users, we can conclude that the users whose learning curve well follow our proposed model have high-quality answers in the testing phase. This is further validated by the fact that the user who has the highest Adj-rsquare (i.e., the 55^{th} user) answers all the questions correctly. Table 4.5 shows the similar phenomena as in Table 4.4. For the *Oriole* dataset, the values of SSE are less than 1.5% and the values of RMSE are less than 4.1% for all the top five users' learning curves. The 62^{th} user has the best Adj-rsquare (99.04%) and the highest accuracy (100%).

4.7 Conclusions

Participant selection has always been a critical problem in MCS. To "select right people for right job", existing methods usually use the number of questions that a user answered correctly during the training phase to judge the user's qualification. We proposed an enhancement method, called



(a) *Butterfly* dataset.



(b) *Oriole* dataset.

Figure 4.9: Fitted logistic regression curves of the top 5 users on different datasets.

Table 4.5: Fitting Performance of Top 5 Learning Curves and the Corresponding Accuracy in the testing phase on the Oriole dataset

User	Metrics			Accuracy
	SSE	RMSE	Adj-rsquare	
48	0.0113	0.0354	0.9869	87.5%
59	0.0146	0.0403	0.9805	87.5%
62	0.0052	0.0240	**0.9904**	**100%**
72	0.0143	0.0399	0.9776	87.5%
108	0.0080	0.0299	0.9887	87.5%

Goldilocks, which (1) estimates users' learning patterns using adaptive teaching and (2) selects users based on the desired learning patterns. Real-world experiments disclose that Goldilocks outperforms existing baselines in terms of efficiency, accuracy, and stability.

Chapter 5

Dynamic Worker Recruitment on MCS/CS Platforms

5.1 Overview

Recruiting qualified crowd workers is critical for any crowdsourcing platform. A widely-used method is to first train candidates with a small sample set and then select a subset of candidates to complete the formal task based on their performance in the training phase. Although many methods have been developed to make use of candidates' performance in the training, no method simultaneously considers individual's learning patterns and dynamically trains promising candidates with suitable training samples. To advance the state-of-the-art in crowd worker recruitment, we propose a deep learning framework, HybrTraining, to dynamically identify the learning patterns of different candidates, based on which the optimized final recruitment decision can be made. HybrTraining utilizes two distinct deep neural networks, long short-term memory (LSTM) and Deep Q-Network (DQN), respectively, to integrate candidates' learning patterns and suitable match between worker and training samples during the training stage. It effectively balances the requirement of high-quality crowd workers and the training time, and trains qualified candidates for the final crowdsourcing task. Experiments with two real-world datasets demonstrate the superior training and recruitment performance of HybrTraining in terms of effectiveness, efficiency, and stability. Meanwhile, HybrTraining shows a degree of transferability across different cases when input parameters change.

5.2 Related Work and Background

5.2.1 Recruitment in CS/MCS

In recent years, CS/MCS has been used broadly to harness public wisdom with optimized costs, efficiency, and scalability. To ensure the quality of crowdsourcing outcomes, a common method is to train crowd workers before assigning them tasks. As introduced in Section 1.3.3, static training [28, 58] and dynamic training [27, 44, 47] strategies have been adopted. As a result, a human-centric

crowdsourcing recruitment system needs to answer a critical question: How to recruit and evaluate users effectively based on the information from the training process? Different methods have been proposed to solve this problem.

To evaluate users, a commonly-used method is to build the profile of workers based on their performance in the training phase, and then assign tasks according to the degree of matching between different tasks and the workers. In [84], users are trained to decipher scanned words from books that optical character recognition (OCR) systems failed to recognize. The system learns each user's reliability based on the correctness on a question set with known ground truth. The work [11] estimates the trustworthiness of a worker by characterizing the work in three dimensions: score, trustworthiness, and status. If the trustworthiness value of the worker is lower than a threshold, this worker will be excluded from task assignments. Although the method considers the interaction between workers and the crowdsourcing system, the purpose of such interaction is only to identify qualified workers from a candidate group, instead of training workers through the interaction. For tasks that require specific domain knowledge (e.g., the task in [84]), in absence of training, such methods would be slow to recruit qualified workers. Therefore, a more personalized and training-oriented system is needed so that workers can be trained and evaluated appropriately.

Some other methods are proposed to identify suitable crowd workers. Pu et al. [73] designed an optimal worker recruitment policy with dynamic programming in mobile crowdsourcing. Two DQNs are applied in [76] to solve the task arrangement problem in crowdsourcing platforms. However, the crowdsourcing tasks in those works are opportunistic [51], where users' qualification (or quality) is usually judged by their locations and computing resources in opportunistic sensing. In this chapter, we design a training and recruitment framework that targets participatory crowdsourcing tasks.

5.2.2 Why Deep Learning?

Deep learning has great success in many real-world applications such as natural language processing (NLP) [96], recommendation systems [13], and computer vision [85]. One main advantage of deep learning is that it can *automatically* capture *latent* features in complex systems. In the context of crowd worker recruitment, deep learning allows us to model the learning patterns and match candidates and training samples based on the latent features.

Long short-term memory (LSTM)

LSTM is a type of recurrent neural network (RNN) architecture [36]. Typically, each LSTM neuron has four inputs to decide the final output of this neuron (i.e., LSTM unit). The four inputs are: the original input z, an input gate that controls if z could be written into the memory cell, an output gate that decides if the value of the current neuron is accessible to other neurons, and a forget gate that decides when to forget the content in the memory cell. Suppose z is the output of the last neuron, z_i, z_o, and z_f are the inputs of the input gate, output gate, and forget gate, respectively. The new value to be saved into the current memory cell is:

$$c' = g(z)f(z_i) + cf(z_f), \tag{5.1}$$

where c is the original content saved in the memory cell, $g(\cdot)$ is the activation function of z. Activation function $f(\cdot)$ usually adopts a sigmoid function [33] that outputs a value between 0 to 1 to decide how much the corresponding gate will be opened. The final output of each LSTM unit is thus:

$$a = h(c')f(z_o), \tag{5.2}$$

where $h(\cdot)$ is the activation function of c'.

LSTM has been recently used in learning pattern recognition. Zhou et al. [101] train an LSTM model for a *collection* of learners to predict their learning trajectory and performance. However, this model could not be applied in crowdsourcing scenarios that ask for more personalized training because it is a clustering-based method for learner-question matching. A cognitive structure enhanced framework is proposed in [55] to sequentially identify the right learning samples to different learners, but it assumes that all participants have the same understanding of the same training sample. While works in [55, 101] cannot be directly applied in our context, they motivated us to develop an event-based, individualized LSTM model for learning pattern recognition.

Deep Q-Network (DQN)

Deep Q-Network is the combination of a model-free reinforcement learning method named Q-learning and a deep neural network. A Markov decision process (MDP) is defined as a tuple $< S, A, P, R >$ where:

- S is a set of states,
- A is a set of actions,
- $P_a(s, s')$ is the probability of taking action a in state s at time t, and transiting to state s' at time $t + 1$,
- $R_a(s, s')$ is the immediate reward of taking action a and transiting from state s to state s'.

Under this definition, at each time step t, given $s_t \in S$, an agent takes action $a_t \in A$ and obtains a reward r_t, and this process transits to the new state s_{t+1}.

The goal of Q-learning is to learn a Q-function $Q : S \times A \to \mathbb{R}$ that characterizes the cumulative discounted reward by taking an action a in state s following an optimal policy. To learn the Q-function, in each step t, the loss function is defined as:

$$L(\theta) = Q(s_t, a_t) - \gamma(r_t + \max_a \widehat{Q}(s_{t+1}, a)), \tag{5.3}$$

where $\widehat{Q}$ denotes a target network that provides the target value for the Q-function to learn. The parameters of $\widehat{Q}$ will be iteratively updated. $\gamma \in (0, 1)$ is a discount factor that decides the importance of the reward of each step. Compared to tabular Q-learning, DQN estimates Q-values using a deep neural network, which will be applied to train the final Q-function.

DQN has great successes in computer games [32, 67] and natural language processing [35]. It has also been successfully applied in the recommendation system recently. [100] proposes a DQN-based dynamic framework for online personalized news recommendation. A page-wise recommendation

framework based on DQN is proposed in [99] to optimize a page of items with proper display based on real-time feedback from users. Since the problem of matching the training samples to the most appropriate candidates during the interactive training process is inherently a recommendation problem, we explore the use of DQN in our application setting.

5.3 Overview of HybrTraining

The role of HybrTraining in the crowdsourcing workflow is illustrated in Fig. 5.1. HybrTraining (marked in the grey box) is integrated into the standard crowdsourcing workflow as follows:

- **Step 1**: A requester posts crowdsourcing tasks and provides a training set T to crowdsourcing platform.
- **Step 2**: Candidates who want to participate in the tasks should go through training first. Candidates' profiles, if any, are accessible to the crowdsourcing platform. We use n to denote the total number of candidates during the training phase.
- **Step 3**: Assign training batches. For each episode j, a fixed number of training samples are randomly selected from T to form a training batch. We use f_{b_j} to denote the features of training batch b_j. A feature extractor on the crowdsourcing platform extracts and concatenates the features of each worker $f_{w_i}(i = 1, 2, \ldots, n)$ and the features of training batch f_{b_j} to build an input to DQN. Based on the input, DQN calculates n Q-values corresponding to the result of assigning b_j to the n candidates. Instead of assigning the training batch to all the candidates, DQN selects the top $k(k < n)$ candidates with the highest k Q-values, and assigns b_j to them. At the same time, b_j can be assigned to k random candidates instead of the top k candidates with small probability. This exploration step is designed to avoid local optimum so that the appropriate matching between candidates and the training batch can be found. We will introduce the exploration strategy in detail in Section 5.4.5.
- **Step 4**: Collect training feedback. The k selected candidates will be trained with the ground truth answer of each question after she submits her answers[1]. Upon receiving the feedback, a DQN model calculates the reward of this episode and updates its internal states based on batch learning tuples sampled from a memory buffer. In each episode, the inputs to the LSTM model include the feature and the ordered feedback of the selected candidates, as well as the feature and the true label of each question in the training batch. The LSTM model then returns the updated features of the k selected candidates, and the worker list is updated accordingly.
- **Step 5**: Based on the latest candidate feature of each candidate at the end of training and the feature of the formal task, the DQN calculates the Q-value for each candidate and selects the top X candidates for the formal task, i.e., the candidates with top X Q-values, where X is a variable determined by the participant selection criteria. Note that a candidate with a higher Q-value means a better match with the formal task.

[1] If they skip some questions, the skipped questions would be labelled as the wrongly answered ones.

- **Step 6**: The selected crowd workers proceed to do the formal task and get paid (or credit) after task completion.

In the next two sections, we introduce the design and training of DQN and LSTM in HybrTraining, respectively.

5.4 DQN-based Worker-Sample Matching (DQN-WSM)

The goal of DQN is to find the best match between candidates and the training batch in each episode. We introduce the structure and the training of the proposed DQN in this section. First, we extract the feature of a training batch and obtain the feature of each candidate through the crowdsourcing platform. Second, The feature of each candidate is concatenated with the training batch's feature as the input of DQN. Next, we define the actions and the rewards of DQN based on the feedback of the selected candidates. Finally, the state transition of DQN and the procedure of updating the DQN parameters based on the tuples in its memory buffer are introduced.

5.4.1 Feature and State Construction

- **Worker feature** f_{w_i}: Different candidates have different expertise levels and learning patterns during the training phase. The goal of worker-sample matching is to assign each training batch to the right candidates in each episode to achieve efficient, personalized, and dynamic training. Here "right" means in selecting candidates, both the correctness of their answers to b_j and past learning progresses reflected from their past answers denote the latest learning pattern of candidate i by feature f_{w_i}. We train an LSTM, which is capable of learning long-term dependencies, to extract f_{w_i} (refer to Section 5.5 for more detail).

- **Batch feature** f_{b_j}: The extraction of f_{b_j} is task-dependent. For example, for image labeling tasks, one can adopt CNNs. Note that the feature of training batch b_j is defined as the average value of the features of all training samples in the batch.

- **State construction**: We concatenate the candidate feature of worker i f_{w_i} and the feature of training batch f_{b_j}, which form one row of the DQN input state. In other words, the input state of DQN in the j-*th* episode is:

$$s_j = \begin{Bmatrix} f_{w_1} & f_{b_j} \\ f_{w_2} & f_{b_j} \\ \cdots & \cdots \\ f_{w_n} & f_{b_j} \end{Bmatrix} \tag{5.4}$$

  In each episode, we use DQN to decide the subset of candidates to participate the training of this episode.

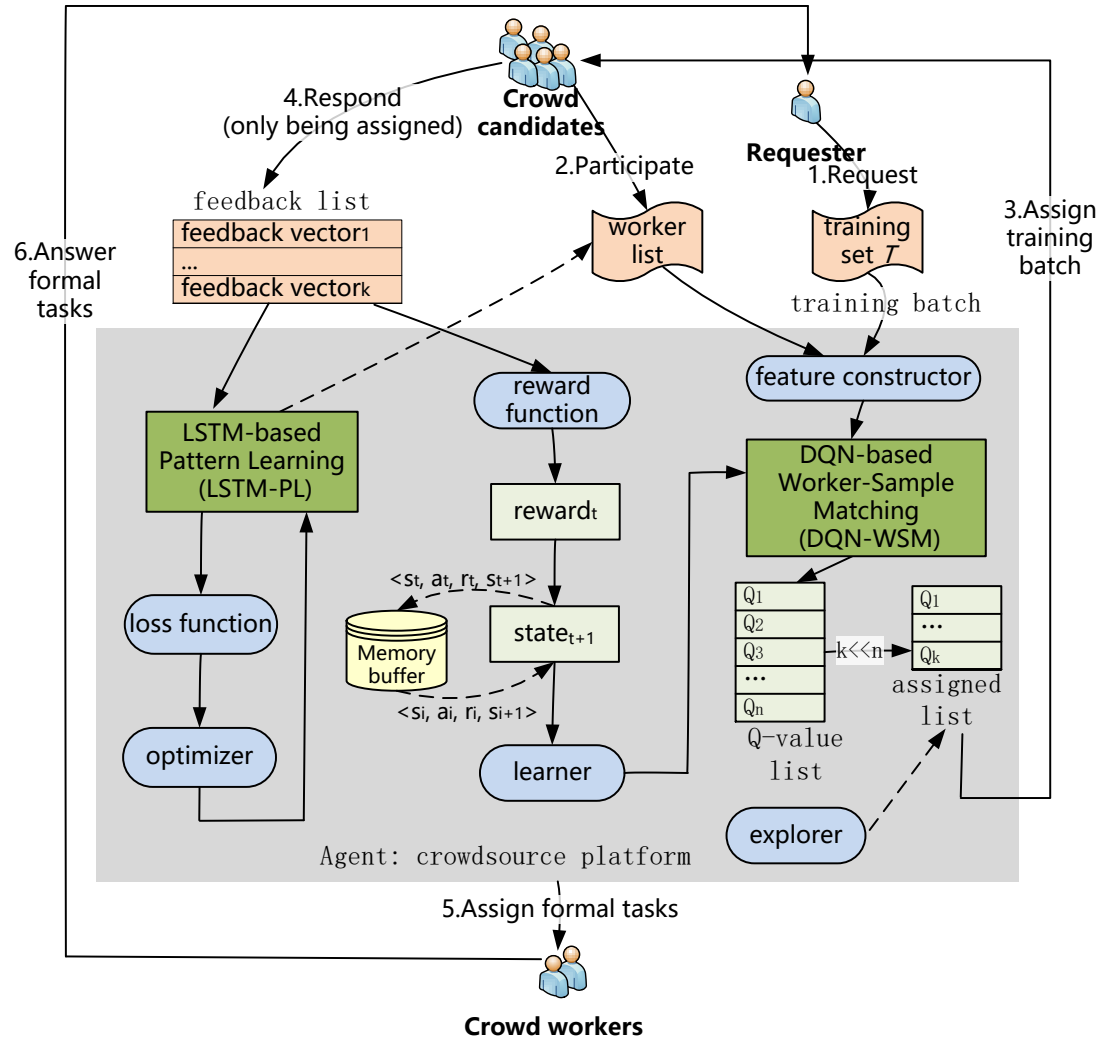


Figure 5.1: The role of HybrTraining in the crowdsourcing platform; solid arrows denote the processing flow; dash arrows denote the data flow; the grey box denotes the internal structure of HybrTraining.

5.4.2 The Structure of Q network

After constructing the input state, a deep network is built to predict the Q values of input state and action pairs. We design the deep network following the structure in [50], as shown in Fig. 5.2. We firstly construct an $n \times (n_{f_{w_i}} + n_{f_{b_j}})$ matrix[2] in (5.4). Next, multiple attention-based layers are applied to project the n rows of the input state into n different Q-values. After each attention layer, there is a feed-forward sublayer that computes the row-wise projection using a softmax activate function:

$$FF(\boldsymbol{h}) = softmax(W^{ff}\boldsymbol{h} + \boldsymbol{b}^{ff}), \tag{5.5}$$

where $\boldsymbol{h}$ is the output from the current sub attention layer, W^{ff} and $\boldsymbol{b}^{ff}$ are the learnable parameters of the current feed-forward layer. At last, the $\boldsymbol{h}$ of the last feed-forward layer will be input to a fully-connected layer to get the final row-wise projected Q value of each candidate. We apply softmax activation in the final fully-connected layer.

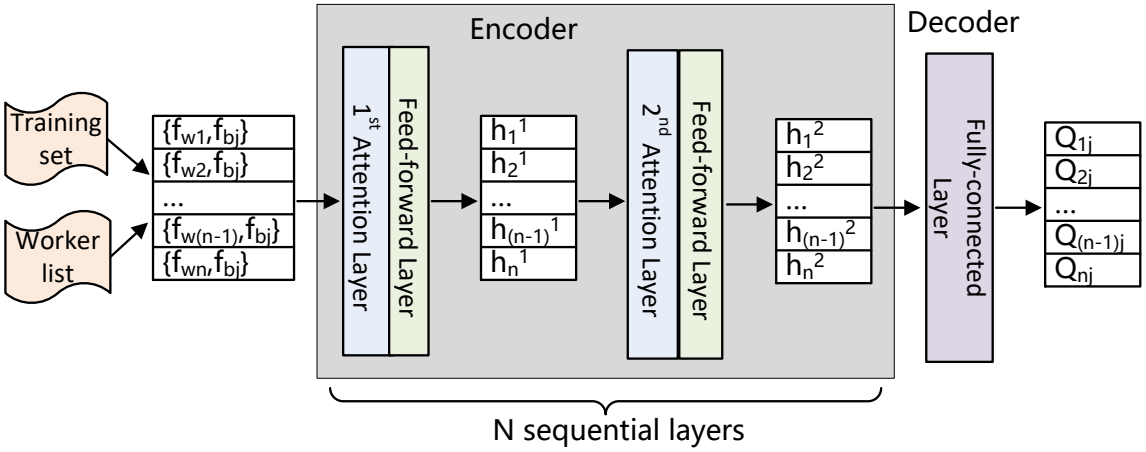


Figure 5.2: The structure of Q network; h_i^k is the node(row)-wise output value of node i after the kth attention layer and the correspond feed-forward sub layer.

An illustration of the attention layer is shown in Fig. 5.3. In the figure, only the information sent and received by the first node (i.e., the first row of input state matrix) is shown for clarity. Clearly, the weights of the information from other nodes to $node_1$ depend on the query of $node_1$ (i.e., q_1) and key value k_is of all other nodes. For each $node_i$, we have:

$$\boldsymbol{q_i} = W^Q\boldsymbol{h_i}, \qquad \boldsymbol{k_i} = W^K\boldsymbol{h_i}, \qquad \boldsymbol{v_i} = W^v\boldsymbol{h_i} \tag{5.6}$$

where W^Q, W^K, and W^V are the learnable matrix parameters of the attention layer. The compatibility of $node_j$ to $node_i$'s query u_{ij} is computed as:

$$u_{ij} = \frac{\boldsymbol{q_i}^T\boldsymbol{k_j}}{\sqrt{d_k}}, \tag{5.7}$$

[2]we use n to denote the total number of candidates, $n_{f_{w_i}}$ to denote the dimension of vector f_{w_i}, and $n_{f_{b_i}}$ to denote the dimension of vector f_{b_i}.

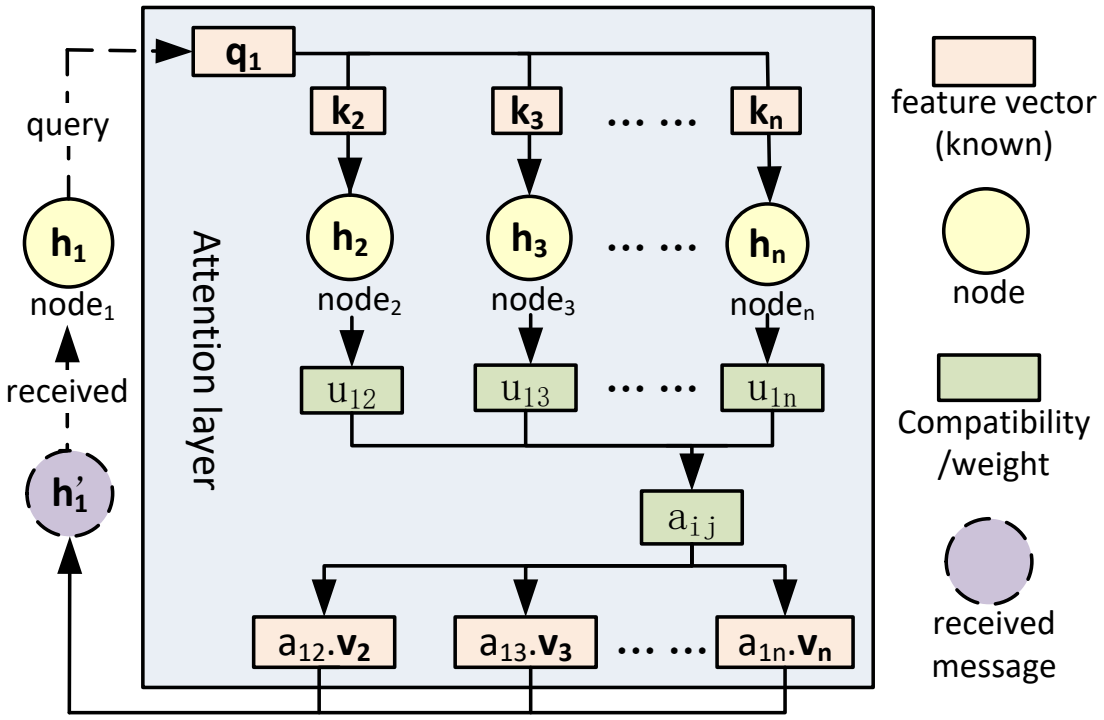


Figure 5.3: One single-head attention layer; For clarity, only the operation on $node_1$ is illustrated in the figure.

where d_k is the dimension of the key value. From u_{ij}, we compute the attention weight of $node_j$ to $node_i$ (i.e, (a_{ij}, $a_{ij} \in [0, 1]$)) using a softmax function:

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}} \tag{5.8}$$

Finally, the information received by $node_i$ from all other nodes is:

$$\boldsymbol{h_i'} = \sum_j a_{ij} \boldsymbol{v_j}. \tag{5.9}$$

It is worth noting that each node will repeat the same operation in each attention layer to get its final Q value in Fig. 5.2.

In general, having multiple heads [83] in each attention layer allows each node to get more comprehensive information from the other nodes so that the parameters of the whole Q network could be learned more efficiently. Given M headers in each attention layer, the final $\boldsymbol{h_i'}$ received by $node_i$ is defined as the liner combination of the outputs from all the M headers:

$$MHA_i = \sum_{m=1}^{M} W_m^O \boldsymbol{h_{im}'} \tag{5.10}$$

where $m \in 1 \ldots M$, $\boldsymbol{h_{im}'}$ is the final information received by $node_i$ by the m-th header, and W_m^O is a learnable parameter matrix that projects back the result of each node after one multi-head attention

layer into a single vector of length h. We set $M = 4$, i.e., four attention heads for each attention layer.

5.4.3 Action and Reward

In each episode j, action a_j is to assign a training batch b_j to k candidates among n candidates ($k < n$). From an input state, DQN returns n Q-values. We sort these Q-values in descending order and select the top k candidates to assign b_j in this episode. Additionally, we also apply an ϵ-greedy policy in deciding whether or not replacing the selected top k candidates by k random candidates when we decide the final action of each episode. The explorer will be introduced in Section 5.4.5.

In episode j, the average correctness of all the selected candidates on b_j is considered as the reward of action a_j. For each candidate assigned to b_j in episode j, her feedback is saved in a vector of the same size as b_j in the feedback list. In this feedback vector, the corresponding element is 1 if a training sample is correctly answered and 0 otherwise.

5.4.4 Next State, Memory Buffer, and Learner

- **Next State**: For each selected candidate in episode j, her latest learning pattern is learned by an LSTM network based on her latest feedback. The binary feedback to each training sample in b_j will be fed sequentially into the LSTM network, and her feature f_{w_i} will be updated according to the hidden state of the LSTM's last layer. Note that for unselected candidates in episode j, their features remain unchanged in this episode. The next state of DQN is then defined as:

$$s_j^{t+1} = \begin{Bmatrix} f_{w_1}^{t+1} & f_{b_j} \\ \cdots & \cdots \\ f_{w_k}^{t+1} & f_{b_j} \\ \cdots & \cdots \\ f_{w_n}^{t+1} & f_{b_j} \end{Bmatrix} \tag{5.11}$$

  following the structure in (5.4).

- **Memory Buffer**: After each episode, we store the new experience (i.e., tuple $< s_j, a_i, r_i, s_j^{t+1} >$) in a memory buffer. If the memory buffer is full, old experiences will be dropped automatically. The size of the memory buffer for DQN is set to 2500 (experiences) in this thesis.

- **Learner**: We sample a batch of experiences from the memory buffer and update the parameters of the DQN by iteratively using these sampled experiences. In addition to the loss function defined in Equation (5.3), we apply double DQN [34] to avoid over-estimating the Q values. In the double DQN, two Q networks are applied, one (Q) for deciding the action of the current time step, and the other ($\widehat{Q}$) for calculating the real Q-value of this selected action. Therefore, the new loss function is:

$$L(\theta) = Q(s_t, a_t) - \gamma(r_t + \widehat{Q}(s_{t+1}, \arg\max_a Q(s_{t+1}, a)). \tag{5.12}$$

5.4.5 Explorer

At the beginning of training, the recruitment system may not have enough knowledge about each candidate. To ensure a training chance for all candidates, we introduce an exploration mechanism in our DQN using the well-known ϵ-greedy [63] method:

At the beginning of training, the recruitment system may not have sufficient knowledge about each candidate. To ensure training opportunities for all candidates, we introduce an exploration mechanism in the DQN using the well-known ϵ-greedy [63] method:

$$a_j = \begin{cases} \arg\max_a Q(a, s), & 1 - \epsilon \\ random, & otherwise \end{cases} \tag{5.13}$$

which means in episode j, the current training batch b_j is assigned to candidates on the list of top k Q-values with probability $1 - \epsilon$ and to k random candidates with probability ϵ.

5.5 LSTM-BASED PATTERN LEARNING (LSTM-PL)

The purpose of the LSTM network is to extract the learning pattern of each candidate so that the DQN could make a better matching decision in new episodes. We adopt LSTM because it can learn the valuable information of a long input sequence better than simple RNNs. In this section, we firstly present the general structure of the LSTM model. Then, we describe its input, hidden state, final output, and learner.

5.5.1 Structure of LSTM

We consider a many-to-one LSTM, which takes a sequence as input and outputs a single vector. Suppose that l samples are randomly selected from the training set to form the training batch in episode j, $\boldsymbol{f_{bj}} = [f^1_{b_j}, f^2_{b_j}, \cdots, f^l_{b_j}]$ denotes the feature of training samples in b_j, and $\boldsymbol{a_{ij}} = [a^1_{ij}, a^2_{ij}, \cdots, a^l_{ij}]$ denotes worker i's sequential feedback on questions in b_j. As shown in Fig. 5.4, $\boldsymbol{f_{bj}}$ and $\boldsymbol{a_{ij}}$ will be fed into the LSTM as two sequences, for each worker i who is assigned the training batch b_j in the j-th episode. Starting with the latest feature of worker i (i.e., f_{w_i}), at each time step k, the LSTM updates all parameters based on the loss at this time step and then produces a hidden state h^k. At the k-th time step, the loss is defined as the cross entropy between the predicted distribution of worker i's answer to b^k_j by the LSTM (we use $\mathcal{D}^k_{ij}$ to denote the predicted distribution of worker i's response to b^k_j) and the worker's actual response to b^k_j (i.e., a^k_{ij}), which has the same dimension as $\mathcal{D}^k_{ij}$.

At the next time step $k+1$, the input includes both $f^{k+1}_{b_j}$ and h^k from the previous step. Then the LSTM parameters are updated by the loss at time step $k+1$, i.e., the cross entropy between $\mathcal{D}^{k+1}_{ij}$ and a^{k+1}_{ij}. For each selected worker, this process is repeated at each time step until all samples in b_j are processed. Instead of considering the hidden states of all l time steps, we take the hidden state h^l, which is the output of the last LSTM unit, as the updated feature of worker i (i.e., $f_{w_i} = h^l$). The worker list is then updated by the new f_{w_i} so that in the next episode, DQN can do better matching with the updated pattern of each candidate.

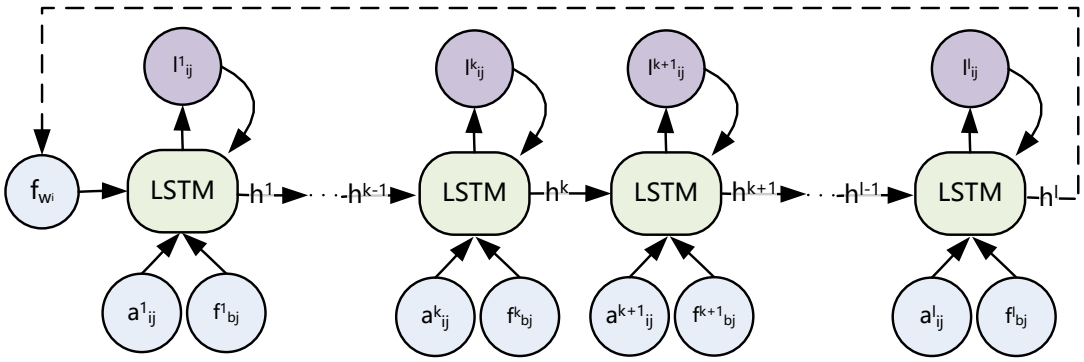


Figure 5.4: The structure of LSTM in HybrTraining. Blue circles denote the input; purple circles denote the loss at different time steps; green boxes denote the LSTM unit at different time steps; dash line denotes the updating flow; solid lines denote the processing flow.

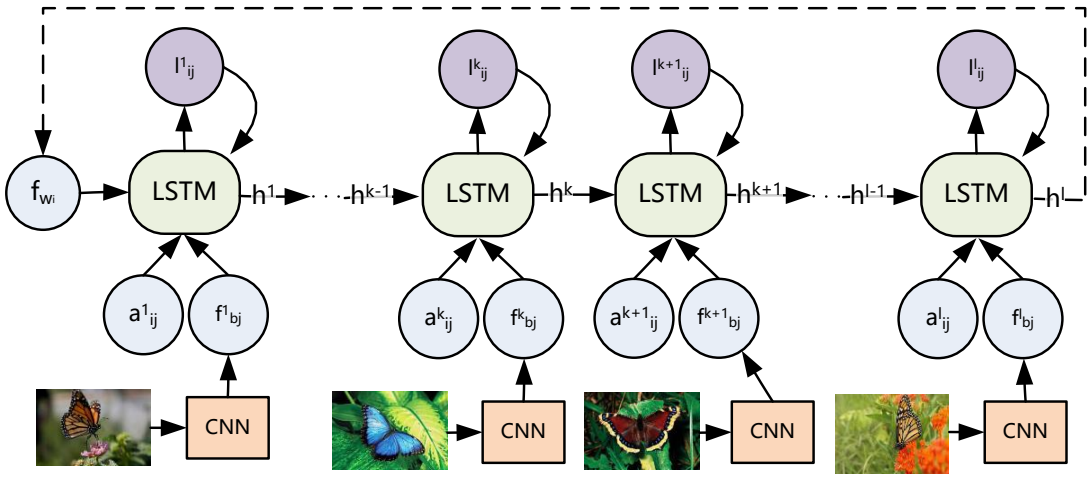


Figure 5.5: Applying CNN in the workflow of LSTM network. blue circles denote the input; purple circles denote the loss at different time steps; green boxes denote the LSTM unit at different time steps; orange squares denote the well-trained CNN; the dash line denotes the updating flow; solid lines denote the processing flow.

5.5.2 Input of LSTM

- ***Worker feature*** The worker feature f_{w_i} is the same as introduced in Section 5.4.1. f_{w_i} is saved as a vector in the worker's profile. Note that LSTM updates f_{w_i} in each iteration. As the input of DQN, f_{w_i} remains unchanged in each episode and is used for training batch assignment in this episode. For a new user whose learning pattern is unknown, we take the average feature value of existing candidates as the value of her initial feature.

- ***Features of a training sample*** The features are application-specific. For example, in the image classification crowdsourcing task, an image is taken as a training sample at each time step. Different convolutional neural network (CNN) models can be employed to extract the features of each image. The role of CNN in the LSTM workflow is shown is Fig. 5.5.

- ***Actual feedback*** The feedback to candidates' responses is used to calculate the loss of LSTM. After each time step, the LSTM parameters are updated based on the loss value.

5.5.3 Learner and Hidden State

Learner: For each selected candidate i in episode j, at time step k, an LSTM is optimized iteratively by minimizing the cross entropy between distribution $\mathcal{D}_{ij}^k$ and a_{ij}^k with a loss function defined as:

$$L(\mathcal{D}_{ij}^k, a_{ij}^k) = -\sum_c \mathcal{D}_{ij}^k(c) \ln(a_{ij}^k(c)) \tag{5.14}$$

where c is the number of classes in classification problems, $\mathcal{D}_{ij}^k$ denotes the probability distribution of worker i's answer for sample b_j^k predicted by LSTM at time step k, and a_{ij}^k is the actual response of worker i for sample b_j^k.

Hidden state: We train one LSTM per user to learn the worker feature from her learning process. Fed with a candidate's feature and the features of training samples in a training batch, the updated LSTM will produce a hidden state h at each time step. The four gates introduced in Section 5.2 cooperatively decide h's value. This hidden state h will then be a part of LSTM inputs at the next time step. Finally, h from the last time step will be used as the latest learning pattern of the candidate. Since the order of training samples is sensitive in the training of the LSTM model, the order of training samples in a training batch should be fixed when the batch is assigned to candidates.

5.6 Experimental Evaluation

In this section, we train the deep learning model and evaluate the performance of different CS/participatory MCS recruitment methods on two real-world datasets. In addition, we show that the knowledge of LSTM and DQN in HybrTraining are transferable when the datasets or candidates change.

5.6.1 Data and Implementation

Data

We evaluate the performance of HybrTraining on two scientific image datasets for image labeling tasks. Correctly labeling these images requires domain knowledge or previous experience on similar tasks, which normal users might not have in advance. The first dataset, called *"Butterfly"*, comes from a museum collection [44]. 320 butterfly color images (of size $480 * 320$) in four categories are used as the training set. Crowdsource workers are asked to label 32 different butterfly images from this dataset to test the performance of different algorithms. The second image dataset, called *"Kuzushiji"*, includes Kuzushiji (cursive Japanese) images [12]. 1280 Kuzushiji grayscale images (of size $28 * 28$) spanning 10 classes are used for training. Workers are asked to label 64 images from this dataset in testing.

To evaluate the performance of HybrTraining, we collect real learning data from 100 people for each dataset using a Python-based Django website over Amazon Elastic Compute Cloud (Amazon EC2). It is worth mentioning that we train LSTMs in HybrTraining for recognizing the learning patterns of candidates rather than classifying images. The learning history of each selected candidate in each episode forms the training set of the LSTM. The setups of the two experiment datasets are shown in Table 5.1.

Table 5.1: Summary of the experimental datasets

	$X = 5$		$X = 10$		$X = 15$		$X = 20$		$X = 25$	
	Butterfly	Kuzushiji	Butterfly	Kuzushiji	Butterfly	Kuzushiji	Butterfly	Kuzushiji	Butterfly	Kuzushiji
# of training images	320	1280	320	1280	320	1280	320	1280	320	1280
# of testing images	32	64	32	64	32	64	32	64	32	64
LSTM training set size	2500	2500	5000	5000	7500	7500	10000	10000	12500	12500
# of classes	4	10	4	10	4	10	4	10	4	10
Image size	$3 * 480 * 320$	$1 * 28 * 28$	$3 * 480 * 320$	$1 * 28 * 28$	$3 * 480 * 320$	$1 * 28 * 28$	$3 * 480 * 320$	$1 * 28 * 28$	$3 * 480 * 320$	$1 * 28 * 28$

Note: X denotes the number of selected candidates in each episode

Implementation

The experiments were performed on a VM with 16 vCPUs and memory of 64 GB launched on Compute Canada [10] (Graham cluster). We create a virtual environment with torch (1.6.0), scipy (1.4.1), numpy (1.18.4) installed in python 3.6.3 on Compute Canada to deal with the reinforcement learning process in HybrTraining.

To extract the features of images in "Butterfly", we apply a *pre-trained* Resnet18 [80] model to encode each image of the shape $(3 * 480 * 320)$ (note that 3 indicates RGB colors) to a $1 * 512$ vector. For the "Kuzushiji", the feature of each image forms a $1 * 784$ vector. Although for both datasets, 100 candidates are asked to label the images in both the training and the testing sets to evaluate HybrTraining and baseline methods, feedback to their labels is assumed to be known only for X candidates selected to be trained in each episode (X is a variable determined by the participant selection criteria). After the training phase, only X candidates would be selected to finish the formal tasks. For "Butterfly", there are 10 learning episodes in each epoch, and 50 epochs are run for each training configuration. For "Kuzushiji", there are 20 learning episodes in each epoch, and 25 epochs are run for each training configuration. After each episode, the running time of this episode is recorded and we employ SummaryWriter from tensorboardX (2.1) to draw the value of DQN reward and the training loss of both DQN and LSTM.

For DQN, a 2-layer double DQN [34] is implemented, with the output sizes of the two layers being 128 and 64, respectively. Two neural networks, main net and target net, are constructed. The parameters in the main net are updated in real-time in response to the environment, while the target net copies the parameters from the main net every 100 episodes. The size of memory buffer is 2500 experiences (refer to Section 5.4.4). We set the learning rate of DQN to 0.001 for both datasets. The size of each training batch is 32 with the "Butterfly" dataset and 64 with the "Kuzushiji" dataset. For ϵ-greedy based exploration, ϵ is set to 0.9 and gradually reduced to 0.05. In LSTM, the output size of each hidden layer and the last layer are 64 and 4 for "Butterfly", respectively, and 64 and 10 for "Kuzushiji", respectively. The candidate features are of dimension $64 * 1$. The learning rate of the LSTM is 0.001.

5.6.2 Baseline Methods

The goal of recruitment algorithm is to find the best match between tasks and crowd workers, as well as giving the crowd workers effective training when the area-specific knowledge is required to finish the tasks. For comparison, we implemented 4 baseline algorithms, including random selection, static algorithm, and two dynamic algorithms.

- **Random**: It randomly selects the required number of candidates without any training. The results presented are the average over 50 random selections.
- **Static Majority (SM)**: It selects the candidates based on their accuracy in the training phase. There are no multiple training batches in SM, and all candidates have to finish the whole training set. The top candidates with the most correctly labeled questions are chosen to participate in the formal tasks.

- **Reinforcement Learning-based Greedy (RLG)**: This is also DQN-based method. Unlike HybrTraining, in this method the accuracy of the selected candidates on the training batch is used directly to update their features. After the training phase, the trained DQN selects the required number of candidates to participate in the formal tasks. For a fair comparison, we use the same DQN structure in RLG as in HybrTraining, and let the DQN learn 500 episodes in both HybrTraining and RLG.

- **LiquidCrowd Algorithm(LCA)**: This algorithm is adapted from [11]. In LCA, we equally divide all training samples into training batches (10 batches for "Butterfly" and 20 batches for "Kuzushiji"), and assign the training samples to all the candidates batch by batch. After the i-th batch, we update the worthiness τ_W^i of each candidate by:

$$\tau_W^{i+1} = \omega_h \tau_W^i + (1 - \omega_h)\alpha, \tag{5.15}$$

  where ω_h is a history weight and $\tau_W^i \in [0, 1]$ is the historical trustworthiness of candidate W calculated by considering the "level of agreement" from other candidates on her historical answers. Award $\alpha \in [0, 1]$ is defined as candidate W's "completion rate"[3]. After all batches are finished, we select the top X candidates (based on the latest worthiness values) for the formal tasks. Since ω_h in Eq (5.15) is a weight for past estimations, we considered both *short-memory* LCA (LCA-S, $\omega_h = 0.2$) and *long-memory* LCA (LCA-L, $\omega_h = 0.8$) in the experiments.

5.6.3 Results and Analysis

Training reward

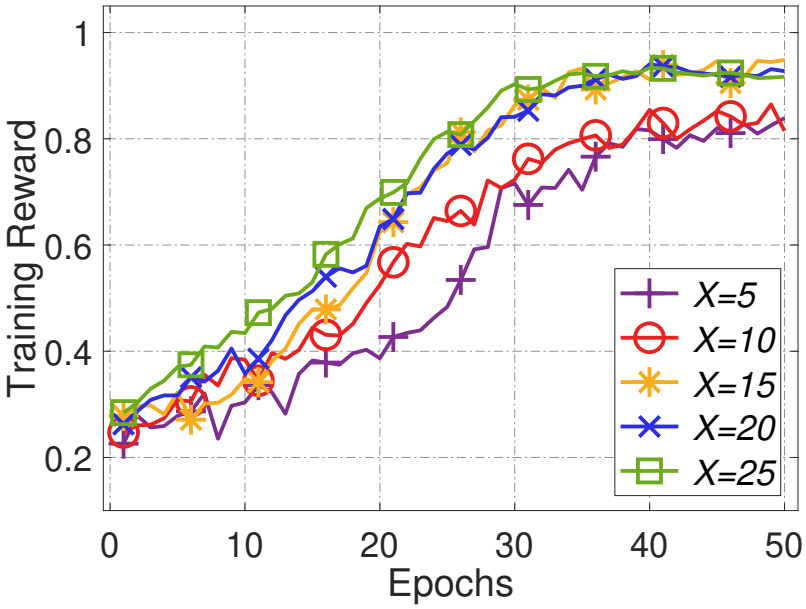
We compare the training reward of HybrTraining and RLG because both are based on reinforcement learning, while other baseline methods are not.

Fig. 5.6 and Fig. 5.7 show the average reward of Q-learning on "Butterfly" and "Kuzushiji" under different X values, respectively. In each episode, DQN selects X candidates from 100 candidates and asks the X candidates to learn from one training batch, and then updates their features based on this process.

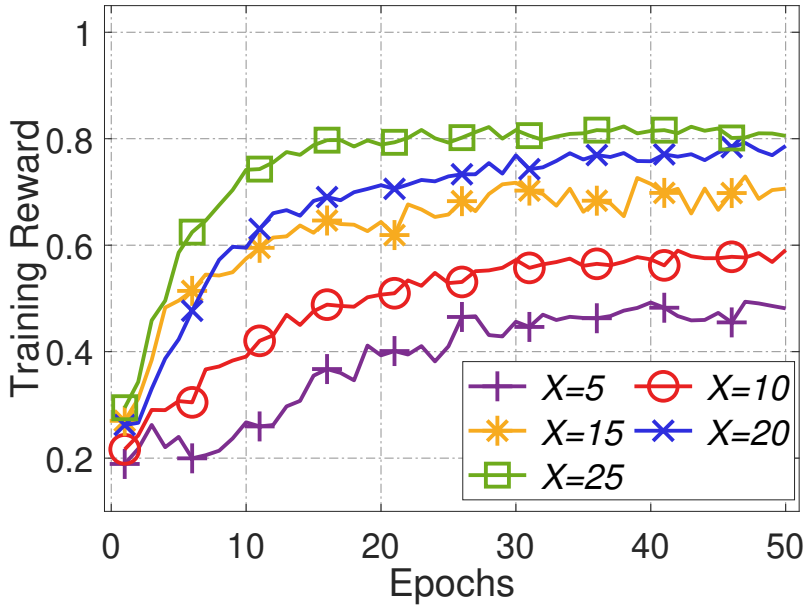
It can be seen from Fig. 5.6 and Fig. 5.7 that compared to RLG, HybrTraining can achieve higher rewards during the training with both datasets. Furthermore, HybrTraining has small reward gaps among different configurations on both datasets, indicating that HybrTraining is more robust. In addition, as the value of X increases, both algorithms perform better because larger X provides more data for DQN to learn in each episode.

The reason HybrTraining needs more epochs to reach convergence on the "Butterfly" is that the LSTM model needs sufficient learning data from candidates to capture their learning patterns. This trend is not pronounced on "Kuzushiji" because different candidates do not exhibit significant difference in learning "Kuzushiji" characters. As a result, HybrTraining can learn the learning pattern of different candidates in fewer epochs.

[3]We only consider the training samples finished in 2 to 30 seconds as the effectively completed ones when we implement LCA. The reason is that the candidate can barely learn anything in 2 seconds, and a long completion time (more than 30) for one training sample may indicate cheating/distraction behaviour during the learning process.

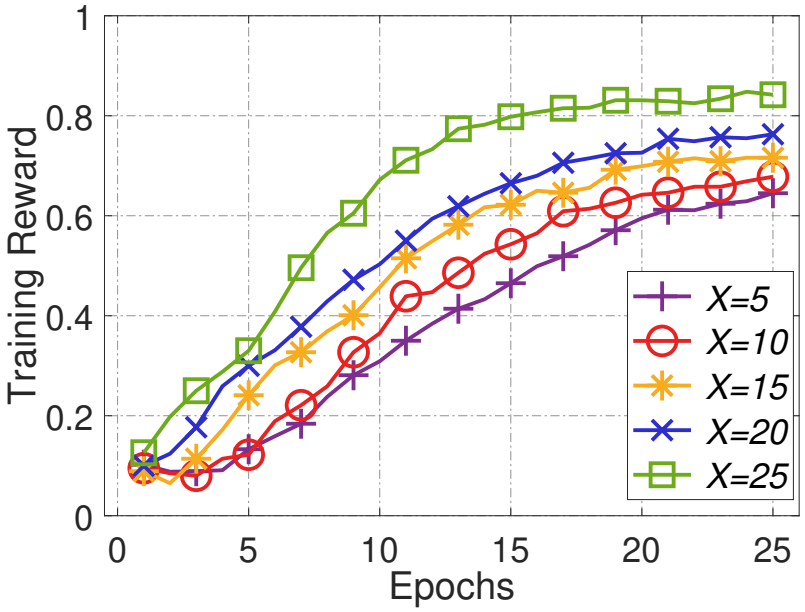


(a) HybrTraining.

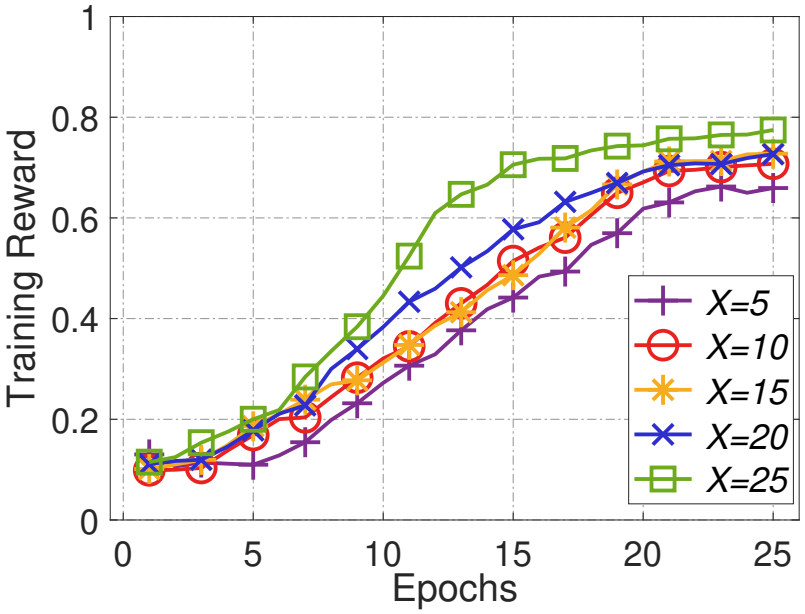


(b) RLG.

Figure 5.6: Average reward of Q-learning in each epoch on "Butterfly" dataset. Experiments are run for different X, which denotes the number of selected candidates in each episode. We run 50 epochs under each X, with each epoch including 10 episodes.



(a) HybrTraining.



(b) RLG.

Figure 5.7: Average reward of Q-learning in each epoch on "Kuzushiji" dataset. Experiments are run for different X, which denotes the number of selected candidates in each episode. We run 25 epochs under each X, with each epoch including 20 episodes.

Accuracy

Accuracy is defined as the percentage of images that are correctly labelled by the selected candidates during the test phase (i.e., the formal tasks). It can be seen from Fig. 5.8 that the candidates selected by HybrTraining outperform those selected by all other baseline algorithms. Although the performance of SM is stable on both datasets, the performance of selected candidates by SM is not as good as those by reinforcement learning-based methods. This is because SM cannot dynamically identify the potential candidates whose learning pattern and learning capacity match the formal tasks well. Moreover, LCA-L performs better than LCA-S because the former has a long memory.

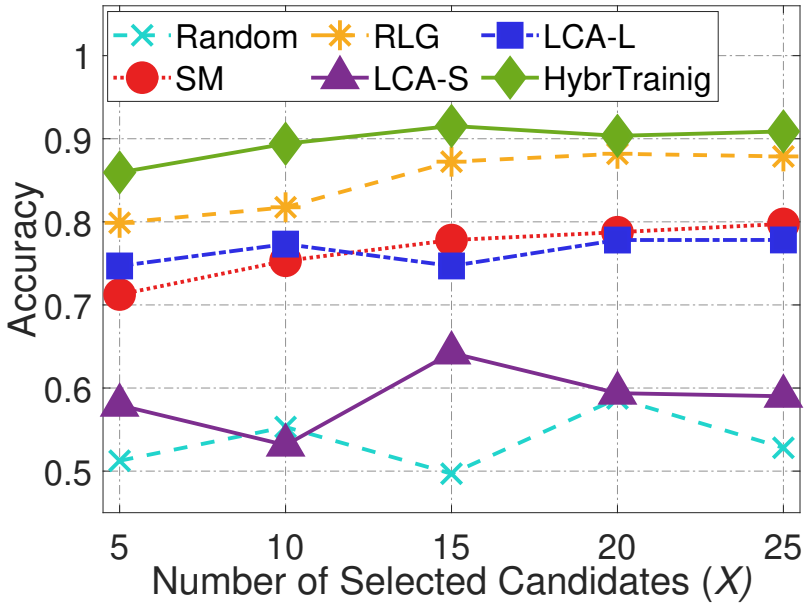
The efficiency of an candidate selection method is another significant indicator to evaluate its performance, because shorter selection time implies smaller cost and faster recruitment. Here *selection efficiency* is defined as the time duration from the beginning of the training phase to the time when X candidates are selected for the formal tasks. The Python-based Django website in Section 5.6.1 is used to collect the response time of all candidates when they label each question in the training phase. Since RLG and HybrTraining are reinforcement learning-based algorithms, only a subset of the candidates are asked to label all the samples in the training phase. Therefore, for each question in each episode in these two algorithms, we take the average response time of all the X selected candidates as the response time of this episode.

Fig. 5.9 shows the selection efficiency of different recruitment methods. For all the algorithms, we take the average result over 20 times for each configuration (i.e., X value). The maximum and minimum selection times of different methods for each configuration are shown with error bars. The error bars of Random and SM are omitted because the maximum and minimum selection times of these two algorithms are very close.

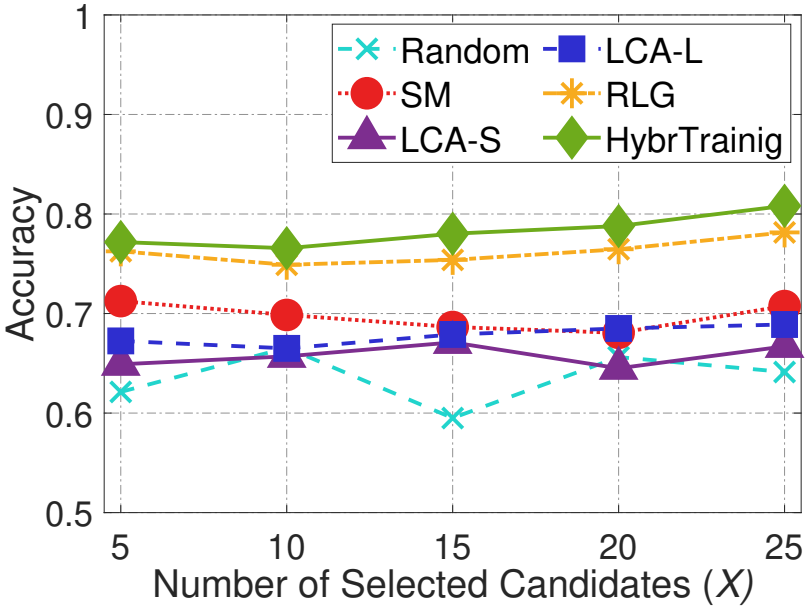
We can observe from Fig. 5.9 that on both datasets, HybrTraining and RLG take shorter selection times than the other baseline methods under different X values. Although Random performs similarly to HybrTraining and RLG when $X = 20$ and $X = 25$, its poorer performance on accuracy implies that the candidates selected by Random are not reliable. As X increases, the selection times of HybrTraining and RLG increase. This is because both methods are learning-based methods. In each episode, the more candidates trained, the more learning data processed. In contrast, the other baseline algorithms ask all the candidates to answer all the training samples and process the collected answers to make the final decision, regardless of the number of crowd workers selected for the formal tasks.

Visualizing the Training Dynamics of HybrTraining

To better understand how HybrTraining finally selects better candidates, we use heat-maps to visualize HybrTraining during its learning process. We set $X = 25$ and randomly select one candidate from those who are eventually selected for the formal tasks (1st candidate) and one candidate among those who are not selected after the training phase (2nd candidate). Fig. 5.10 shows the frequency of assigning training batches to the two candidates in each episode for every 5 epochs, where each epoch contains 10 episodes of learning in “Butterfly”. Similarly, Fig. 5.11 presents the frequency of assigning training batches to the two candidates in each episode for every 5 epochs in “Kuzushiji”. Each epoch contains 20 episodes of learning in the “Kuzushiji”.

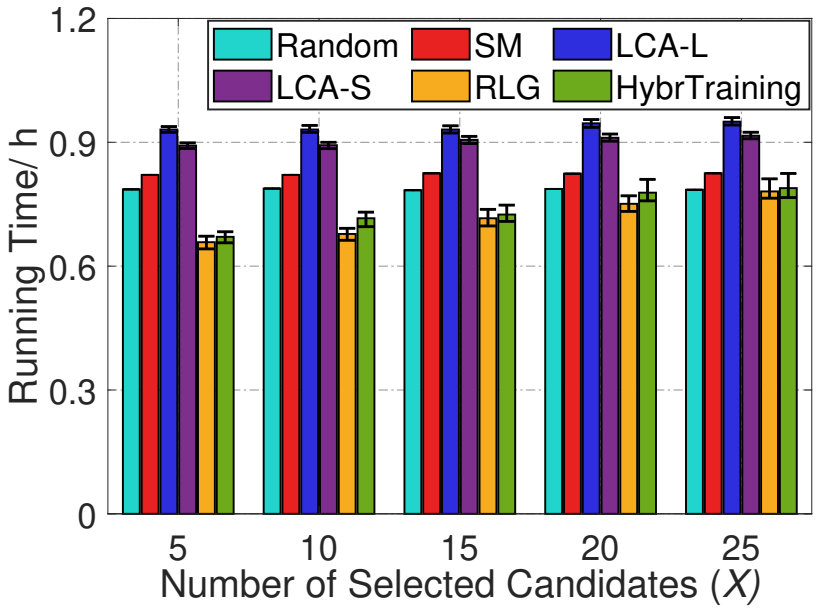


(a) Butterfly Dataset.

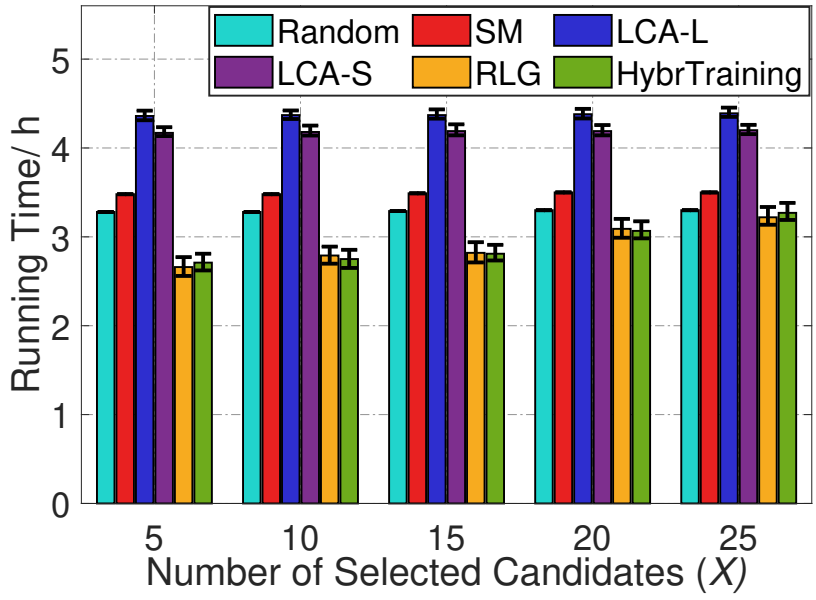


(b) Kuzushiji Dataset.

Figure 5.8: Average accuracy of the candidates selected by different algorithms for two real dataset. Experiments are run for different X (the number of selected users).



(a) Butterfly Dataset.



(b) Kuzushiji Dataset.

Figure 5.9: Selection efficiency of different algorithms on different datasets when the number of selected users for the formal task is different.

In the figures, colors indicate different assignment frequencies, namely, the percentage of the times the candidate is assigned to a training batch among all training batches in every five epochs. A higher assignment frequency corresponds to a brighter color, whereas darker colors indicate small assignment frequencies.

From Fig. 5.10 and Fig. 5.11, we observe that as the number of epochs grows, HybrTraining learns that the first candidate for either dataset has a high potential to learn the training samples better, and thus starts to assign more training batches to her. It is opposite for the under-performing candidate in Fig. 5.10 and Fig. 5.11. From HybrTraining's high accuracy ($X = 25$) in Fig. 5.8, we know that the final selected candidates are indeed better trained and achieve better performance for the formal tasks. When comparing different episodes, we find that the frequency of assignments varies because all training batches are sampled randomly in each epoch.

5.6.4 Model Transferability

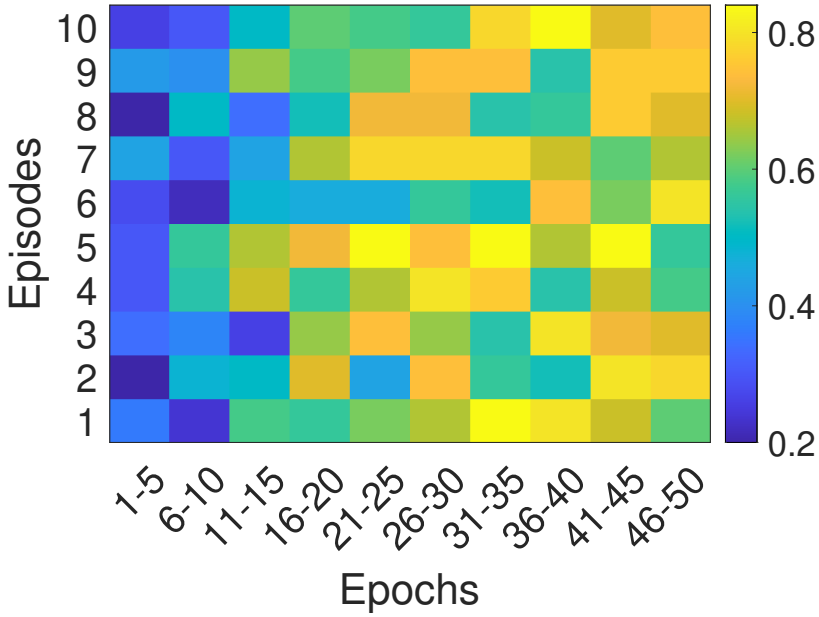
In practice, thousands of tasks are posted on crowdsourcing platforms every day. Hence, a natural concern of HybrTraining is its scalability. In general, the scalability issue exists in nearly all deep learning-based models. Transfer learning is an effective approach to tackling this problem. In this section, we investigate whether or not deep learning models of HybrTraining used for one task can be adapted for similar problem situations. Specifically, we comprehensively study the performance of pre-trained HybrTraining's DQN and LSTM on different but related problems.

Transferability of DQN

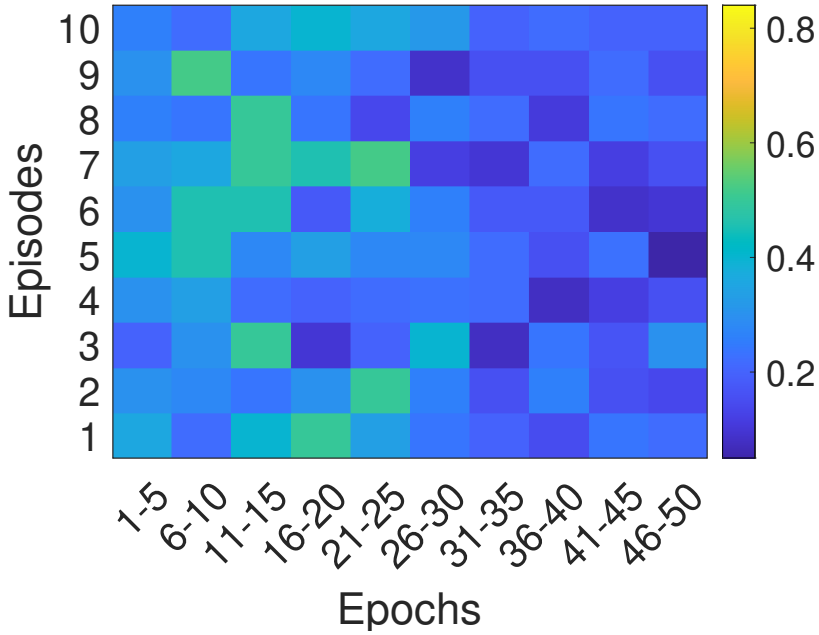
To evaluate the transfer learning performance of DQN, the pre-trained DQN model in Section 5.6.1 is used to initialize a new DQN model in new problem situations. When training the new DQN model, we freeze the inherited parameters from the pre-trained DQN except those of the last layer to test the transferability of the pre-trained DQN. We set different configurations (different kinds of new images and different replacement ratios) to comprehensively evaluate pre-trained DQN's transferability in HybrTraining.

- *Replaced by similar images*
  We use images of other kinds of butterflies to replace the images in the original "Butterfly" dataset introduced in Section 5.6.1. For "Kuzushiji" dataset, images from an extension "Kuzushiji" dataset, named "Kuzushiji-49", which contains $270,912$ hiragana images spanning 49 classes, are used to replace the images in the original "Kuzushiji" dataset. λ denotes the ratio of replaced images. X denotes the number of selected candidates in each episode. On both datasets, the training reward from randomly initialized DQN model and from pre-trained model are compared.

  Compared to the randomly initialized DQN model, the benefit of transfer learning can be clearly observed from the faster convergence speed and the high initial training reward in Fig. 5.12 and Fig. 5.13. As the replacement ratio increases, the advantage is less significant. In addition, it can be seen that larger X leads to more stable and efficient transfer learning because more training data can be learned by the DQN model as X grows.

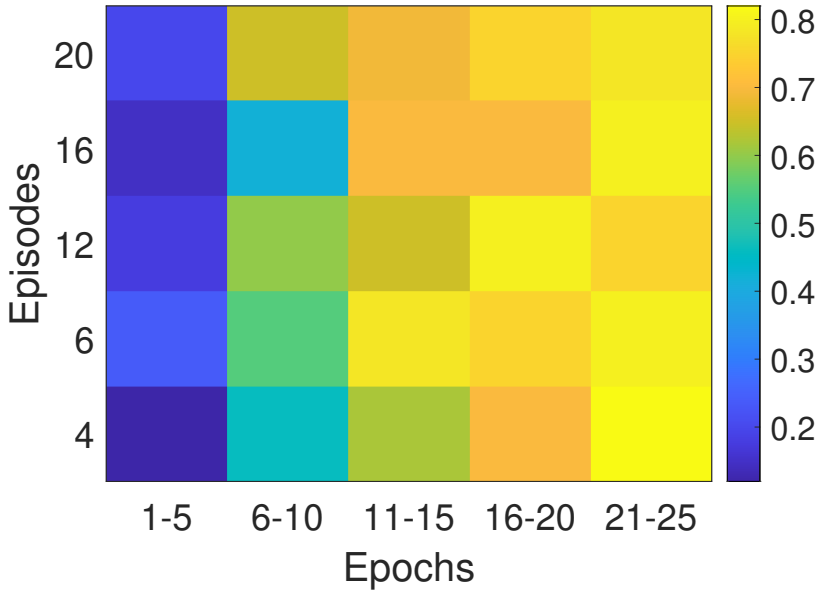


(a) Example of a suitable candidate who is selected to answer the formal tasks on "Butterfly" dataset.

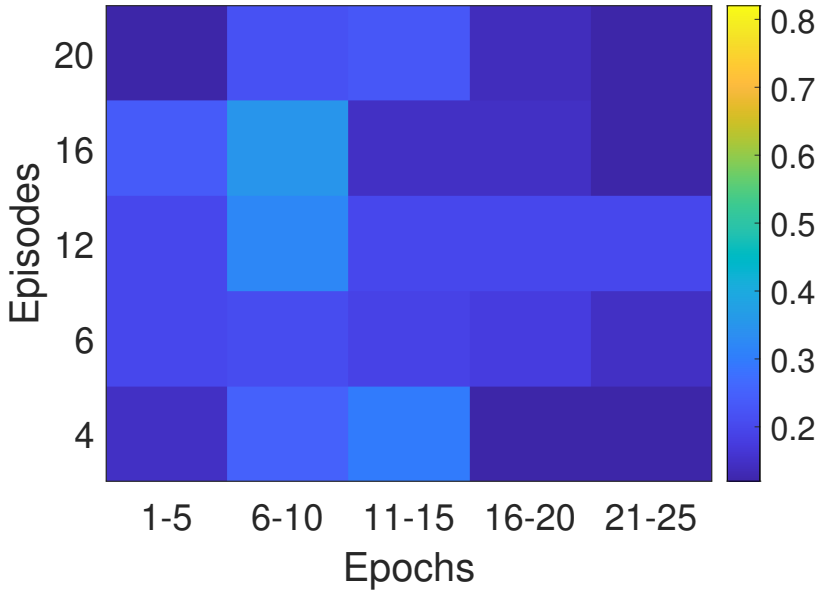


(b) Example of an unsuitable candidate who is not selected to answer the formal tasks on "Butterfly" dataset.

Figure 5.10: Visualizing the training batches assigned to two candidates in HybrTraining ($X = 25$) on the "Butterfly" dataset.

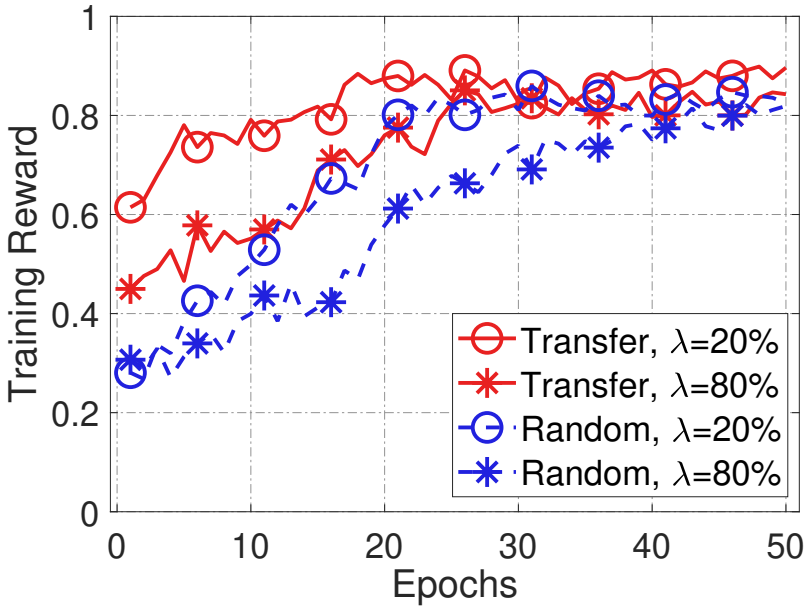


(a) Example of a suitable candidate who is selected to answer the formal tasks on "Kuzushiji" dataset.

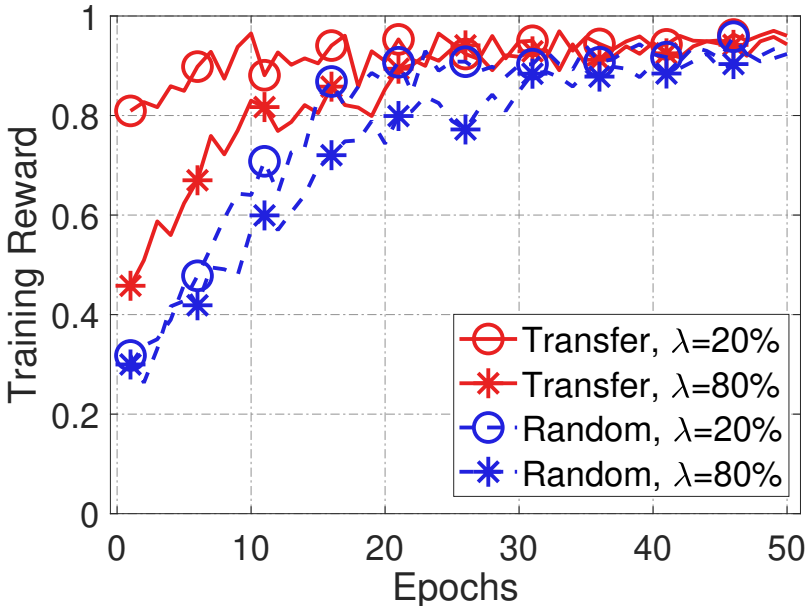


(b) Example of an unsuitable candidate who is not selected to answer the formal tasks on "Kuzushiji" dataset.

Figure 5.11: Visualizing the training batches assigned to two candidates in HybrTraining ($X = 25$) on the "Kuzushiji" dataset.

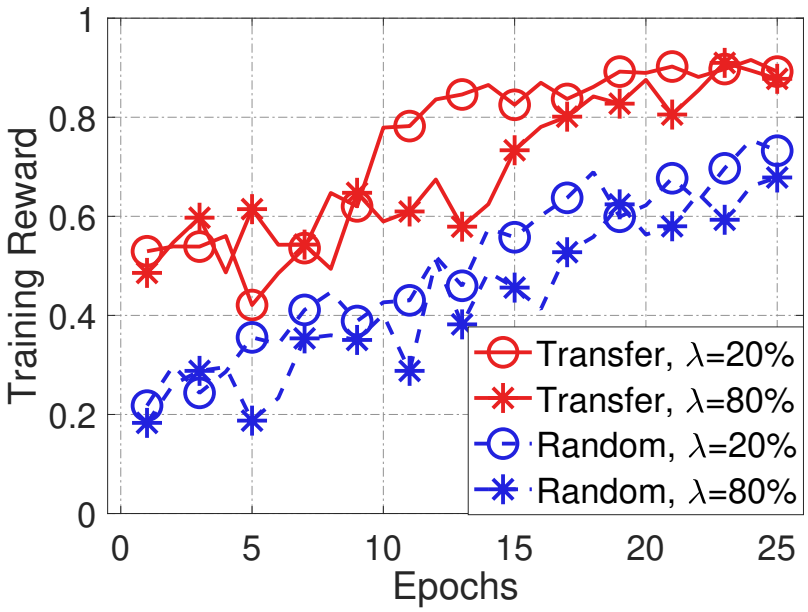


(a) $X = 5$.

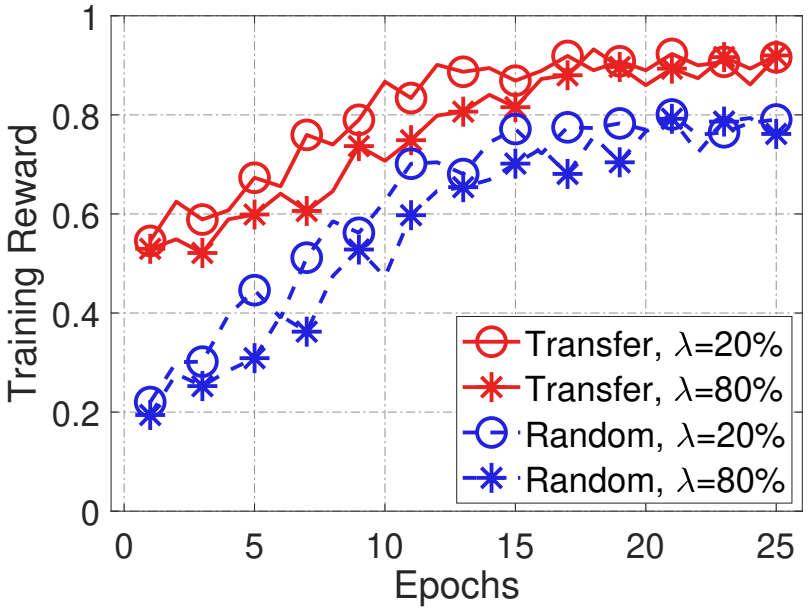


(b) $X = 25$.

Figure 5.12: Transfer learning with images of butterfly from other types. λ denotes the replacement ratio. Experiments are run for $X = 5$ and $X = 25$, where X denotes the number of selected candidates in each episode. We run 50 epochs under each X, with each epoch including 10 episodes.



(a) $X = 5$.



(b) $X = 25$.

Figure 5.13: Transfer learning with other Japanese hiragana images. λ denotes the replacement ratio. Experiments are run for $X = 5$ and $X = 25$, where X denotes the number of selected candidates in each episode. We run 25 epochs under each x, with each epoch including 20 episodes.

- *Replaced by less related images*
  To further evaluate the effect of transfer learning in more difficult problem situations, we use new images, which are less related to the images in the original 'Butterfly" and "Kuzushiji", to replace the images in both datasets. For "Butterfly", we use images of different kinds of seabeds to replace the original butterfly images. Seabed is the earth in the bottom of the ocean. Different areas generally have different types of seabed characterized by soil composition, topography, and rocks, all of which need human annotation. For "Kuzushiji", kanji characters, which are another kind of characters, are used to replace images in the original dataset. As it can be seen from Fig. 5.14 and Fig. 5.15, on the modified datasets, transfer learning shows faster convergence speed and higher initial training reward compared to random initialization. At the same time, it can be observed that larger replacement ratio leads to less benefit of transfer learning. Note that the impact on "Kuzushiji" is less than that on "Butterfly". This is possibly because more adaptable rules can be found in identifying objects than learning characters.

It is worth mentioning that for all the randomly initialized DQN models in this section, their initial training reward value is not close to zero. The reason is that in HybrTraining, DQN model cooperates with an LSTM model to decide which candidates can get training in each episode. In other words, when doing transfer learning on the modified datasets, a pre-trained LSTM is already equipped in HybrTraining. To evaluate the transferability of the LSTM model, we perform more experiments in the following section.

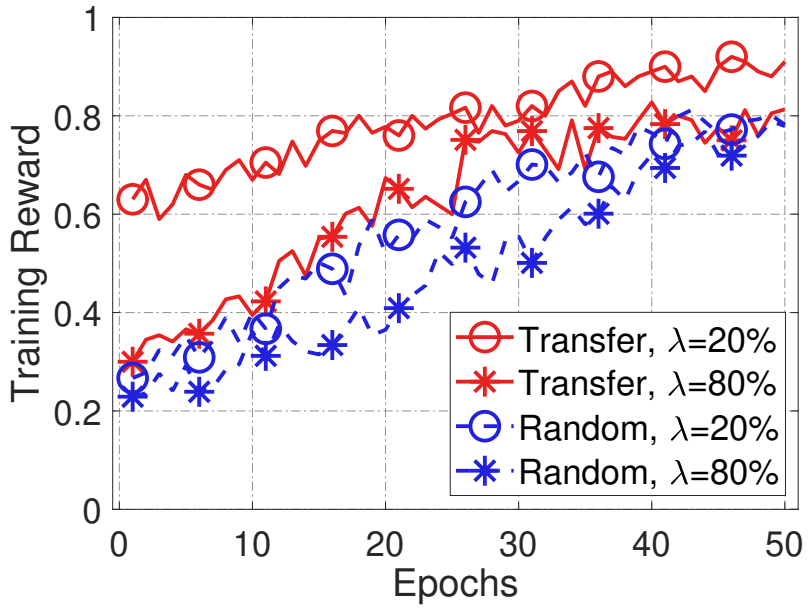
Transferability of LSTM

In this part, the pre-trained LSTM models in Section 5.6.1 will be used to initialize LSTM models in different problem situations. We set different configurations (different kind of new images to replace original images, different replacement ratios, different candidates) to evaluate transferability of the pre-trained LSTM model in HybrTraining.

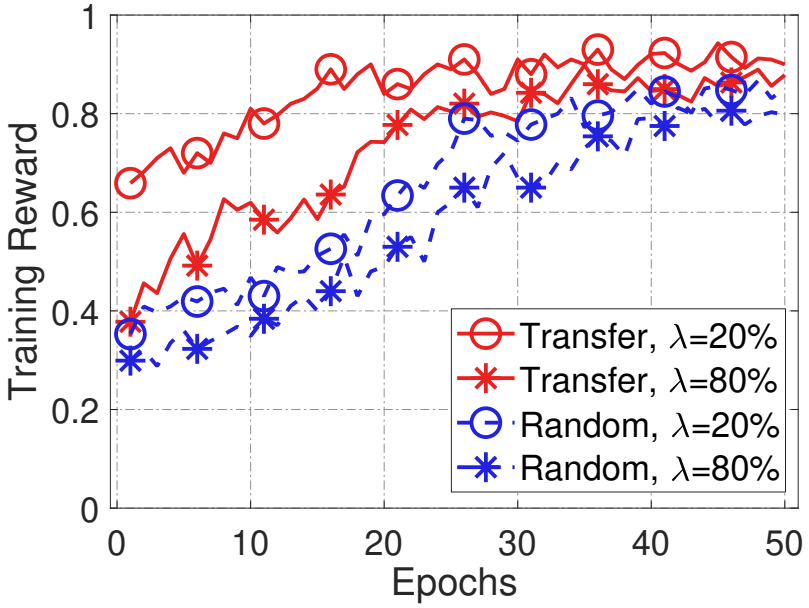
We scale all the LSTM loss to $[0, 1]$. In each episode, we take the average loss of ϵ candidates. For "Butterfly", each epoch contains 10 episodes. Each value plotted in the figures of this part is the average value of 10 episodes in each epoch. For "Kuzushiji", 20 episodes are contained in one epoch, so each value plotted in this part is the average of 20 episodes in each epoch.

- *Replaced by same category*
  Fig. 5.16 and Fig. 5.17 show the effect of LSTM model's transfer learning when a part of images in "Butterfly" and "Kuzushiji" are replaced by the images of other butterflies and other Japanese hiragana characters, respectively. We can observe that, when the ratio of replacement is 20%, transfer learning brings great benefit, especially on "Butterfly". This is because identifying similar objects may be easier than learning new characters. Meanwhile, we find that although HybrTraining's LSTM model is transferable even if λ is 80%, the advantage of transfer learning becomes less obvious when λ becomes high.

- *Replaced by different category*
  Fig. 5.18 and Fig. 5.19 show the transfer learning results of LSTM when less related images (i.e.,

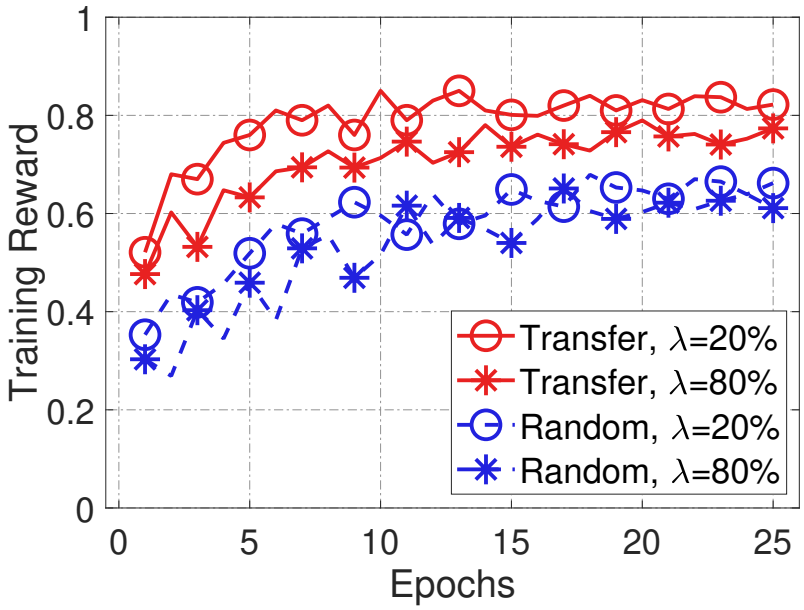


(a) $X = 5$.

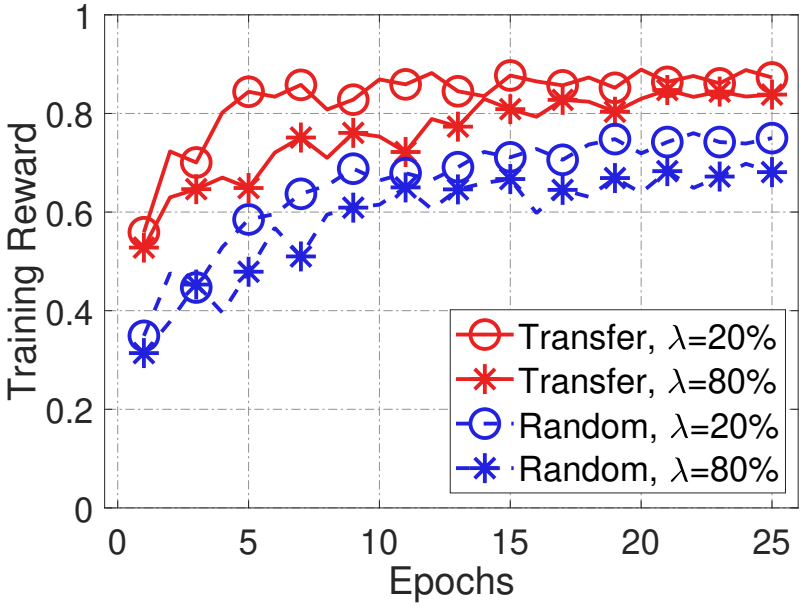


(b) $X = 25$.

Figure 5.14: Transfer learning with less related images on "Butterfly" dataset. λ denotes the replacement ratio. Experiments are run for $X = 5$ and $X = 25$, where X denotes the number of selected candidates in each episode. We run 50 epochs under each X, with each epoch including 10 episodes.



(a) $X = 5$.



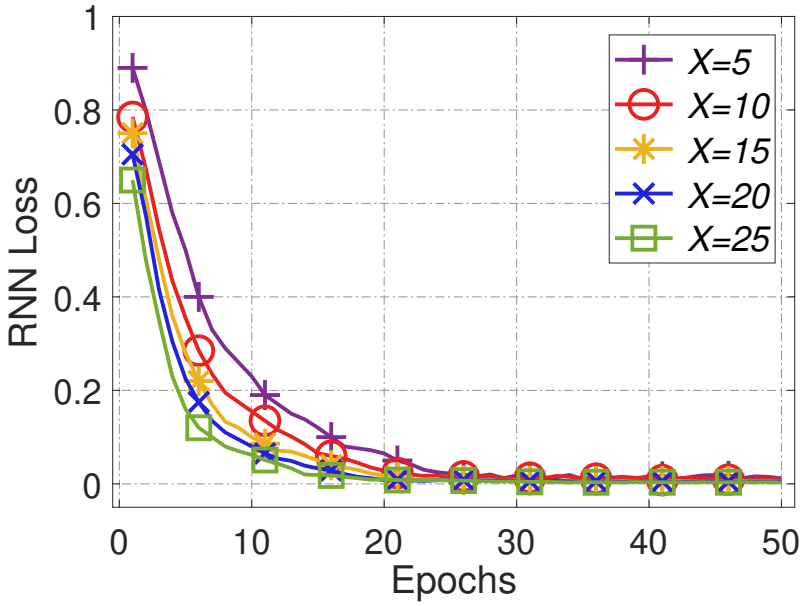
(b) $X = 25$.

Figure 5.15: Transfer learning with kanji images on "Kuzushiji" dataset. λ denotes the replacement ratio. Experiments are run for $X = 5$ and $X = 25$, where X denotes the number of selected candidates in each episode. We run 25 epochs under each X, with each epoch including 20 episodes.

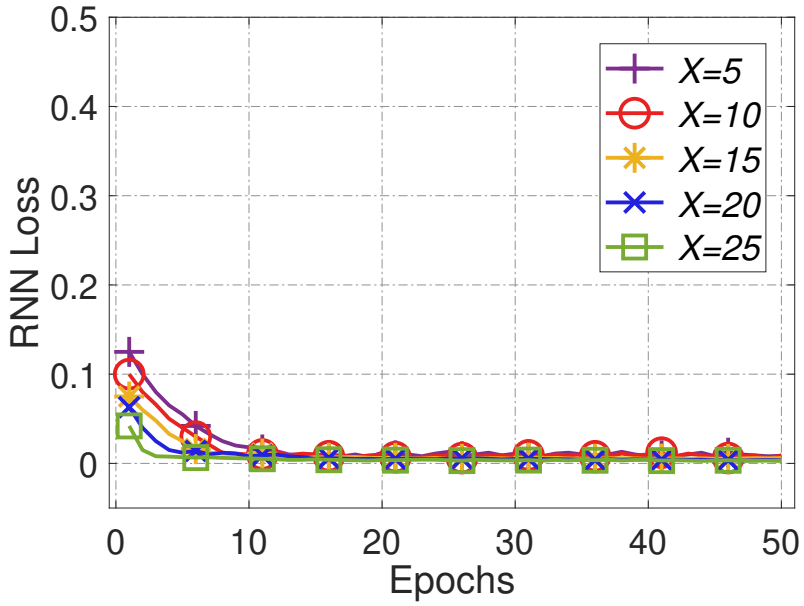
images of seabeds and kanji characters) are used to replace the original images in "Butterfly" and "Kuzushiji", respectively. Although the benefit brought by transfer learning is not as obvious as in Fig. 5.16 and Fig. 5.17 (i.e., when images of the new dataset are more related to the old ones), the advantage of transfer learning can still be observed when compared to random initialization.

- *Replace candidates*
  To investigate if pre-trained LSTM can be well transferred to work on different candidates, we ask another 100 candidates to participate in the training on original "Butterfly" and "Kuzushiji" in Section 5.6.1, then collect their feedback.

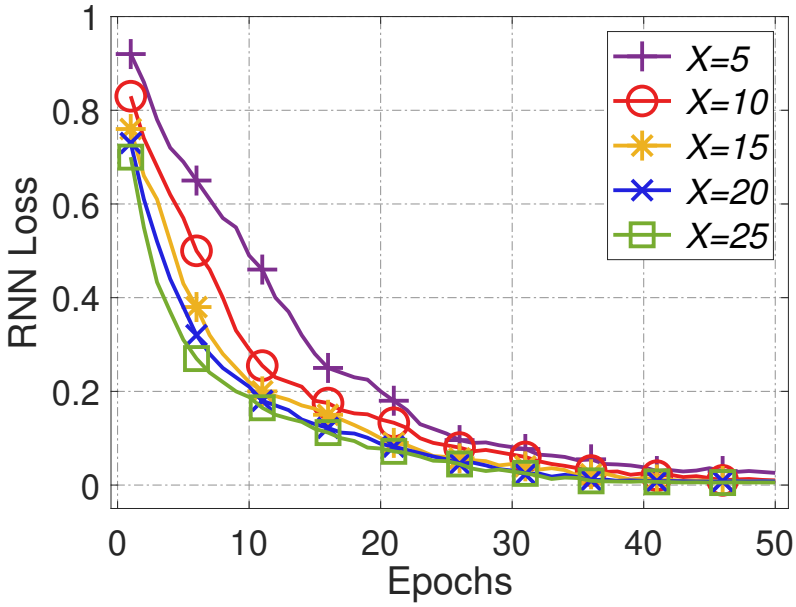
  We can clearly see from Fig. 5.20 that the pre-trained LSTM model outperforms randomly initialized LSTM model on the new candidates in terms of convergence speed and loss value. This means that the pre-trained LSTM model can be quickly adjusted to capture the learning pattern of new candidates. We can also observe that as X grows, the loss value converges faster. The reason is that LSTM model gets more data to learn in each episode as X increases. The setup of training set for LSTM under different X values is shown in Table 5.1. For "Kuzushiji", Fig. 5.21 demonstrates that transfer learning also outperforms randomly initialized LSTM model for different X values.



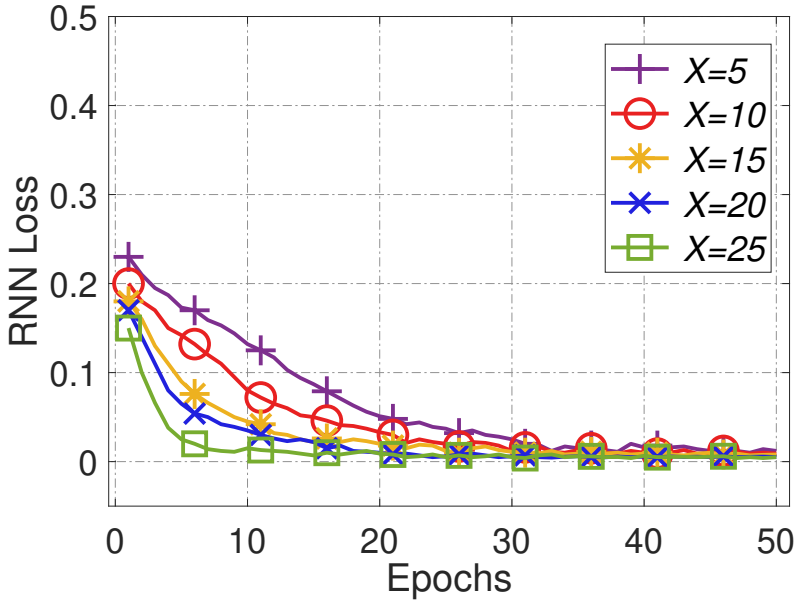
(a) $\lambda = 20\%$ with random initialization.



(b) $\lambda = 20\%$ with transfer learning.

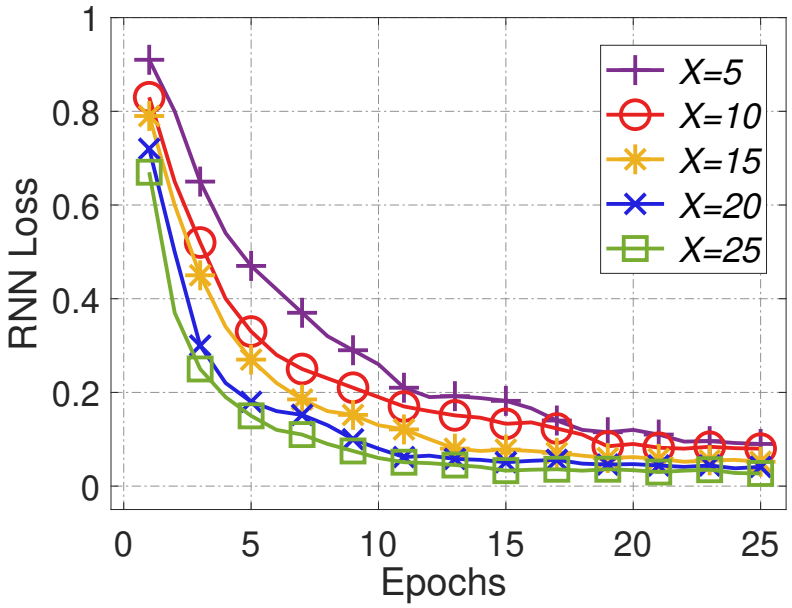


(c) $\lambda = 80\%$ with random initialization.

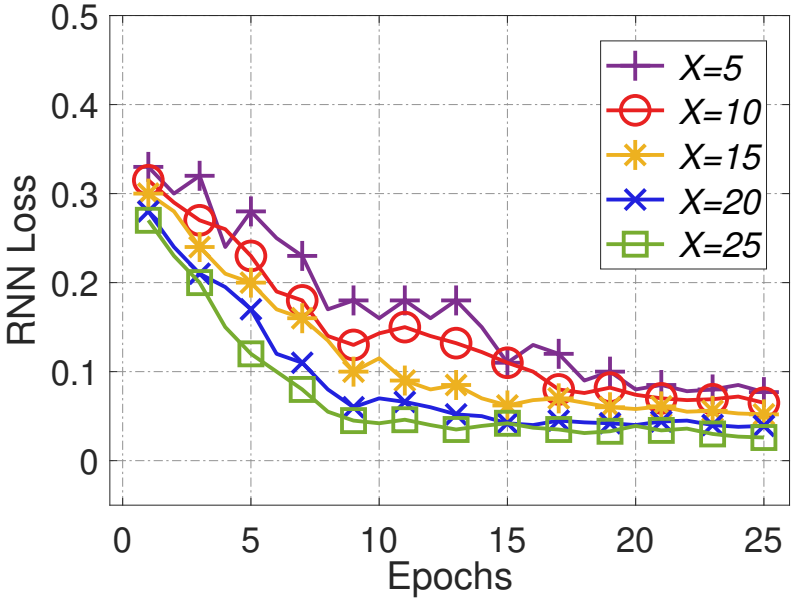


(d) $\lambda = 80\%$ with transfer learning.

Figure 5.16: LSTM transfer learning with images of other butterflies on "Butterfly" dataset. Experiments are run for different X and λ values, where X denotes the number of selected candidates in each episode and λ denotes the replacement ratio. For each λ, we run 50 epochs under each X, with each epoch including 10 episodes.



(a) $\lambda = 20\%$ with random initialization.



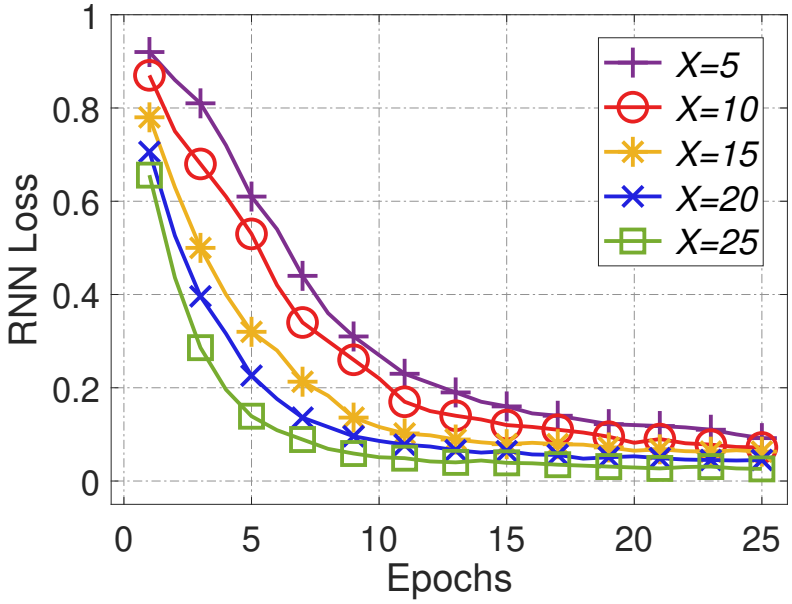
(b) $\lambda = 20\%$ with random initialization.

From the above comparisons we can conclude that the learned models in HybrTraining are transferable, particularly when the changes on the original datasets are moderate. This is because both DQN and LSTM can learn from past experience. When the problem scenarios are similar, the transition tuples saved in their experience buffers would be close. Thus, model parameters can be adapted quickly.

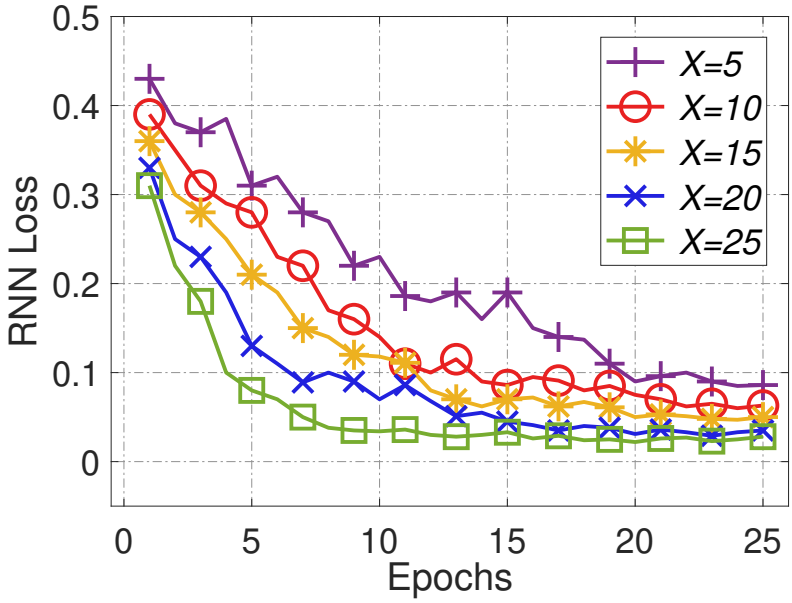
5.7 Conclusions

In this chapter, a reinforcement learning-based recruitment solution to crowdsourcing called "HybrTraining" was proposed. HybrTraining considers both the latent features of crowdworkers and the latent features of tasks and utilizes these features to increase training efficiency as well as the overall quality of the recruited candidates. In particular, HybrTraining orchestrates two deep learning models, LSTM and DQN, during the iterative training process, with LSTM to capture candidates' learning patterns and DQN to find the best match between the candidates and the training batches. This dynamic training strategy can reduce the training time, because candidates with consistently poor performance in the iterative training process will be kept from further training. It can also improve the overall recruitment quality, because it only selects the candidates, who perform well during the training and best match the formal tasks at the end of training, for the formal tasks.

Real-world experiments revealed that compared with a greedy Q-learning based strategy, HybrTraining recruited candidates who achieved better performance on the formal tasks with similar selection efficiency. Furthermore, it outperformed both static and dynamic popular worker recruitment algorithms in terms of efficiency, accuracy, and stability. We also demonstrated that both the DQN and LSTM in HybrTraining are transferable when different changes happen in input parameters.

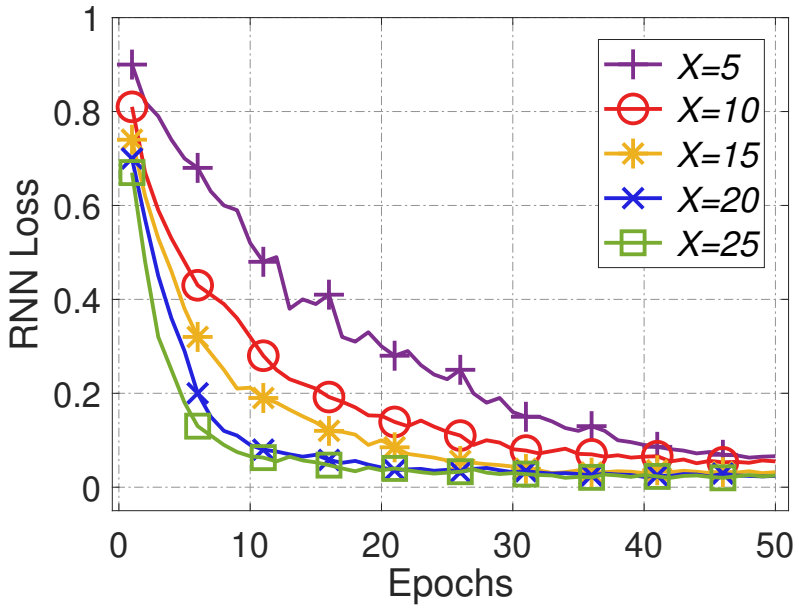


(c) $\lambda = 80\%$ with random initialization.

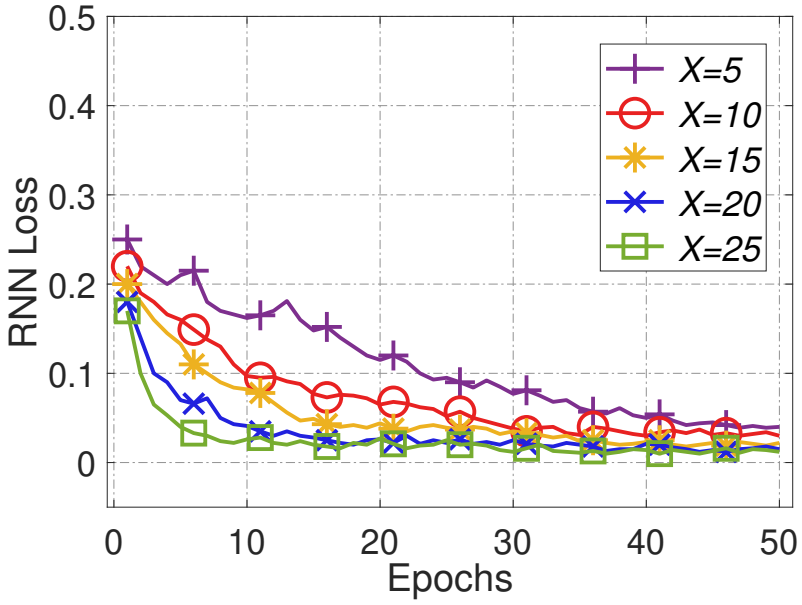


(d) $\lambda = 80\%$ with transfer learning.

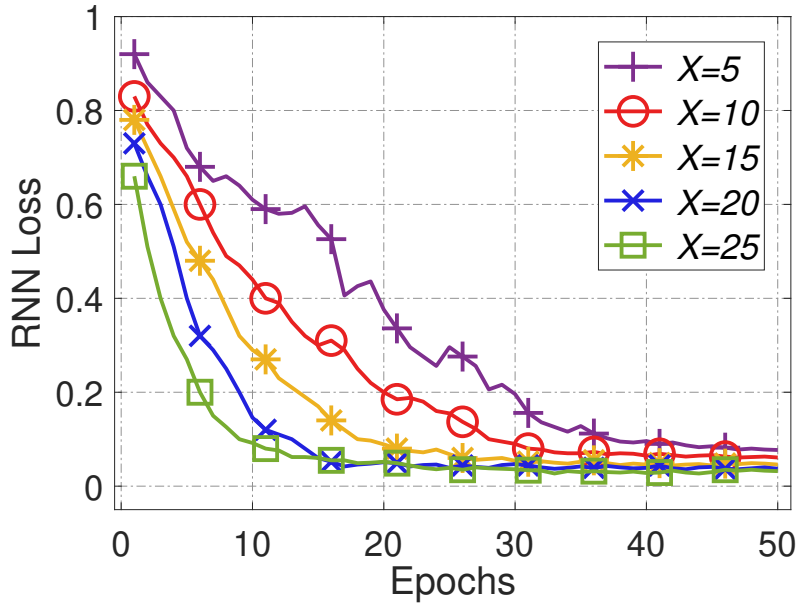
Figure 5.17: LSTM transfer learning with images of other Japanese hiragana characters on "Kuzushiji" dataset. Experiments are run for different X and λ values, where X denotes the number of selected candidates in each episode and λ denotes the replacement ratio. For each λ, we run 25 epochs under each X, with each epoch including 20 episodes.



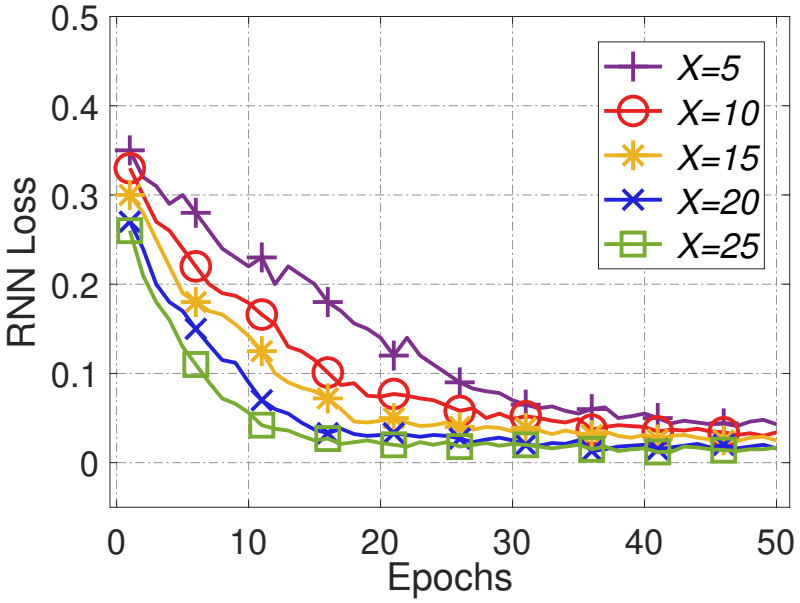
(a) $\lambda = 20\%$ with random initialization.



(b) $\lambda = 20\%$ with transfer learning.

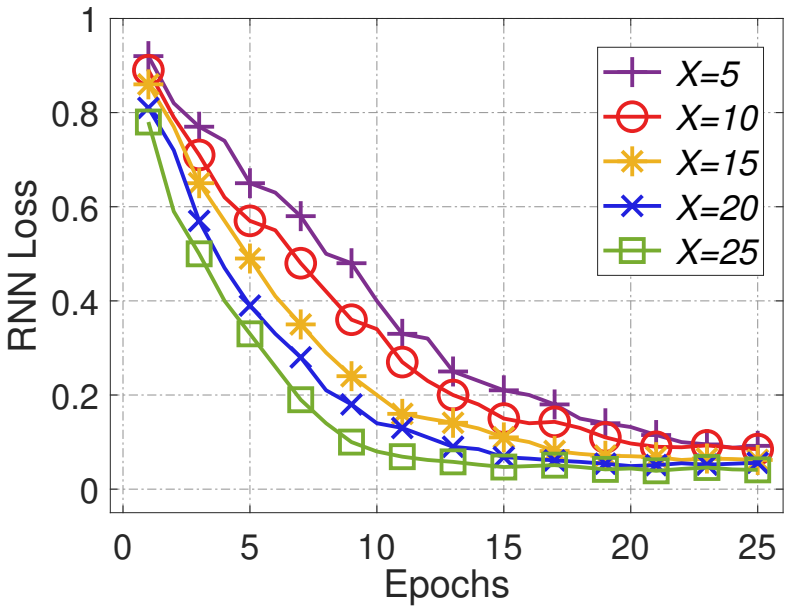


(c) $\lambda = 80\%$ with random initialization.

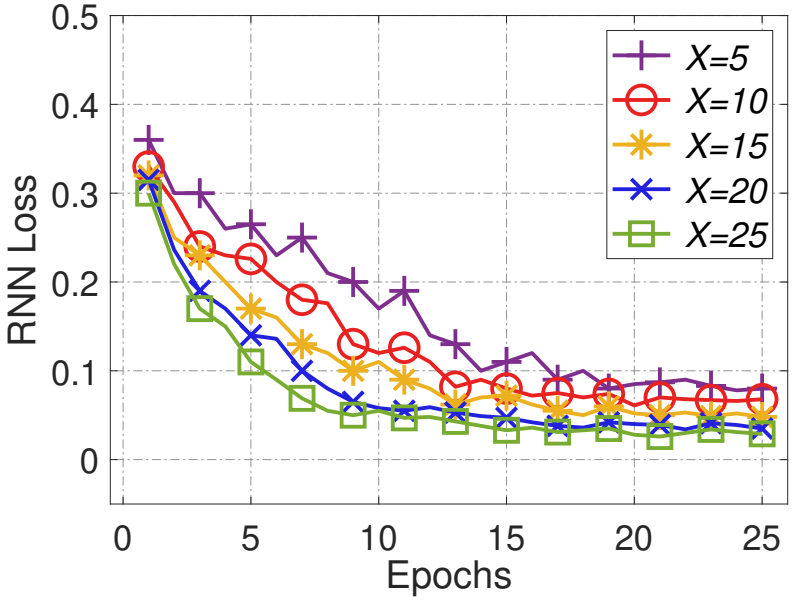


(d) $\lambda = 80\%$ with transfer learning.

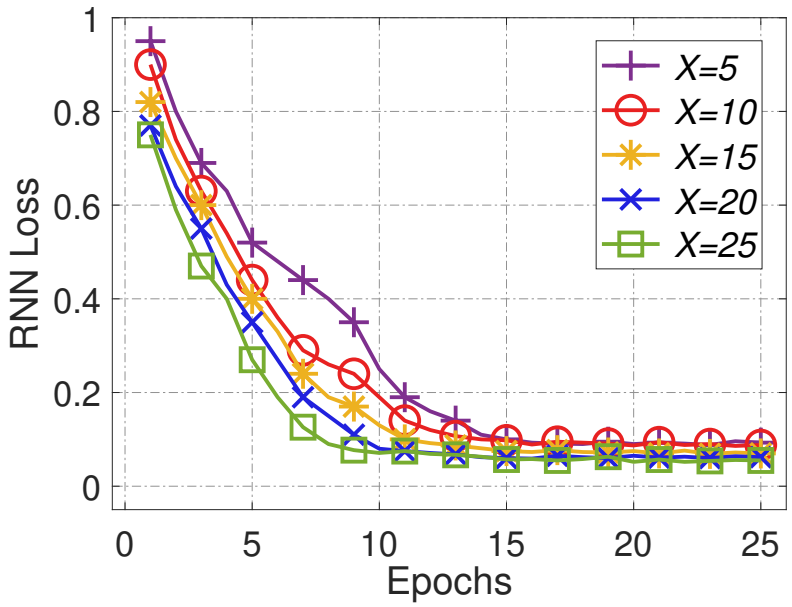
Figure 5.18: LSTM transfer learning with images of seabeds on "Butterfly" dataset. Experiments are run for different X and λ values, where X denotes the number of selected candidates in each episode and λ denotes the replacement ratio. For each λ, we run 50 epochs under each X, with each epoch including 10 episodes.



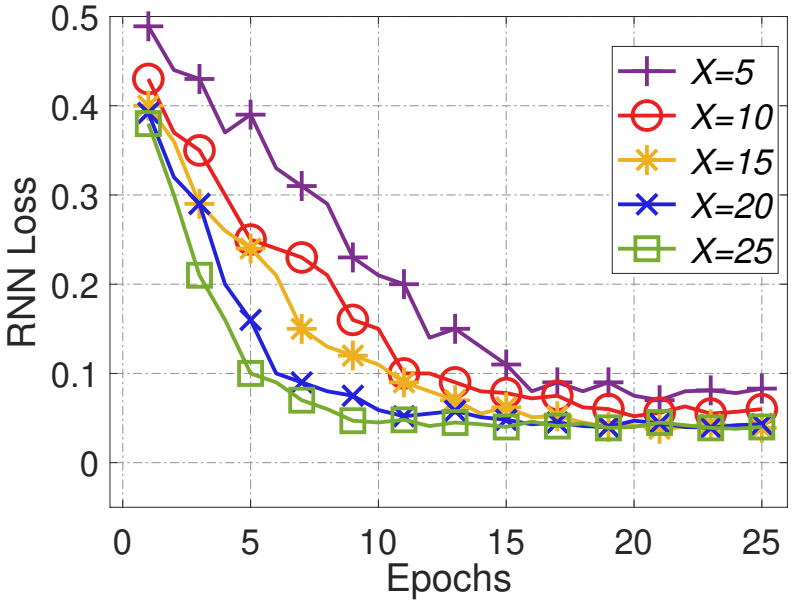
(a) $\lambda = 20\%$ with random initialization.



(b) $\lambda = 20\%$ with transfer learning.

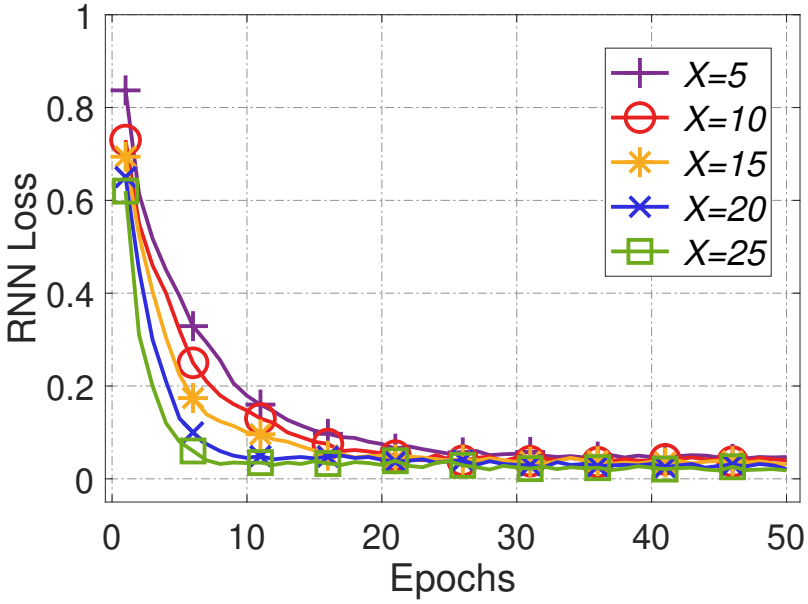


(c) $\lambda = 80\%$ with random initialization.

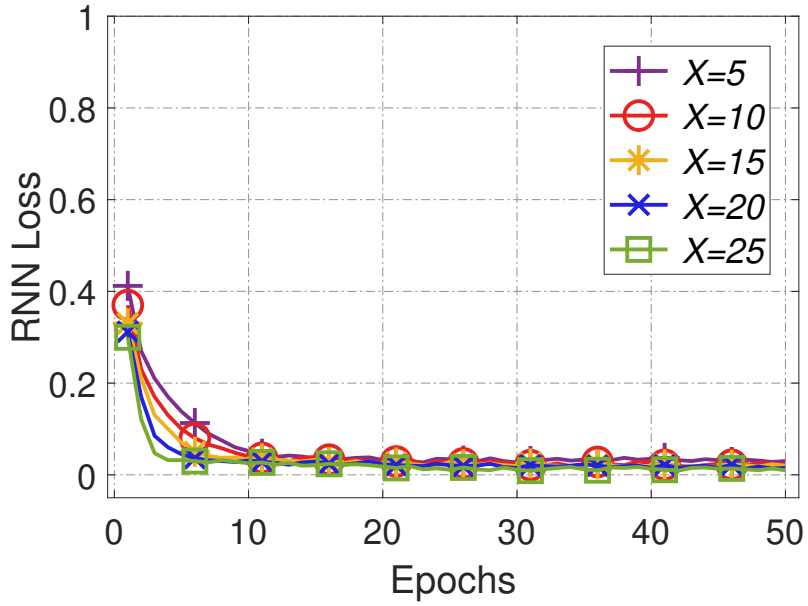


(d) $\lambda = 80\%$ with transfer learning.

Figure 5.19: LSTM transfer learning with images of kanji characters on "Kuzushiji" dataset. Experiments are run for different X and λ values, where X denotes the number of selected candidates in each episode and λ denotes the replacement ratio. For each λ, we run 25 epochs under each X, with each epoch including 20 episodes.

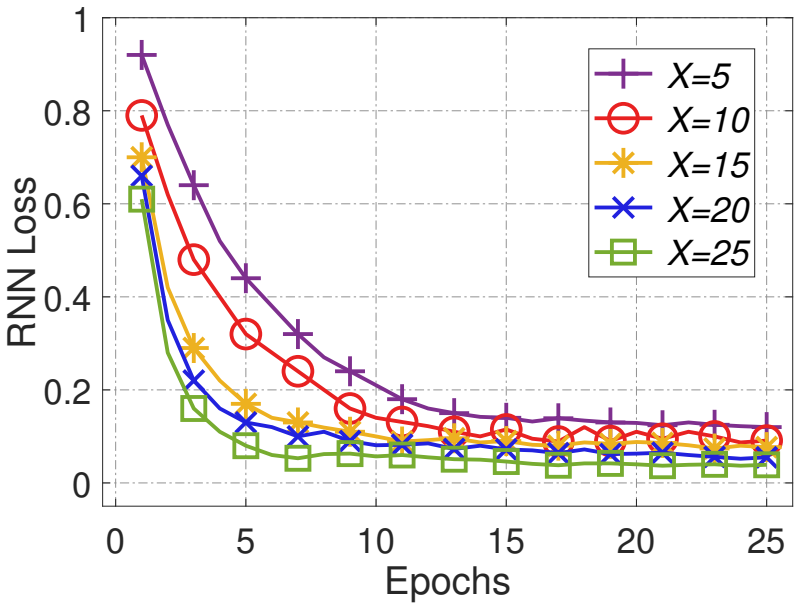


(a) Training performance of random initialized LSTM model.

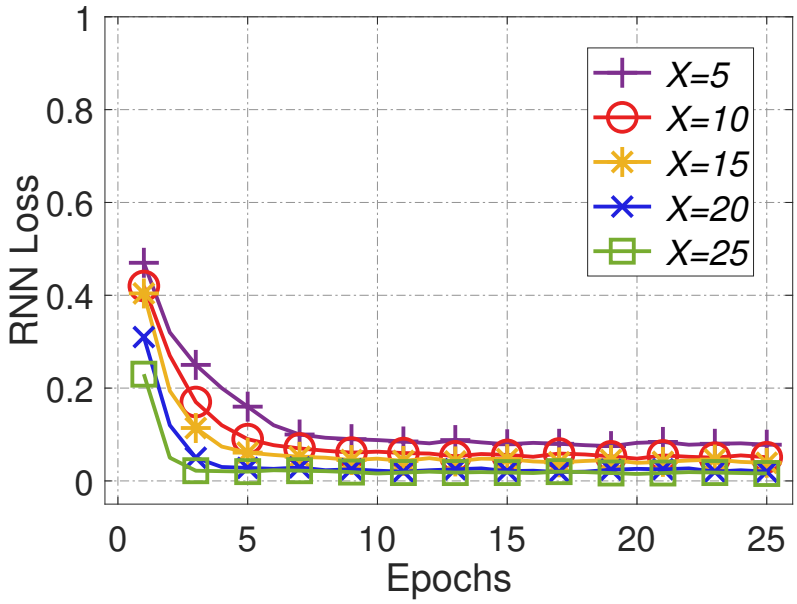


(b) Training performance with transfer learning.

Figure 5.20: LSTM transfer learning performance on new candidates without/with transfer learning in "Butterfly" dataset. Experiments are run for different X values, where X denotes the number of selected candidates in each episode. We run 50 epochs under each X, with each epoch including 10 episodes.



(a) Training performance of random initialized LSTM model.



(b) Training performance with transfer learning.

Figure 5.21: LSTM transfer learning performance on new candidates without/with transfer learning in "Kuzushiji" dataset. Experiments are run for different X values, where X denotes the number of selected candidates in each episode. We run 25 epochs under each X, with each epoch including 20 episodes.

Chapter 6

Comparison of Goldilocks and HybrTraining

We introduced a learning pattern recognition framework "Goldilocks" in Chapter 4 and a dynamic recruitment model "HybrTrainig" in Chapter 5, respectively. Since both models work in the same MCS/CS stage and both of them are designed for optimizing MCS/CS recruitment process, we compare the two models in this chapter for MCS/CS platforms to employ the suitable recruitment model based on the feature of different situations.

6.1 Goldilocks

In Goldilocks, one training sample is assigned to candidates each time to align candidates' learning curve. Candidates are required to finish all the questions in the training set. The matching between a candidate and a training sample each time is calculated based on this candidate's historical performance during training, because the learning pattern recognition in Goldilocks is for worker selection. The final recruitment result depends on the degree of fitting between candidates' learning curve and the desired learning curve from task requester.

Compared to HybrTraining, Goldilocks is recommended for more subjective scenarios where task requester has clear idea or preference on the desired learning curves. For example, facing an urgent task, the requester may sacrifice quality for time by defining a desired learning curve in which the learning speed plays a major role. Another suitable scenarios for Goldilocks is when the training set is small (e.g. less than 100 training samples) and the task is one-time task. In this case, the candidates need to finish all training samples because a complete learning curve is required for the final recruitment. Since the desired learning curve of a candidate on one task may not be a good indication of their performance on other tasks, we need to run Goldilocks separately for different tasks.

6.2 HybrTraining

HybrTraining is a reinforcement learning-based recruitment model where a training batch is only assigned to k candidates from n candidates a time. We apply an explorer [63] to ensure the training chance for all candidates at the beginning when the system does not know each candidate well. The feature of training samples in each training batch is also considered during the matching between training batches and workers. The learning pattern recognition in HybrTraining is for training candidates in a personalized way, i.e., capturing the training batch that can better train each candidate. The final recruitment result is decided by the feature of formal task and each candidate' feature learnt from the training. Therefore, even if some candidates may not perform well on a certain training batch, it is still possible to select them for the formal task when the feature of the formal task is not similar to that in the training batch.

HybrTraining is recommended for scenarios where the training set is large so the two deep neural networks can be trained, and the trained model can be reused for other similar tasks. As we showed in Section 5.6.4, HybrTraining has good transferability.

Overall, Goldilocks and HybrTraining are designed to handle different application scenarios. As such, we can only provide a qualitative comparison summarized in Table 6.1.

Table 6.1: Qualitative Comparison between Goldilocks and HybrTraining

Method / Feature	Goldilocks	HybrTraining
Selection Basis	fitting degree between learning curve and desired curve	fitting degree between candidate feature and formal batch feature
Pattern Learning	for worker selection	for both worker training and work selection
Training	same training batch for all candidates	different training batches for different candidates
Suitable Scenario	(1) clear description of the desired learning pattern from the requester (2) small training set	(1) needs for re-using the trained model (2) large training set

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we proposed three machine learning based methods to solve the data quality enhancement problem in MCS/CS.

First, to find the true answers from the conflicting data collected with MCS/CS, we proposed a novel approach that infuses expert knowledge into the crowded data by asking experts to label a small amount of data. The results from real-world case studies as well as simulation demonstrated that (1) our model outperforms other existing MCS/CS truth discovery methods in terms of effectiveness and efficiency, and (2) our method works well on estimating the expertise level of different participants and the difficulty level of different tasks.

Second, to adaptively train different crowd workers so that the right workers can be matched to the right MCS/CS job, we developed a model for interactive training and learning pattern recognition. Based on the extracted learning patterns of individuals after the training phase, we can effectively select suitable candidates for the formal task. By evaluating our model over two datasets, we validated the benefits of learning pattern recognition during the MCS/CS recruitment process. The experiment results show that the candidates recruited by our model can finish the formal MCS/CS tasks more accurately and more efficiently than the candidates recruited by other baseline methods.

Last but not least, we extended the interactive training framework to dynamic, deep learning-based recruitment framework. In our extended framework, a DQN-based worker-task matching model and an LSTM-based learning pattern recognition model cooperate together to improve both the training efficiency and the worker-task matching efficiency on MCS/CS platforms. Experimental results over two real-world datasets show that our solution can train and estimate candidates from a long-term perspective to eventually achieve both good training result and satisfactory recruitment result.

7.2 Future Work

7.2.1 Automated Knowledge Infusion Strategy Discovery

In this thesis, we focus on finding ground truth from conflicting MCS/CS data by infusing a certain amount of expert labelled data in Chapter 3. We believe in practice this method can be further improved by considering special features in different MCS/CS scenarios. In addition, our knowledge infusion framework would be more powerful if it could automatically determine the right amount of data that should be labelled by experts for a posted MCS/CS task. In more detail, the following research problems deserve further investigation:

- How should we automatically adjust the amount of expert knowledge based on the similarity in crowd workers' answers?
- How should we design knowledge infusion strategies under limited budget?
- In chapter 3, we do not distinguish human workers from AI-based smart devices. If we consider the difference between these two kinds of crowd workers, what is the impact of detailed AI algorithms on the final truth discovery results?

7.2.2 Extension on the Interactive Learning Pattern Recognition Framework

The main drawback of Goldilocks is that we need to build the learning curve for each worker. This overhead may be too high in large-scale MCS systems. To overcome this problem, a good strategy is to use a small number of fixed learning patterns, based on which we infer the probability distribution of each worker corresponding to the fixed learning patterns. To be more specific, assume that we have a set of learning pattern $\mathcal{L} = \{L_1, L_2, \ldots, L_N\}$, we only need to infer the probability distribution of each worker $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$, where p_i denotes the probability that the user's learning curve belongs to learning pattern L_i. The benefit is that we only need to ask a smaller number of questions to infer the distribution.

The research agenda along this line of research includes:

- Based on existing data, group users' learning curves into fixed number of learning patterns.
- Based on the fixed learning patterns, develop a *questioning strategy* such that we can infer a user's $\mathcal{P}$ quickly and accurately (enough for the MCS assignment). We can use the entropy concept to determine when the accuracy meets the requirement.
- Under the same Goldilocks framework, evaluate the benefit of the extended research, e.g., saving on the number of asked questions and on the training time.

7.2.3 Battery-aware Fine-Grained User Profiling in MCS Recruitment

Finding the best match between users and tasks has been a crucial problem in MCS. In our thesis, we solved this matching problem by considering participants' capability and learning pattern. During

the research, we also found that in practice, people are highly sensitive to the battery level of their mobile devices and may drastically change their behavior if the battery level becomes low, even for the same sensing task. Using photographing as an example, when the battery level drops to a certain threshold (e.g., 20% according to an online survey [81]), the majority of users are unlikely to accept new tasks. At the same time, to save their device energy, some users may quickly take a photo of the requested object without caring about the image quality, which will lead to low MCS sensing data quality. Based on these observations, we should consider the behavior changes (e.g., a user's energy-anxiety curve [81]) caused by mobile devices' energy level.

The research agenda along this line of research includes:

- Analyze the survey data and build an empirical model to capture users' energy-anxiety-attribute relation. The model will output the inferred likelihood of accepting a given sensing task based on the battery level of a user's mobile phone, her anxiety degree, and her personal attribute (such as age, gender). Also, the model should be able to infer the quality/reliability of a sensing task based on the battery level of the worker's mobile phone.

- In real-world application, a user may not be willing to report the battery level of her mobile device. Nevertheless, there is an opportunity to infer her device's energy level (at least the range), based on her behavior observed over the MCS platform, for example, whether or not the user has quickly accepted a task, whether or not the user is using GPS, the distance between the user and some anchor places that may have charging stations (e.g., airport and cafeteria).

- Develop an analytical model for MCS task assignment by considering (1) the energy level of mobile device (reported or inferred) and (2) the user's behavior profile with the changing energy level.

- Evaluate the performance of proposed new MCS task assignment strategy.

Bibliography

[1] Appen, CO. Appen, 2021-05-17.

[2] L. Aroyo and C. Welty. The three sides of crowdtruth. *Human Computation*, 1(1), 2014.

[3] AWS. *Amazon Web Services*.

[4] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver. How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing. *arXiv preprint arXiv:1206.6386*, 2012.

[5] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] D. Boud, R. Cohen, and D. Walker. *Using experience for learning*. McGraw-Hill Education (UK), 1993.

[7] D. C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90, 2008.

[8] D. C. Brabham. Moving the crowd at istockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. *First monday*, 2008.

[9] T. Burger-Helmchen and J. Pénin. The limits of crowdsourcing inventive activities: What do transaction cost theory and the evolutionary theories of the firm teach us. In *Workshop on Open Source Innovation*, pages 1–26, 2010.

[10] C. Canada. Computecanada, 2021-05-17.

[11] S. Castano, A. Ferrara, L. Genta, and S. Montanelli. Combining crowd consensus and user trustworthiness for managing collective tasks. *Future Generation Computer Systems*, 54:378–388, 2016.

[12] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

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

[14] G. Creamer and S. Stolfo. A link mining algorithm for earnings forecast and trading. *Data mining and knowledge discovery*, 18(3):419–445, 2009.

[15] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40, 2018.

[16] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[17] K. Diaz-Chito, A. Hernández-Sabaté, and A. M. López. A reduced feature set for driver head pose estimation. *Applied Soft Computing*, 45:98–107, 2016.

[18] Django. *Django Web Framework*.

[19] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, 2011.

[20] J. Du and C. X. Ling. Active teaching for inductive learners. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 851–861. SIAM, 2011.

[21] J. Dutta, F. Gazi, S. Roy, and C. Chowdhury. Airsense: Opportunistic crowd-sensing based air quality monitoring system for smart city. In *2016 IEEE SENSORS*, pages 1–3. IEEE, 2016.

[22] H. Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

[23] EC2. *Elastic Compute Cloud*.

[24] B. Efron and T. Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

[25] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Trans. Sens. Networks*, 6(1):6–1, 2009.

[26] C. Fiandrino, B. Kantarci, F. Anjomshoa, D. Kliazovich, P. Bouvry, and J. Matthews. Sociability-driven user recruitment in mobile crowdsensing internet of things platforms. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[27] U. Gadiraju, G. Demartini, R. Kawase, and S. Dietze. Crowd anatomy beyond the good and bad: Behavioral traces for crowd worker modeling and pre-selection. *Computer Supported Cooperative Work (CSCW)*, 28(5):815–841, 2019.

[28] U. Gadiraju, B. Fetahu, and R. Kawase. Training workers for improving performance in crowdsourcing microtasks. In *Design for Teaching and Learning in a Networked World*, pages 100–114. Springer, 2015.

[29] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE communications Magazine*, 49(11):32–39, 2011.

[30] T. P. Grantcharov, L. Bardram, P. Funch-Jensen, and J. Rosenberg. Learning curves and impact of previous operative experience on performance on a virtual reality simulator to test laparoscopic surgical skills. *The American journal of surgery*, 185(2):146–149, 2003.

[31] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han. Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems. *IEEE Transactions on Human-Machine Systems*, 47(3):392–403, 2017.

[32] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346, 2014.

[33] J. Han and C. Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[34] H. V. Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[35] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*, 2015.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[37] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[38] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, 2006.

[39] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer. An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 1–15. Springer, 2013.

[40] InnoCentive. *InnoCentive*.

[41] iStockphoto. *iStockphoto*.

[42] J. Jiang, Y. Dai, K. Wu, and R. Zheng. Goldilocks: Learning pattern-based task assignment in mobile crowdsensing. In *Quality, Reliability, Security and Robustness in Heterogeneous Systems- 15th EAI International Conference, QShine 2019, Shenzhen, China, November 22-23, 2019, Proceedings*, volume 300 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 63–83. Springer, 2019.

[43] J. Jiang, K. Wu, H. Wang, and R. Zheng. Automatic data quality enhancement with expert knowledge for mobile crowdsensing. In *38th IEEE International Performance Computing and Communications Conference, IPCCC 2019, London, United Kingdom, October 29-31, 2019*, pages 1–7. IEEE, 2019.

[44] E. Johns, O. Mac Aodha, and G. J. Brostow. Becoming the expert-interactive multi-class machine teaching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2624, 2015.

[45] V. Jones and J. H. Jo. Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology. In *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, volume 468, page 474. Perth, Western Australia, 2004.

[46] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos. User recruitment for mobile crowdsensing over opportunistic networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2254–2262. IEEE, 2015.

[47] A. R. Khan and H. Garcia-Molina. Crowddqs: Dynamic question selection in crowdsourcing systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1447–1462, 2017.

[48] S. Kim, C. Robson, T. Zimmerman, J. Pierce, and E. M. Haber. Creek watch: pairing usefulness and usability for successful citizen science. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2125–2134, 2011.

[49] S.-a. Knight. *User perceptions of information quality in World Wide Web information retrieval behaviour*. Edith Cowan University Perth, 2007.

[50] W. Kool, H. Van Hoof, and M. Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[51] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9):140–150, 2010.

[52] H. Li, T. Li, and Y. Wang. Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 136–144. IEEE, 2015.

[53] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198, 2014.

[54] C. H. Liu, B. Zhang, X. Su, J. Ma, W. Wang, and K. K. Leung. Energy-aware participant selection for smartphone-enabled mobile crowd sensing. *IEEE Systems Journal*, 11(3):1435–1446, 2017.

[55] Q. Liu, S. Tong, C. Liu, H. Zhao, E. Chen, H. Ma, and S. Wang. Exploiting cognitive structure for adaptive learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 627–635, 2019.

[56] S. Liu, Z. Zheng, F. Wu, S. Tang, and G. Chen. Context-aware data quality estimation in mobile crowdsensing. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[57] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[58] P. Mavridis, D. Gross-Amblard, and Z. Miklós. Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing. In *Proceedings of the 25th International Conference on World Wide Web*, pages 843–853, 2016.

[59] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th acm conference on embedded networked sensor systems*, pages 169–182, 2015.

[60] C. Meng, H. Xiao, L. Su, and Y. Cheng. Tackling the redundancy and sparsity in crowd sensing applications. In *Proceedings of the 14th ACM Conference on embedded network sensor systems CD-ROM*, pages 150–163, 2016.

[61] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 183–196, 2015.

[62] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[63] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[64] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336, 2008.

[65] Mturk. *Mturk*.

[66] A. Nguyen, B. Wallace, and M. Lease. Combining crowd and expert labels using decision theoretic active learning. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 3, 2015.

[67] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[68] R. W. Ouyang, M. Srivastava, A. Toniolo, and T. J. Norman. Truth discovery in crowdsourced detection of spatial events. *IEEE transactions on knowledge and data engineering*, 28(4):1047–1060, 2015.

[69] K. R. Patil, X. Zhu, Ł. Kopeć, and B. C. Love. Optimal teaching for limited-capacity human learners. In *Advances in neural information processing systems*, pages 2465–2473, 2014.

[70] D. Peng, F. Wu, and G. Chen. Pay as how well you do: A quality based incentive mechanism for crowdsensing. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 177–186, 2015.

[71] S. S. Pereira, R. Lòpez-Valcarce, and A. Pagès-Zamora. A diffusion-based em algorithm for distributed estimation in unreliable sensor networks. *IEEE Signal Processing Letters*, 20(6):595–598, 2013.

[72] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2011.

[73] L. Pu, X. Chen, J. Xu, and X. Fu. Crowdlet: Optimal worker recruitment for self-organized mobile crowdsourcing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[74] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen. Image browsing, processing, and clustering for participatory sensing: lessons from a dietsense prototype. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 13–17, 2007.

[75] N. Shah, D. Zhou, and Y. Peres. Approval voting and incentives in crowdsourcing. In *International conference on machine learning*, pages 10–19. PMLR, 2015.

[76] C. Shan, N. Mamoulis, R. Cheng, G. Li, X. Li, and Y. Qian. An end-to-end deep rl framework for task arrangement in crowdsourcing platforms. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 49–60. IEEE, 2020.

[77] A. Sheshadri and M. Lease. Square: A benchmark for research on computing crowd consensus. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 1, 2013.

[78] A. Singla, I. Bogunovic, G. Bartók, A. Karbasi, and A. Krause. On actively teaching the crowd to classify. In *NIPS Workshop on Data Driven Education*, 2013.

[79] A. Singla, I. Bogunovic, G. Bartók, A. Karbasi, and A. Krause. Near-optimally teaching the crowd to classify. In *ICML*, number 2 in 1, page 3, 2014.

[80] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, San Francisco, California, USA, 2017. AAAI Press.

[81] G. Tang, K. Wu, D. Guo, Y. Wang, and H. Wang. Alleviating low-battery anxiety of mobile users via low-power video streaming. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 998–1008, 2020.

[82] W. Tang and M. Lease. Semi-supervised consensus labeling for crowdsourcing. In *SIGIR 2011 workshop on crowdsourcing for information retrieval (CIR)*, pages 1–6, 2011.

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

[84] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[85] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[86] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, 2011.

[87] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. On truth discovery in social sensing: A maximum likelihood estimation approach. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 233–244, 2012.

[88] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang. An efficient prediction-based user recruitment for mobile crowdsensing. *IEEE Transactions on Mobile Computing*, 17(1):16–28, 2018.

[89] S. Wang, L. Su, S. Li, S. Hu, T. Amin, H. Wang, S. Yao, L. Kaplan, and T. Abdelzaher. Scalable social sensing of interdependent phenomena. In *Proceedings of the 14th international conference on information processing in sensor networks*, pages 202–213, 2015.

[90] J. Ward, K. Kapadia, E. Brush, and S. D. Salhanick. Amatoxin poisoning: case reports and review of current therapies. *Journal of Emergency Medicine*, 44(1):116–121, 2013.

[91] J. Whitehill, T.-f. Wu, J. Bergsma, J. Movellan, and P. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems*, 22:2035–2043, 2009.

[92] Wikipedia contributors. Amazon mechanical turk — Wikipedia, the free encyclopedia, 2021. [Online; accessed 10-April-2021].

[93] Q. Xu and R. Zheng. When data acquisition meets data analytics: A distributed active learning framework for optimal budgeted mobile crowdsensing. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[94] S. Yang, K. Han, Z. Zheng, S. Tang, and F. Wu. Towards personalized task matching in mobile crowdsensing via fine-grained user profiling. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2411–2419. IEEE, 2018.

[95] X. Yin, J. Goudriaan, E. A. Lantinga, J. Vos, and H. J. Spiertz. A flexible sigmoid function of determinate growth. *Annals of botany*, 91(3):361–371, 2003.

[96] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[97] X.-Y. Zhang, S. Wang, and X. Yun. Bidirectional active learning: a two-way exploration into unlabeled and labeled data set. *IEEE transactions on neural networks and learning systems*, 26(12):3034–3044, 2015.

[98] L. Zhao, G. Sukthankar, and R. Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 728–733. IEEE, 2011.

[99] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103, 2018.

[100] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pages 167–176, 2018.

[101] Y. Zhou, C. Huang, Q. Hu, J. Zhu, and Y. Tang. Personalized learning full-path recommendation model based on lstm neural networks. *Information Sciences*, 444:135–152, 2018.

[102] J. Zhu. Machine teaching for bayesian learners in the exponential family. In *Advances in Neural Information Processing Systems*, pages 1905–1913, 2013.

[103] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.