FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

INTELLIGENT DATA MINING TECHNIQUES FOR AUTOMATIC SERVICE MANAGEMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Qing Wang

2018

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Qing Wang, and entitled Intelligent Data Mining Techniques for Automatic Service Management, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Mark Alan Finlayson

Ning Xie

Debra VanderMeer

Shu-Ching Chen, Co-Major Professor

Sundaraja Sitharama Iyengar, Co-Major Professor

Date of Defense: Novermber 07, 2018

The dissertation of Qing Wang is approved.

Dean John Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of University Graduate School

Florida International University, 2018

DEDICATION

I dedicate this dissertation work to my beloved family, my parents, Bao'an Wang and Meiqin Fan, as well as my sister, Jing Wang. Without their patience, understanding, support, or love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

Time flies! At the beginning of my Ph.D., I thought it would be a long, struggled and painful period of my life. By the time that I am almost closed to finish it, I have to admit it is indeed full of stress, but also surprising short that makes me keep wondering: will I be given another chance to explore my interesting research topics with full freedom? It is the support from many people that brings me with the possibility to complete the dissertation for concluding my Ph.D. study.

First and foremost, I would like to express my most profound appreciation to my advisor Dr. S. S. Iyengar and my deceased advisor Dr. Tao Li, for their inimitable support, meticulous guidance, insightful advice, and for leading me immersed with an excellent research atmosphere. It is their great passion and endless patience that inspire me for my research career and provide me with countless support for almost any matters.

Second, I would like to extend my gratitude to my co-advisor, Dr. Shu-Ching Chen, who has not only provided valuable guidance to my research but also given me a constructive suggestion in developing my Ph.D. career. His rigorous research attitude will influence the rest of my life.

Third, my sincere thanks go to all my dissertation committee members: Dr. Mark Alan Finlayson, Dr. Ning Xie, and Dr. Debra VanderMeer, for their helpful advice, insightful comments on my dissertation research and future research career plans.

Fourth, I am thankful to my beloved mentors including Dr. Larisa Shwartz, Dr. Genady Ya. Grabarnik, Dr. Maja Vukovic and Dr. Nicholas Fuller of my three internships at IBM Research. The patient guidance, continuous encouragement and enlightenment from them let me accumulate valuable experience and establish my confidence in my Ph.D. study.

Moreover, I would like to extend my thanks to our department staff who always helps me during my doctoral study: Dr. Jason Liu, Olga Carbonell, Ariana Tagliorett-ti, Vanessa Cornwall, Lian Zhang, Eric S. Johnson, Luis Rivera, etc.

Additionally, I'm so lucky to join the Knowledge Discovery and Research Group (KDRG) where I have built up my research experience and enhanced my professional knowledge. I am very grateful to all my colleagues of KDRG including Dr. Chunqiu Zeng, Dr. Wubai Zhou, Dr. Wei Xue, Shekoofeh Mokhtari, Wentao Wang, Ramesh Baral, Xiaolong Zhu, and Boyuan Guan.

Last but not least, I would like to express my utmost gratitude to my parents, Bao'an Wang and Meiqin Fan, as well as my sister, Jing Wang. They have been and will always be my strongest support!

ABSTRACT OF THE DISSERTATION

INTELLIGENT DATA MINING TECHNIQUES FOR AUTOMATIC SERVICE MANAGEMENT

by

Qing Wang

Florida International University, 2018

Miami, Florida

Professor Sundaraja Sitharama Iyengar, Co-Major Professor

Professor Shu-Ching Chen, Co-Major Professor

Today, as more and more industries are involved in the artificial intelligence era, all business enterprises constantly explore innovative ways to expand their outreach and fulfill the high requirements from customers, with the purpose of gaining a competitive advantage in the marketplace. However, the success of a business highly relies on its IT service. Value-creating activities of a business cannot be accomplished without solid and continuous delivery of IT services especially in the increasingly intricate and specialized world. Driven by both the growing complexity of IT environments and rapidly changing business needs, service providers are urgently seeking intelligent data mining and machine learning techniques to build a cognitive "brain" in IT service management, capable of automatically understanding, reasoning and learning from operational data collected from human engineers and virtual engineers during the IT service maintenance.

The ultimate goal of IT service management optimization is to maximize the automation of IT routine procedures such as problem detection, determination, and resolution. However, to fully automate the entire IT routine procedure is still a challenging task without any human intervention. In the real IT system, both the step-wise resolution descriptions and scripted resolutions are often logged with their

corresponding problematic incidents, which typically contain abundant valuable human domain knowledge. Hence, modeling, gathering and utilizing the domain knowledge from IT system maintenance logs act as an extremely crucial role in IT service management optimization. To optimize the IT service management from the perspective of intelligent data mining techniques, three research directions are identified and considered to be greatly helpful for automatic service management: (1) efficiently extract and organize the domain knowledge from IT system maintenance logs; (2) online collect and update the existing domain knowledge by interactively recommending the possible resolutions; (3) automatically discover the latent relation among scripted resolutions and intelligently suggest proper scripted resolutions for IT problems.

My dissertation addresses these challenges mentioned above by designing and implementing a set of intelligent data-driven solutions including (1) constructing the domain knowledge base for problem resolution inference; (2) online recommending resolution in light of the explicit hierarchical resolution categories provided by domain experts; and (3) interactively recommending resolution with the latent resolution relations learned through a collaborative filtering model.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

As more and more industries are involved in the age of artificial intelligence [KDF+], it has become essential to adopt information techniques for all business enterprises to gain a competitive advantage. Business enterprises constantly explore innovative ways of expanding their outreach and fulfill high requirements from customers. However, value-creating activities cannot be accomplished without a solid and continuous delivery of IT services in this increasingly complex world. Service providers are expected to focus on assisting customers in their core business areas and resolving tough problems. Also, the time spending on fixing operational issues has to be minimized as well. Therefore, intelligent data mining and machine learning techniques urgently need employing to maximize the automation of subroutine procedures such as *problem detection*, *determination*, and *resolution* of the service infrastructure, which is an ultimate goal of IT service management, prescribed by the Information Technology Infrastructure Library (ITIL) specification [urlg]. A traditional workflow of an IT routine maintenance illustrated in Figure 1.1 includes four steps.

At the first step, anomalies are detected by the monitoring system, one of the most critical components in IT service management. Some popular monitoring systems are available in the marketplace, such as IBM Tivoli Monitoring [urlf], HP Open View [urlb], etc. A monitoring system can track the status of a system and detect various problems related to CPU utility, memory usage, network connection condition and so on. Based on the regularly collected system data, it computes the metrics situation compared with some predefined acceptable thresholds. An alert will be

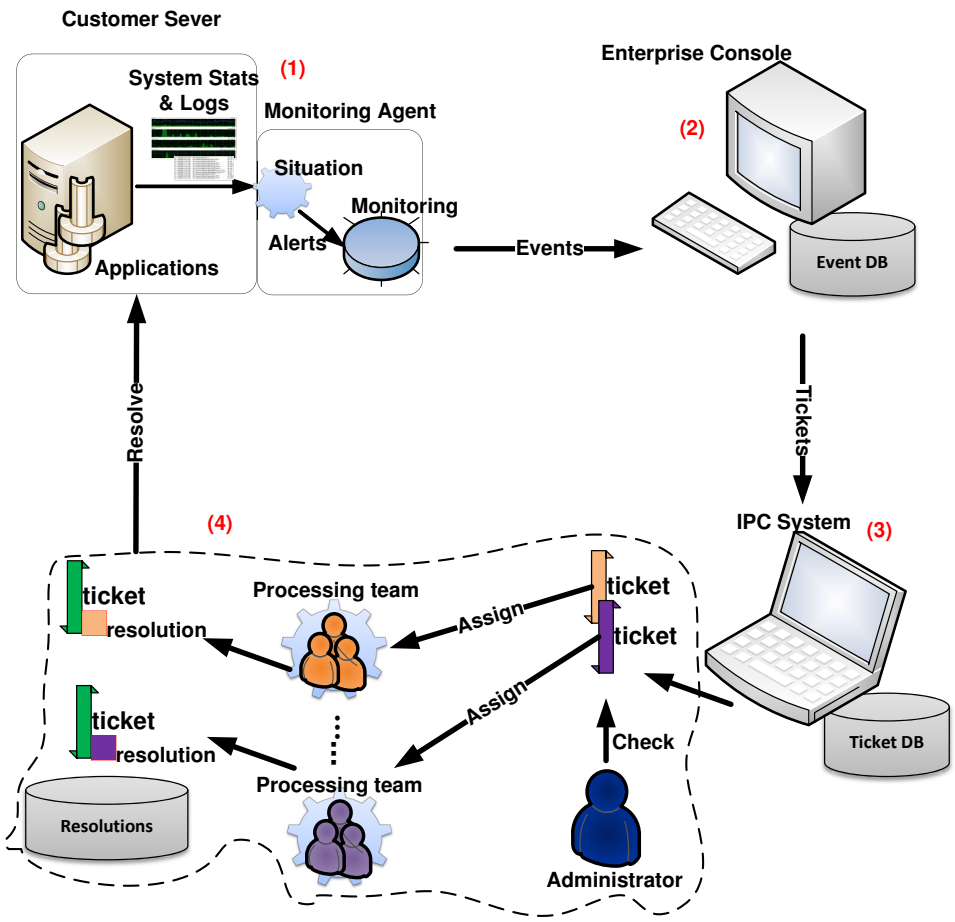

Figure 1.1: A typical workflow of a traditional IT routine maintenance.

raised if any violation occurred. When an anomaly persists beyond a predefined duration, the monitoring system generates an event for further inspection.

For the second step, events generated from an entire IT environment are consolidated in an enterprise console and finally stored into an event database. Each event consists of three parts: event type, occurring timestamp, and the description. The system determines whether or not to create an incident ticket having a business impact or a business risk, also referred to as a monitoring ticket by employing rules, cases or a knowledge-based engine to analyze these events.

At the third step, the reported incident tickets illustrated in Figure 1.2 are collected by an IPC (Incident, Problem, and Change) system. The information recorded in

TICKET IDENTIFIER:	XXXX:APPS:LogAdapter:NALAC:STARACTUAT_6600

NODE	FAILURECODE	ORIGINAL SEVERITY	OSTYPE	COMPONET	CUSTOMER
XXXX	UNKNOWN	4	WIN2K3	APPLICATION	XXXX

TICKET SUMMARY:	STARACTUAT_6600 03/01/2014 04:30:28 STARACTUAT_6600 GLACTUA Market=CAAirMiles:Report_ID=MRF600:ReportPeriod From: 2014/02/01 to 2014/02/28:ErrorDesc=For CAAirMiles Actuate is out of balance with STAR BalanceMRF600 & MRF601 Counts. Reconcilation Difference = 2MRF600 & MRF601 Net Fee. Reconcilation Difference = 25MRF600 & MRF601 Gross Fee .Reconcilation Difference = 25

RESOLUTION

ProblemSolutionText:***** Updated by GLACTUA ******
Problem Reported : Reconciliation difference Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs.
Solution provided : *Reconciliation was re-run after the next set of reports completed*.There was no user impact.
Closure code : WRKS_AS_DSIGND
RCADescription:***** Updated by GLACTUA ******
Problem Reported : Reconciliation difference
Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs.
Solution provided : Reconciliation was re-run after the next set of reports completed.There was no user impact.
Closure code : WRKS_AS_DSIGND

Figure 1.2: A ticket example in IT service management. Ticket summary logs the specific problem symptoms. A step-wised resolution written by human engineers is recorded in ticket resolution.

the ticket summary is the description of the underlying problem for further problem diagnosis, determination and resolution.

At the fourth step, human engineers start to inspect possible root causes based on the ticket summary. In general, the role of human engineers is limited to correctly inferring the possible categories (e.g., database, application, OS, etc.) of the underlying IT problem and assigning it to the corresponding processing teams for the final problem resolution. They make the complex root cause analysis and log a step-wise resolution into the historical database after they have fixed this issue. The resolution part contains considerable domain expert knowledge seen in Figure 1.2. However, human engineers are often overcharged in this workflow by the abundant incident ticket data for problem diagnosis, determination, and resolution. Developing more

intelligent and efficient solutions is required to fully automate these processes, and thus alleviate human efforts involved in IT service management.

Recent advances in artificial intelligence have led to a new revolution in traditional IT industries. For decades, service providers have already developed various cognitive techniques to optimize the automated processes. They are trying to address the customers' concerns about how they could quickly detect and fix problems in their infrastructure as soon as the issues occurred. With this purpose, human engineers begin with automatically resolving repetitive problems in an IT system. They identify the problem patterns from data and write the corresponding scripts, which could quickly fix specific issues in an automated way.

Then, enterprise IT automation services [urld] has been developed, where *virtual engineer* is incorporated as a cognitive engine for automated corrective actions (i.e., scripted resolutions or automations) and closure of incident records immediately. It includes several key components:

- Virtual engineer is a software component that uses algorithms to detect and take operational actions on problems without any human intervention.
- Patterns are identified by human engineers, which virtual engineer must follow to respond to an incident.
- Analytics is used as the gauge of the effectiveness of automation and to learn experience from new changes or root problem determination.

In this process, it logs detailed information while it executes operations. Figure 1.3 shows an example of an IT service management ticket that was automatically generated by the monitoring system, and successfully fixed by IT automation services. The summary and monitoring class (i.e., an alert key) of the incident ticket provide an initial symptom description, which is used for automation service to identify existing

automation or lack thereof. If the issue is resolved by the recommended automation (i.e., a scripted resolution), the value of "AUTORESOVLED" will be marked non-zero. If it could not be completely resolved, this ticket problem is then directly escalated to human engineers due to the lack of the corresponding automation.

ALERT_KEY	cpc_cpuutil_gntw_win_v3		AUTOMATON_NAME		CPC:WIN:GEN:R:W:System Load Handler		
OPEN_DTTM	CLIENT_ID	HOSTNAME	ORIGINAL SEVERITY	OSTYPE	COMPONET	SUBCOMP OMET	AUTO RESOVLED
2016-04-30 12:43:07	136	LEXSBWS01 VH	2	WIN	WINDOWS	CPU	1
TICKET SUMMARY	CPU Workload High. CPU 1, busy 99% time.		TICKET RESOLUTION		The CPU Utilization was quite reduced, hence closing the ticket.		

Figure 1.3: A sample ticket solved by IT automation services.

To sum up, IT service management relies on partial automation of the IT routine maintenance procedure, where operations of both human engineers and virtual engineers are closely intertwined.

1.2 Motivation and Problem Statement

Optimizing the automation of IT routine maintenance procedures can significantly alleviate the involvement of human efforts, and thereby increase the IT industry's productivity as well as efficiency, and further reduce human errors. The optimization of IT service management is urgently dictated in practice. Especially, when the systems are growing more complex, it further aggravates the difficulty in controlling the quality of delivery services and needs more human interventions in the routine maintenance procedures. As we mentioned before, artificial intelligence is dramatically improving traditional technologies of IT service management by empowering its cognitive capability. There are three critical areas outlined as follows [urla].

- **Incident or request creation.** Automated IT service management processes are designed to work well when the incident information is provided correctly. Unfortunately, many of them turn out to be ineffective due to the incorrect information provided by end-users. Incorporating AI into IT service management will enable it to interpret incidents from end-users and thereby improve the efficiency of the service management.
- **Automated backend processes.** Automated IT service management processes can detect anomalies and automatically open an incident ticket without any human intervention. The power of AI in IT service management would make the system capable of learning experience from past operations of human engineers and automated backend processes. In the future, automated IT service management processes will intelligently correct any issue, even if it is new to the system by learning and inferring with no human efforts.
- **Knowledge management.** Automated IT service management processes incorporated AI would search for answers against the trusted knowledge databases (e.g., AI cloud) automatically. An AI-enabled system will not only recommend the proper resolution to any IT problem, but also train itself using interactive feedback to optimize the provided answers adaptively.

In recent years, data mining and machine learning techniques have been involved in addressing the practical issues in the system and service management by researchers [MSGL09, ABD+07, DJL09, LPG02, ABCM09, KWI+11, TLS12, TLS+13, ZLSG15a, LZJ+17, LZZ+16, LZZ+17]. In an IT system, the operational data of both human engineers and virtual engineers stores in the historical database, which has not been well exploited yet. My dissertation focuses on proposing and implementing intelligent solutions to learn domain knowledge from these valuable data and leverage

the learned or inferred domain knowledge to facilitate the automated processes of *problem determination*, *diagnosis*, and *resolution*.

From the perspectives of intelligent data mining techniques, three research directions are identified and considered to be helpful for IT service management optimization.

1. ***Efficiently recommend problem resolution using a constructed domain knowledge base.*** A problematic incident is logged as an incident ticket and contains the ticket summary (i.e., problem description). When such tickets are resolved, the system administrators will log the step-wise resolution description, which is a free-form text but with valuable domain human knowledge. It is almost an impossible task to fully automate the entire IT service management without the help of domain experts. Therefore, modeling, gathering, and utilizing the domain knowledge during ticket resolution become increasingly crucial. However, both ticket summary and resolution contain domain-specific terms such as SLAs and RCA. Besides, they contain many typos and grammatical errors. As a result, it becomes infeasible to identify useful information using only traditional Natural Language Processing (NLP) techniques without any domain expertise. All these issues pose new challenges in constructing a domain knowledge base.

2. ***Interactively recommend the best matching automation using a hierarchical multi-armed bandit algorithm.*** IT automation services (i.e., ITAS) has been introduced into IT service management as an engine for automated corrective actions (i.e., scripted resolutions) and closure of incident records. The summary and monitoring class (i.e., an alert key) of the ticket provide an initial symptom description, which is used for automation services to identify existing automation or lack thereof. To improve the efficiency of the

recommended strategies of the automation engine, it is essential to understand how the symptoms could be mapped to the corresponding scripted resolutions. Based on preliminary studies, three key challenges are identified. The first challenge is the well-known cold start problem [SPUP02, ZWML16] making the enterprise automation engine ineffective which translates into significant human efforts. Adaptively optimizing the recommending strategies of the enterprise automation engine by utilizing the interactive feedback is the second challenge. Besides, domain experts usually define the taxonomy (i.e., hierarchy) of the IT problems explicitly. Correspondingly, the scripted resolutions (i.e., automations) also contain the underlying hierarchical problems' structure. The third challenge is how to improve the performance of a recommendation using the automation hierarchy.

3. ***Intelligently recommend proper scripted resolutions using an interactive collaborative topic regression model.*** The reality of IT environments is such that not all automations are properly set in a hierarchical structure due to the lack of sufficient information and some may fall into the unknown category. Furthermore, as a result of the imperfection of log information recording, a large number of tickets are logged with an error code only with no detailed symptom information, which becomes a significant challenge to infer a match between the ticket symptoms and the corresponding scripted resolutions. The mapping function could be naturally formalized as an interactive collaborative filtering problem.

Figure 1.4 summarizes my three research directions aiming to introduce a cognitive brain into the current automatic service management. This brain can intelligently and effectively understand, reason and learn from data collected from human engineers

and virtual engineers. In the next section, I briefly present the contributions of my dissertation along these research directions.

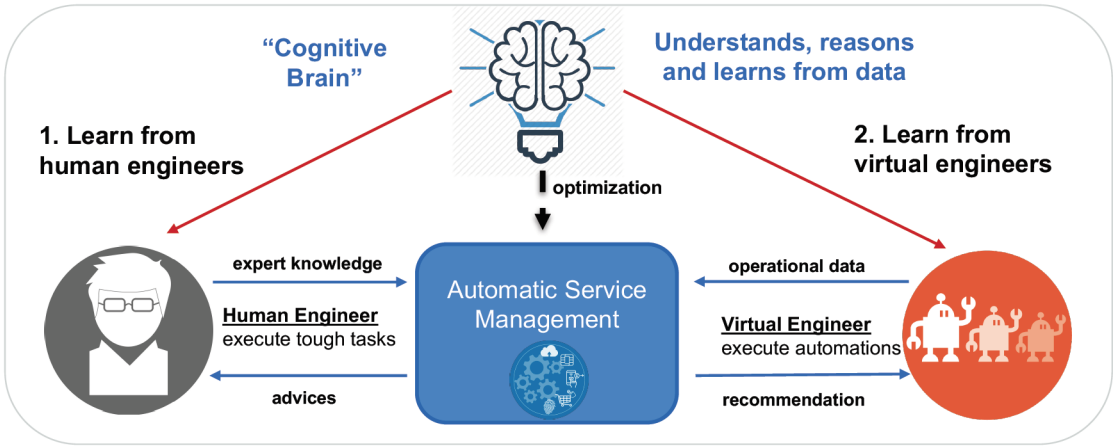

Figure 1.4: Overview of research directions.

1.3 Contributions

My dissertation addresses the challenges relevant to the research topics outlined above, by designing and developing data-driven approaches, with the purpose of helping system administrators better manage the system and alleviate the human efforts involved in IT service management. Notably, the main outcomes of my dissertation are highlighted as follows: (1) constructing the domain knowledge base for improving the performance of the resolution recommendation; (2) an online learning approach for a context-based automation recommendation; (3) an online interactive collaborative filtering model for a context-free automation recommendation. The contributions of the three research directions are carefully discussed in the reminder of Section 1.3.

1.3.1 Learn Human Intelligence by Domain Knowledge Base Construction

Since it is a tough task to fully automate the entire IT service management with no involvement of domain experts, domain knowledge base construction becomes increasingly crucial by modeling, gathering, and utilizing the domain knowledge. An integrated framework is proposed to construct a domain knowledge base for inferring the problem resolution. In order to improve the efficiency of the problem resolution process, it is crucial to formalize problem records and discover relationships between elements of the records, the records overall and other technical information. In the proposed framework, the domain knowledge is modeled using ontology techniques, of which the key contribution is a novel domain-specific approach for extracting useful phrases that makes it possible to learn from human engineers [WZZ+17b].

1. A novel domain-specific approach, designed to analyze free-form text in both ticket summary and resolution for useful phrase extraction.
2. Utilization of the ontology modeling techniques, constructing a knowledge base by combining domain expertise with extracted useful phrases.
3. Automation improvement of IT service management, through development of a resolution recommendation component based on domain knowledge.
4. A closed feedback loop system, to facilitate learning from an outcome of resolution recommendation, and thus continuous extension of the knowledge base.

1.3.2 Learn Automation Intelligence by Hierarchical Multi-armed Bandit Model

For many years, service providers have focused on developing intelligent automating processes to improve the efficiency and quality of delivered services. Recent advances in data mining and machine learning techniques have significantly powered the cognitive development of traditional IT industries. With this purpose, human engineers begin with automatically resolving repetitive problems in an IT system. They identify and determine the patterns and write scripts, making operational fixes from months to minutes and reducing human errors. In this work, we try to understand how the problem symptom could be mapped to the corresponding automation (i.e., a scripted resolution). To the best of my knowledge, it is the first work to formulate the automation recommendation of IT automation services as a contextual multi-armed bandit problem, while considering the dependencies among arms in the form of hierarchies. We develop a hierarchical multi-armed bandit model leveraging the hierarchical information, which can match the coarse-to-fine feature space of arms. [WLI+18]

The contribution mainly focuses on proposing a new hierarchical multi-armed bandit model to interactively learn the best mapping function between problem symptoms and automations. The key features of our contribution include:

1. A new online learning approach, designed to (1) solve the cold-start problem, and (2) continuously recommend an appropriate automation for the incoming ticket problem and adapt based on the feedback to improve the goodness of match between the problem and automation in IT automation services.
2. Utilization of the hierarchies, integrated into bandit algorithms to model the dependencies among arms.

3. Automation improvement of IT automation services, through development of an online recommendation for ticket resolution.

1.3.3 Learn Automation Intelligence by Interactive Collaborative Topic Regression Model

In the practical IT system, a large number of tickets are logged with an error code only and no detailed symptoms, which means no context information about ticket problem is provided. In order to effectively infer a proper automation for ticket problem as well, we propose an interactive collaborative topic regression model to solve it, which is capable of learning hidden features for ticket problems and automations, while automatically identifying automation dependencies in the form of clusters. We first provide the formulation of a general interactive recommender system used for recommending an interesting news article or a favorite movie and then a ticket automation recommendation problem. We explicitly formulate item dependencies as the clusters of arms in the bandit setting, where the arms within a single cluster share the similar latent topics. In light of topic modeling techniques, we come up with a novel generative model to generate the items from their underlying topics. Furthermore, an efficient particle-learning based online algorithm is developed for inferring both latent parameters and states of our model by taking advantage of the fully adaptive inference strategy of particle learning techniques [WLI+18, WZZ+17a].

1. An online interactive collaborative filtering mode, proposed to (1) balance the tradeoff between exploration and exploitation, and (2) interactively and continuously recommend a proper item in the context-free mode.
2. Identification of item dependencies in the form of clusters, leveraged topic modeling into bandit model to model the dependencies among arms.

3. An effective online inference algorithm using particle learning, developed to solve the generative model.
4. Continual automation services improvements, through an intelligent interactive recommendation strategy.

1.4 Summary and Roadmap

Large and complex systems with a large number of heterogeneous components are difficult to monitor, manage and maintain. Traditional approaches to system management mainly rely on the knowledge from the domain experts, where the domain-specific knowledge is used to compose operational rules, policies, and dependency models. However, those routine maintenance procedures are well known and experienced as a cumbersome, labor intensive, and error-prone processes. In the dissertation, focusing on optimizing the automation processes, we design and employ intelligent data-driven approaches applied into IT service management learning domain knowledge from human engineers and virtual engineers.

To facilitate the reading and understanding of the research problems, the organization of this dissertation is outlined as follows. First, the preliminary and related works of the three research directions above are discussed in Chapter 2. To continue, we carefully study the three research problems in Chapter 3, Chapter 4 and Chapter 5, respectively. Particularly, in Chapter 3, the domain knowledge base construction is studied, where both natural language processing and ontology modeling are utilized. In Chapter 4, we mathematically formulate ticket automation recommendations as a contextual multi-armed bandit problem, where the dependencies among arms are in the form of hierarchies. In Chapter 5, we study the interactive collaborative filtering problem about how to recommend an appropriate item using only interaction data. We propose a novel interactive collaborative topic regression model, where arm de-

pendencies are formulated using topic modeling techniques. Finally, in Chapter 6, we conclude the dissertation and discuss the future work along our research. The work was supported in part by the National Science Foundation under Grant Nos. CNS-1461926 and FIU Dissertation Year Fellowship.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

In the previous chapter, I have highlighted three concrete problems on the existing IT service management system. In order to make the system more intelligent and efficient, three popular research directions are proposed along with the corresponding solutions, trying to learn expert experience from both human engineers and virtual engineers. In this chapter, I will provide background knowledge on the research topics mentioned above and highlight the closely related state-of-art literature. Section 2.1 introduces the existing work related to the traditional automation techniques in IT service management and the domain knowledge base construction as well as the relevant techniques such as ontology modeling. Contextual multi-armed bandit algorithms and interactive recommender systems will be reviewed in Section 2.2. Section 2.3 discuss the related literature on the interactive collaborative filtering problem.

2.1 Related Work of Learn Human Intelligence by Domain Knowledge Base Construction

2.1.1 Traditional Automation techniques in IT service management

The automation of IT service management is largely achieved through service-providing facilities in combination with automation of subroutine procedures such as *problem detection*, *problem determination*, and *ticket resolution recommendation* for the service infrastructure. Automatic problem detection is typically realized by the monitoring systems, such as IBM Tivoli Monitoring [urlf] and HP OpenView [urlb].

In [XHF+09], Xu et al. developed the automated system runtime problem detection by analyzing console logs. Tang et al. [TLS+13] proposed an integrated framework to minimize the false positive and maximize the coverage for system fault detection to optimize this procedure. For problem determination, significant efforts have been put on analyzing structured logs or unstructured text fields. A hierarchical multi-label classification method [ZLSG14, ZZL+17] was proposed to classify the problem types in the monitoring IT tickets. In order to determine the root cause, authors [ZTL+14, ZL15, ZTZ+17, ZWML16] analyzed the historical events to reveal the underlying temporal causal relationship between these sequential data. Automated ticket resolution recommendation [TLSG13] is a big challenge in IT service management since it requires vast domain knowledge about the target infrastructure.

However, these studies mainly work on structured data, ignoring valuable domain-specific knowledge hidden in those unstructured text fields/logs. Therefore, we urgently need to develop an intelligent framework collecting the precious human knowledge to facilitate all the subroutine procedures. In [ZXB+17], a deep neural network ranking model was utilized for a recommendation of the best top-n matching resolutions by quantifying the quality of each historical ticket resolution. It perfectly demonstrated the domain knowledge base is not only fundamental to the understanding of the system problems but also can significantly benefit those aforementioned tasks.

2.1.2 Ontology Modeling

Over the past decades, ontology technology has become common and been moving out from the realm of Artificial Intelligence [NM+01a] to desktops of domain experts, which has been defined as the study of the categories of things that exist or may exist in some domain [S+00, KMSS03]. Ontology modeling represents domain

knowledge by specifying the classes and relations among the classes. It makes the structure of domain information sharing a common understanding and enables the domain knowledge can be reused as well. Therefore, ontology modeling has been extensively investigated by many researchers due to its effectiveness and simplicity, and applied into various research domains (e.g., knowledge management, natural language processing [MN+95], recommender system [IGFH10], etc.).

2.1.3 Knowledge Base Construction

There is a fine line where the ontology ends and the knowledge base begins [NM+01a]. After ontology modeling, the great challenge lies in the knowledge base construction (KBC), which has a long history dating back to the expert systems of the 1970s. Due to the recent advances in machine learning and artificial intelligence, KBC has become perspective again by powering the AI-based knowledge system (i.e., Google Assitant, Amazon Alexa, and Apple Siri). KBC is extremely challenging due to its goal is to extract the structure information automatically from unstructured data, involving high complex subtasks including parsing, extracting, cleaning, linking and integrations.

The authors in [DHR08] analyzed natural structured English text to construct the knowledge base. In [DLT+13], the authors proposed a framework to incrementally build, maintain, and use knowledge bases from Wikipedia semi-structured articles. Lee et al. [LKKW07] adopt an episode-based ontology construction mechanism to extract domain knowledge from text documents. However, these studies build their ontology models by taking natural language text as an input. This work focuses on mining domain-specific phrases from unstructured texts with little syntax structure and mapping them onto predefined domain knowledge classes to facilitate ontology construction.

2.2 Related Work of Learn Automation Intelligence by Hierarchial Multi-armed Bandit Model

2.2.1 Interactive Recommender Systems

In order to boost sales as well as improve users' visiting experience, many practical applications in major companies (e.g., Google, Amazon, and Netflix) provide efficient online recommendation services to help consumers deal with the overwhelming information. Most recently, interactive recommender systems (see figure 2.1) have emerged striving to promptly feed an individual with proper items (e.g., news articles, music, movies, and etc.) according to the current observable context, adaptively optimize the underlying recommendation model using the up-to-date feedback and continuously maximize his/her satisfaction in a long run [ZZW13]. To achieve this goal, it becomes a critical task for such modern recommender systems to identify the goodness of match between users' preferences and target items. However, successful personalized recommendation prediction needs adequate observations of user's behaviors to learn his/her preferences, which pose a well-known *cold-start* problem since a significant number of users/items might be completely new to the system with no consumption history at all.

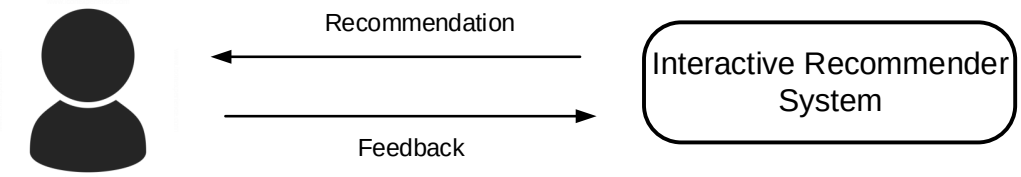

Figure 2.1: Recommendation-feedback loop in an interactive recommender system.

Existing work addresses this problem in two phrases [ZZW13]: (1) to learn the user profile using active learning [JS04, RESK15] or interview process [ZYZ11] and (2)

to make a personalized recommendation based on the established profile. However, it is still time-consuming to take interviews with users. Many users have no patience letting the system ask them a bunch of questions, especially when most of these questions are involving the information about their privacy. Before the system established user profiles, they may have already left it. Different from previous methods, multi-armed bandit algorithms [LZ07, LCLS10, CL11, TJLL14, TJL+15, ZWML16, HNL+17] can successfully deal with this dilemma without distinguishing between the two phrases and continuously learn the user's preference while recommending the best items.

2.2.2 Multi-armed Bandit Problems with Dependent Arms

Multi-armed bandits are widely adopted in diverse applications such as online advertising [ZWML16, PO07], web content optimization [ACE09, LCLS10], and robotic routing [AK08]. The core task of bandit problem is to balance the tradeoff between exploration and exploitation. A series of algorithms have been proposed to deal with this problem including ϵ-greedy[Tok10], UCB[BBG12, MRTM12], EXP3 [ACBFS02], Thompson sampling [AG13]. Contextual multi-armed bandit problem is an instance of bandit problem, where the contextual information is utilized for arm selection. Many existing multi-armed bandit algorithms have been extended to incorporating the contextual information.

In [BBG12], contextual ϵ-greedy algorithm has been introduced by extending the ϵ-greedy strategy with the consideration of context. This algorithm chooses the best arm based on current knowledge with the probability $1 - \epsilon$, while chooses one arm uniformly with the probability ϵ. Both LinUCB and LogUCB algorithms are contextual bandit models [BBG12, MRTM12] by extending the UCB algorithm with context. LinUCB [LCLS10] is proposed to do personalized recommendation on news article assuming a linear regression mapping function between the expected reward

of an arm and its corresponding context. In [MRTM12], LogUCB is proposed to deal with the contextual bandit problem based on logistic regression, where the reward is a binary value (e.g.,click or not click on an ad). Thompson sampling [CL11], one of earliest heuristics for bandit problems, belongs to the probability matching family. Its main idea is to randomly allocate the pulling chance according to the probability that an arm gives the largest expected reward given the context.

However, most prior work (e.g., LinUCB [LCLS10] and Thompson sampling [CL11]) assumes independent arms, which rarely holds in reality. Since the real-world items tend to be correlated with each other, a delicate framework [PCA07] is developed to study the bandit problem with dependent arms. Pandey et al. [PACJ07] used the taxonomy structure to exploit dependencies among the arm in the context-free bandit setting. CoFineUCB approach [YHG12] utilized a coarse-to-fine feature hierarchy to reduce the cost of exploration, where the hierarchy was estimated by a small number of existing user profiles. In our work, we study the contextual bandit problem with a given hierarchical structure of arms, where the hierarchy is constructed by domain experts based on the features of items.

2.3 Related Work of Learn Automation Intelligence by Interactive Collaborative Regression Model

2.3.1 Interactive Collaborative Filtering

In recommender systems, collaborative filtering (CF) has gained extensive popularity in recent decades due to its capability of identifying the user preference from the historical interactions between users and items [SFHS07, KBK+15, SKKR01, MS08, SM08, PBK17, WHS17]. The existing CF methods typically fall into two

primary categories: the memory-based methods [SKKR01, HKBR99] mainly make a recommendation by computing the similarity between items or users, while and the model-based methods [MS08, SM08] develop models using data mining and machine learning techniques to find patterns based on ratings. In addition, it handles the sparsity better than memory-based methods such as matrix factorization [KBV09] (MF) technologies, which project both users and items into a shared low dimension latent factor space. Our proposed approaches are related to the model-based methods.

Matrix factorization (MF), one of the model-based methods gained popularity due to the Netflix Prize and other recommendation competitions. A significant variety of MF-based methods are proposed. Probabilistic Matrix Factorization (PMF) [MS08] models the ratings as products of users' and items' latent features considering Gaussian observation noise. PMF can scale linearly with a number of observations and perform well on very sparse and imbalanced data. Bayesian Probabilistic Matrix Factorization (BPMF) [SM08] presents a fully Bayesian treatment of the PMF model with priors controlling model complexity automatically. Notwithstanding the fact that MF-based methods have been successfully applied to various recommender systems,

However, it is still an immense challenge to effectively predict preferences for new users. This challenge typically referred to as the well-known *cold-start* problem [BP17, Ahn08, SPUP02]. A straightforward solution to address this issue involves two separate stages, where it first explicitly figures out the user profile, then makes a further recommendation based on the established user profile [RAC+02, RKR08]. By contrast, some preliminary work, referred to as interactive collaborative filtering (ICF), have recently emerged as an alternative way to deal with the *cold-start* issue [ZZW13, KBK+15]. These works do not explicitly fulfill the two stages separately but formulate the recommendation problem as a multi-armed bandit problem,

and then naturally integrate the two stages together by striking a balance between exploration and exploitation. Our work is primarily relevant to this research area addressing the ICF problem.

The ICF problem is first introduced in [ZZW13], where several multi-armed bandit algorithms (e.g., Thompson sampling [CL11], UCB [BBG12]) are used for item recommendation in light of the user-item rating prediction with the probabilistic matrix factorization (PMF) framework [MS08]. However, the proposed method in [ZZW13] does not work in a completely online interactive mode since the multi-armed bandit algorithms partially rely on the latent item feature vector distributions, which are learned with the offline Gibbs sampling in advance. In [KBK$^+$15], an efficient Thompson sampling algorithm named particle Thompson sampling (PTS) addresses the ICF problem with Bayesian probabilistic matrix factorization (BPMF) [SM08] in a completely online mode. To reduce the reward prediction uncertainty, Wang et al. [WWW16] incorporated the contextual features into the learned latent feature vectors for ICF problem. However, these methods assume the latent item feature vectors in the ICF setting are independent. Although the work in [PCA07] formulates the arm dependencies as an arm clustering problem, it fails to present an efficient online method to learn arm dependencies.

By comparison, we explicitly learn the dependent arms with a generative topic model in the ICF setting and develop an efficient online solution capable of tracking the dependencies between arms as well as addressing the online recommendation.

2.3.2 Multi-armed Bandit Problems for Group Recommendation

Some recent studies explore the bandit dependencies for a group recommendation delivery by assuming that users in the same group react with similar feedback to the same recommended item [GLZ14, WWGW16, WWW17, WHLE17, STvdS16]. Most existing works utilize the contextual information for users or predefined social network to build the user dependencies. Wu et al. [WWGW16] exploit social information to find dependency among users for improving the accuracy of reward prediction. Wang et al. [WWW17] propose a context-aware collaborative bandit model, which could incorporate mutual influence among users directly for matrix completion. In [WHLE17], an interactive social recommendation model is proposed to predict the target user's preference using a weighted combination of a user's preferences and his/her friends' preferences. A context-dependent clustering of bandits algorithm [GLK+17] is investigated, where the clusters over users are based on the current item content. Our work is orthogonal to those studies since we investigate the arm (item) dependencies in a bandit model rather than the dependencies among users. Wang et al. [WLI+18] come up with a hierarchical multi-armed bandit model leveraging the explicit taxonomy information among items for online recommendation. Our proposed method is capable of instantly learning the item dependencies during the online interactive recommendation process without explicit context information provided.

It leverages topic modeling [BNJ03] to formulate arm dependencies and sequential online inference to infer the latent states and learn the unknown parameters. Popular sequential learning methods include sequential monte carlo sampling [Hal62, DDFG01] and particle learning [CJLP10, ZWW+16].

2.3.3 Sequential Online Inference

Sequential Monte Carlo (SMC) sampling consists of a set of Monte Carlo methodologies to solve the filtering problem [DGA00]. It provides a set of simulation based methods for computing the posterior distribution. These methodologies allow inference of full posterior distributions in general state space models, which may be both nonlinear and non-Gaussian. These methods approximate the distributions using a considerable number of samples (particles). As the number of particles N increases toward ∞, this converges to actual distribution. These methods allow inference of full posterior distributions in general state space models, which may be both nonlinear and non-Gaussian.

Particle filtering was first introduced in [GSS93]. Since then, they have become a very popular class of numerical methods for the solution of optimal estimation problems in non-linear non-Gaussian scenarios. It uses a genetic type mutation selection particle algorithm for the filtering equation. Particle filters implement the prediction-updating transitions of the filtering equation directly by using a genetic type mutation-selection particle algorithm. The samples from the distribution are represented by a set of particles. Each particle has a likelihood weight assigned to it that represents the probability of that particle being sampled from the probability density function. Weight disparity leading to weight collapse is a common issue encountered in these filtering algorithms. However, it can be mitigated including a resampling step before the weights become too uneven. Several adaptive resampling criteria can be used, including the variance of the weights and the relative entropy with respect to the uniform distribution. In the resampling step, the particles with negligible weights are replaced by new particles in the proximity of the particles with higher weights.

Particle learning provides state filtering, sequential parameter learning and smoothing in a general class of state space models [CJLP10]. Particle learning is used to approximate the sequence of filtering and smoothing distributions in light of parameter uncertainty for a wide class of state space models. The central idea behind particle learning is to create a particle directly from the approximation to the joint posterior distribution of states and conditional sufficient statistics of fixed parameters in a fully-adapted `resample-propagate` framework. In this paper, we leverage the idea of particle learning for both latent state inference and parameter learning.

2.4 Summary

In this chapter, I have listed all the necessary background knowledge and related work to understand the three research problems discussed in my dissertation, i.e., domain knowledge base construction, multi-armed bandit problem with dependent arms, and the interactive collaborative filtering problem.

CHAPTER 3

LEARN HUMAN INTELLIGENCE BY DOMAIN KNOWLEDGE BASE CONSTRUCTION

Recent advances in artificial intelligence have led to the renaissance of knowledge base construction (KBC) [RR18], a highly complex process involving extracting knowledge, understanding the knowledge structure, reasoning, and learning, which makes it possible for service providers to automatically expand services and continuously make service delivery better. In an IT system, abundant valuable domain human knowledge is contained in step-wise resolution descriptions, which are logged with the corresponding problematic incidents. Modeling, gathering, and utilizing the domain knowledge become increasingly crucial, since domain knowledge plays a critical role to fully automate the entire IT service management.

In this chapter, an integrated framework is proposed for a problem resolution. In order to improve the efficiency of the problem resolution process, it is crucial to formalize problem records and discover relationships between elements of the records, records overall and other technical information. In the proposed framework, the domain knowledge is modeled using ontology modeling techniques, of which the key contribution is a novel domain-specific approach for extracting useful phrases, that enables an automation improvement through a resolution recommendation utilizing the ontology modeling technique, which provides the possibility to learn domain expert knowledge from human engineers.

3.1 Introduction

3.1.1 Background

Driven by the rapid changes in the economic environment, business enterprises constantly evaluate their competitive position in the market and attempt to come up with innovative activities to gain competitive advantage. Value-creating activities cannot be accomplished without solid and continuous delivery of IT services. The increasing complexity of IT environments dictates the usage of cognitive incident management [urlc], one of the most critical processes in IT service management [urlg, LZJ+17], resolves the incident and restores the provision of services, while relying on monitoring or human intervention.

A typical workflow of IT service management is illustrated in Figure 3.1. It usually involves five steps. (1) As problems detected by a monitoring agent on a server, alerts are generated, and the monitoring emits an event if the alert persists beyond a predefined duration. (2) Events coming from an IT environment are consolidated in an enterprise console, which analyzes the monitoring events and determines whether to create an incident ticket for IT problem reporting. (3) Tickets are collected by IPC (Incident, Problem, and Change) system and stored in the ticket database [urlg]. (4) The system administrators perform the problem determination, diagnosis, and resolution based on the ticket description. The ticket resolution part of IT service delivery workflow is often a labor-intensive process. (5) In order to alleviate human efforts and maximize the automation of IT service management, the workflow incorporates an enrichment engine which in turn uses various data mining techniques to create, maintain and apply knowledge about the underlying IT system and its possible issues. This chapter focuses on the construction of the knowledge base by processing

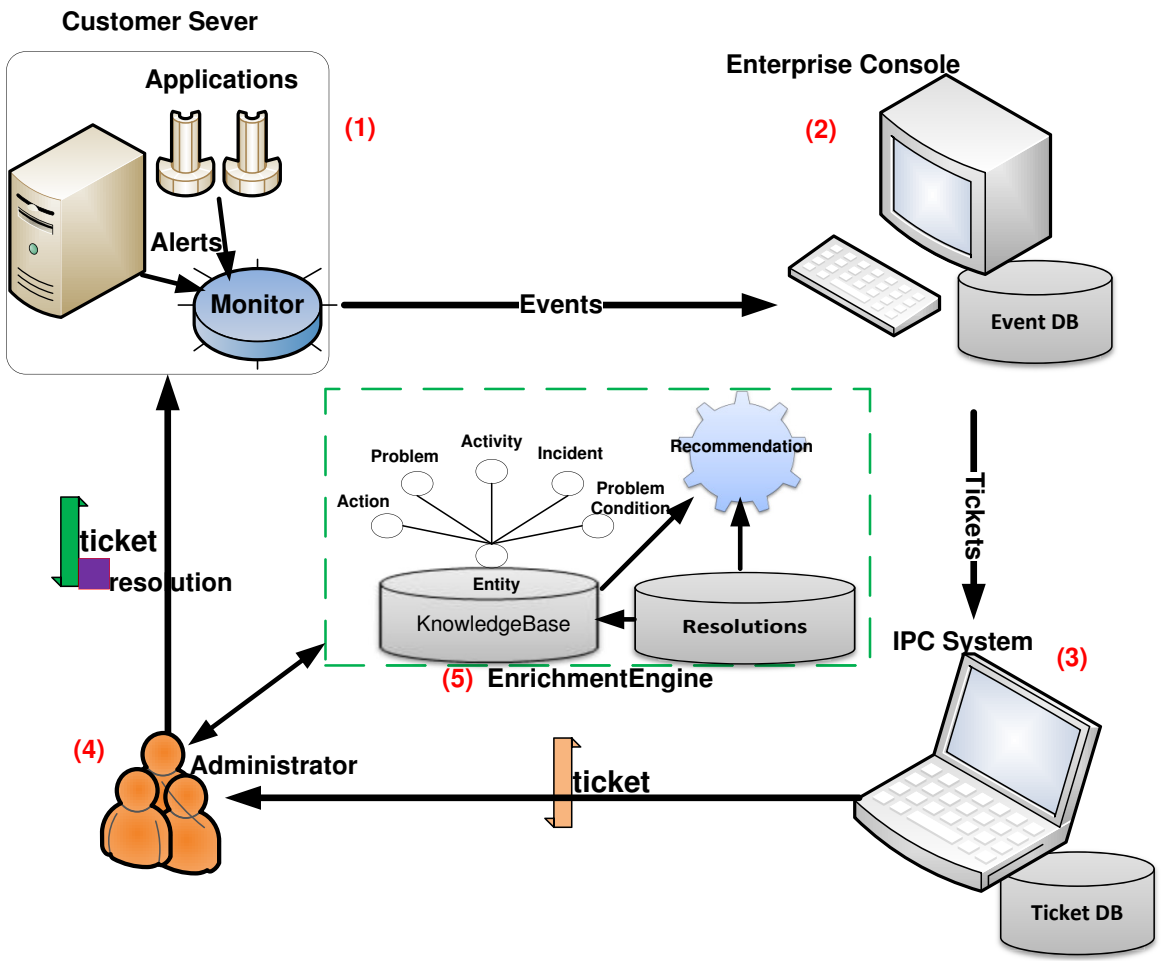

Figure 3.1: The overview of IT service management workflow.

ticketing information; it outlines an integrated solution that uses obtained knowledge to optimize problem resolution.

3.1.2 Motivation

An example of an IT service management ticket is shown in Figure 3.6. It consists of both structured fields (e.g., *OSTYPE*, *COMPONENT*) and unstructured free-form text fields (i.e., *SUMMARY* and *RESOLUTION*). Note that tickets are either generated automatically or reported by the system's user. The structured fields and the summary of a ticket provide the initial problem description for the system adminis-

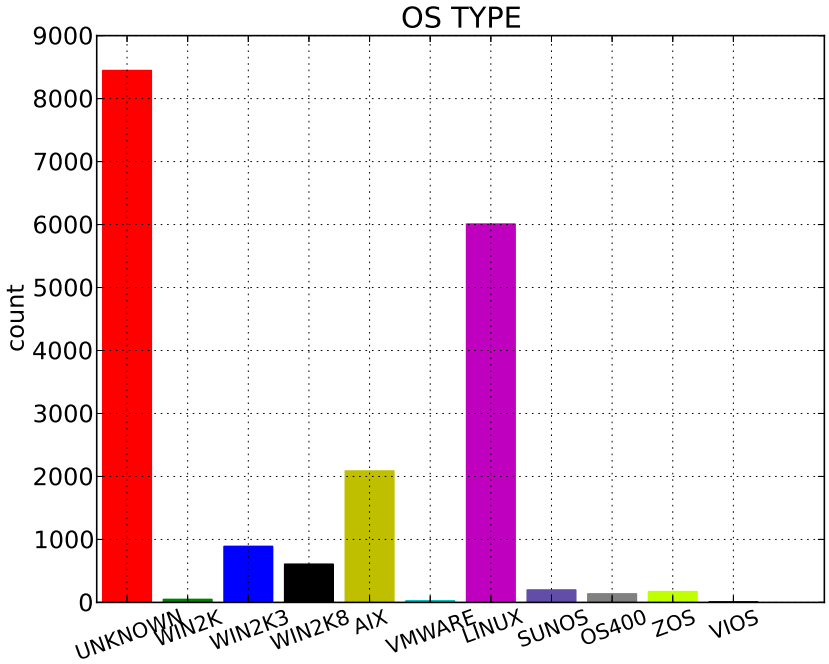

Figure 3.2: Ticket distribution across OS types.

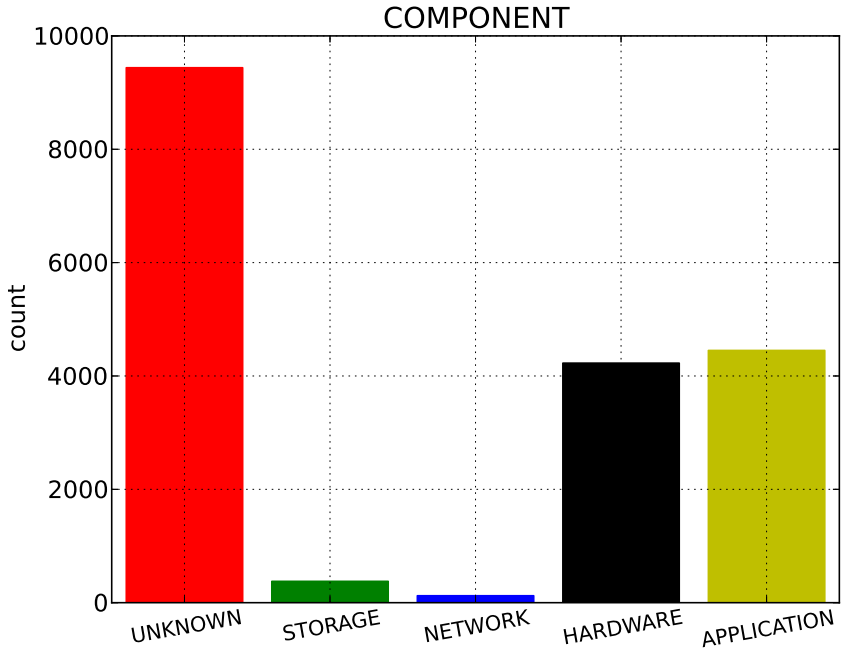

Figure 3.3: Ticket distribution across components.

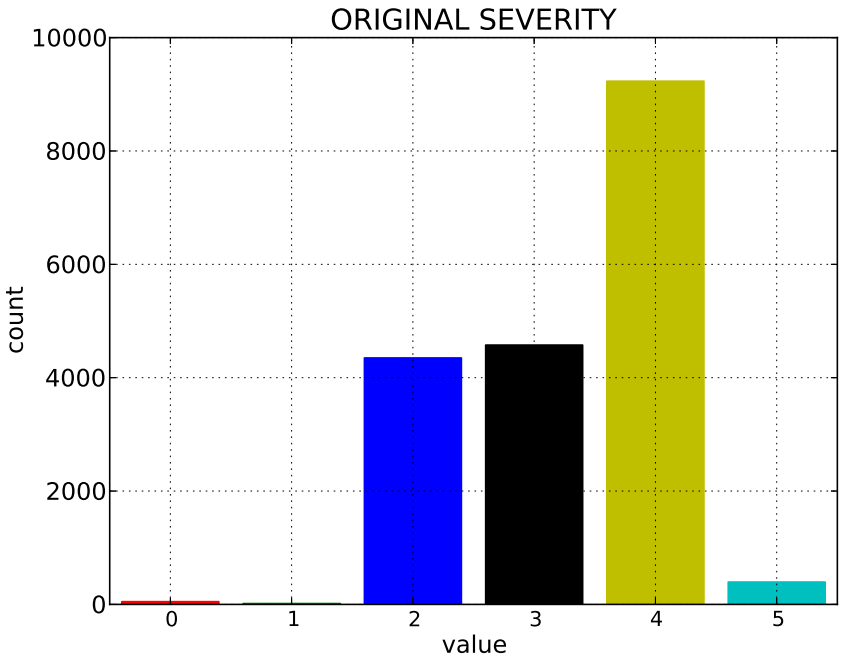

Figure 3.4: Ticket distribution across original severity.

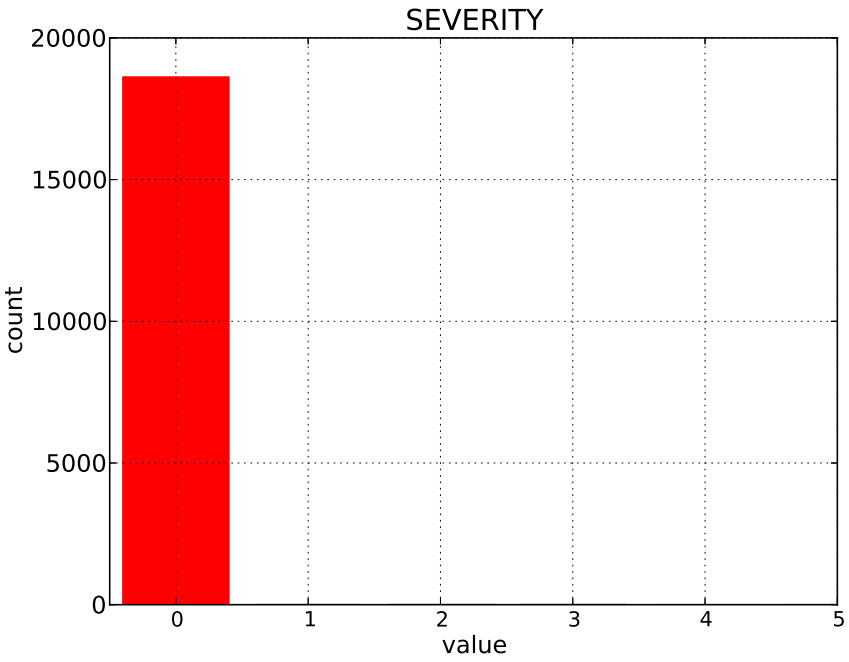

Figure 3.5: Ticket distribution across severity.

STRUCTURED

TICKET IDENTIFIER:	WPPWA544:APPS:LogAdapter:NALAC:STARACTUAT_6600

NODE	FAILURECODE	ORIGINAL SEVERITY	OSTYPE	COMPONET	CUSTOMER
WPPWA544	UNKNOWN	4	WIN2K3	APPLICATION	XXXX

UNSTRUCTURED

TICKET SUMMARY:	STARACTUAT_6600 03/01/2014 04:30:28 STARACTUAT_6600 GLACTUA Market=CAAirMiles:Report_ID=MRF600:ReportPeriod From: 2014/02/01 to 2014/02/28:ErrorDesc=For CAAirMiles Actuate is out of balance with STAR BalanceMRF600 & MRF601 Counts. Reconcilation Difference = 2MRF600 & MRF601 Net Fee. Reconcilation Difference = 25MRF600 & MRF601 Gross Fee .Reconcilation Difference = 25

UNSTRUCTURED

RESOLUTION

ProblemSolutionText:***** Updated by GLACTUA ******
Problem Reported : Reconciliation difference Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs.
Solution provided : *Reconciliation was re-run after the next set of reports completed.*There was no user impact.
Closure code : WRKS_AS_DSIGND
RCADescription:***** Updated by GLACTUA ******
Problem Reported : Reconciliation difference
Root cause : Reconciliation was run before all reports completed. This is as per the new SLAs.
Solution provided : Reconciliation was re-run after the next set of reports completed.There was no user impact.
Closure code : WRKS_AS_DSIGND

Figure 3.6: A ticket in IT service management and its corresponding resolution are given.

trators (SAs) to start ticket resolution. SAs usually record the troubleshooting steps in the resolution field as an unstructured free-form text.

In order to improve the efficiency of the problem resolution process, it is crucial to formalize the content of the ticket and, if possible, to discover a mapping between symptoms or a ticket's summary and resolutions. This is the initial motivation of our study. After a meticulous study and detailed analysis of the problem, a number of obstacles are identified.

Challenge 1 *Even in cases where the structured fields of a ticket are properly set, they either have small coverage or do not distinguish tickets well, and hence they contribute little information to the problem resolution.*

A subset of tickets are extracted from the historical ticket data set collected by IBM Global Services. Several fields such as *OSTYPE*, *COMPONENT*, and *SEVERITY* are investigated. The distributions of the field values are shown in Figures 3.2 to 3.5. As illustrated, the distributions are highly imbalanced in general. Specifically, most values of *OSTYPE* and *COMPONENT* fields are missing and labeled as *UNKNOWN*. We also observe that the field values such as *STORAGE*, *NETWORK*, *HARDWARE*, and *APPLICATION* only provide general information for problem type inference. Additionally, we provide the distributions of both original severity values generated by the monitoring and the severity values revised by human, denoted as *ORIGINAL SEVERITY* and *SEVERITY*, respectively. The severity values are considerably subjective since the two distributions of *SEVERITY* and *ORIGINAL SEVERITY* are extremely inconsistent.

Consequently, these structured fields are useful but by far not sufficient for precise problem inference. Thus we need to focus more on the free-form text fields in order to gain further insights into the underlying problem. The analysis of free-form text fields reveals the following.

Challenge 2 *The ambiguity brought by the free-form text in both ticket summary and resolution poses difficulty in problem inference, although more descriptive information is provided.*

Both ticket summary and resolution, illustrated in Figure 3.6, contain domain-specific terms such as *SLAs*, *RCA*, and *WRKS_AS_DSIGND*. In addition they contain a number of typos and grammatical errors, such as *ErrorDesc* and *ProblemSolutionText*. Moreover, some text snippets may be repeated multiple times in a single ticket and resolution. An example is shown in Figure 3.6 where phrases such as *Reconciliation Difference* and several other sentences appear in both ticket summary and resolution.

As a result, it becomes infeasible to identify useful information for problem inference using only traditional Natural Language Processing (NLP) techniques without any domain expertise. As illustrated further, our proposed integrated framework is capable of gathering domain knowledge from logs, ticketing systems, and system administrators.

Challenge 3 *IT service management and particularly problem determination, diagnosis, and resolution require a large investment of manual effort by system administrators.*

It is still a formidable task to fully automate the entire IT service management without the help of domain experts. Therefore, modeling, gathering, and utilizing the domain knowledge during ticket resolution become increasingly crucial.

In the proposed framework, the domain knowledge is modeled using ontology (see [BSW+08] for another application of ontology to IT Management) and organized into a knowledge base. In order to improve IT service management by making a number of steps toward its automation, a recommendation component leveraging the domain knowledge is explored to facilitate the ticket resolution.

3.1.3 Contribution

The contribution of our work mainly focuses on proposing and implementing an integrated framework that significantly improves the automation of IT service management. The key features of the proposed cognitive framework include:

- A novel domain-specific approach, designed to analyze free-form text in both ticket summary and resolution for useful phrase extraction.
- Utilization of the ontology modeling techniques, constructing a knowledge base by combining domain expertise with extracted useful phrases.

- Automation improvement of IT service management, through development of a resolution recommendation component based on domain knowledge.
- A closed feedback loop system, to facilitate learning from an outcome of resolution recommendation, and thus continuous extension of the knowledge base.

The effectiveness and efficiency of our framework are verified on a large data set of tickets from IBM Global Services.

The remainder of this chapter is organized as follows. The overall framework is briefly introduced in Section 3.2. The detail design and implementation of the proposed framework is provided in Section 3.3. Section 3.4 describes an extensive empirical study conducted over the real ticket data. This chapter is summarized and concluded in Section 3.5.

3.2 System Overview

Taking the aforementioned challenges into account, an integrated framework is proposed. The framework is capable of constructing a knowledge base from discovered useful phrases mined from the tickets. It also incorporates the domain knowledge provided by domain experts. The framework shows how the constructed knowledge base is used to optimize the IT service maintenance. The overall architecture of the integrated framework is illustrated in Figure 3.7. Our proposed integrated framework consists of three stages: (1) Phrase Extraction, (2) Knowledge Construction, and (3) Ticket Resolution.

The entire framework starts with the stage of Phrase Extraction. The input of Phrase Extraction is a set of the historical tickets, and the output are the useful domain knowledge phrases. The Phrase Extraction stage involves two main components: the Phrase Composition and Initial Summary Analysis component, and the Phrase

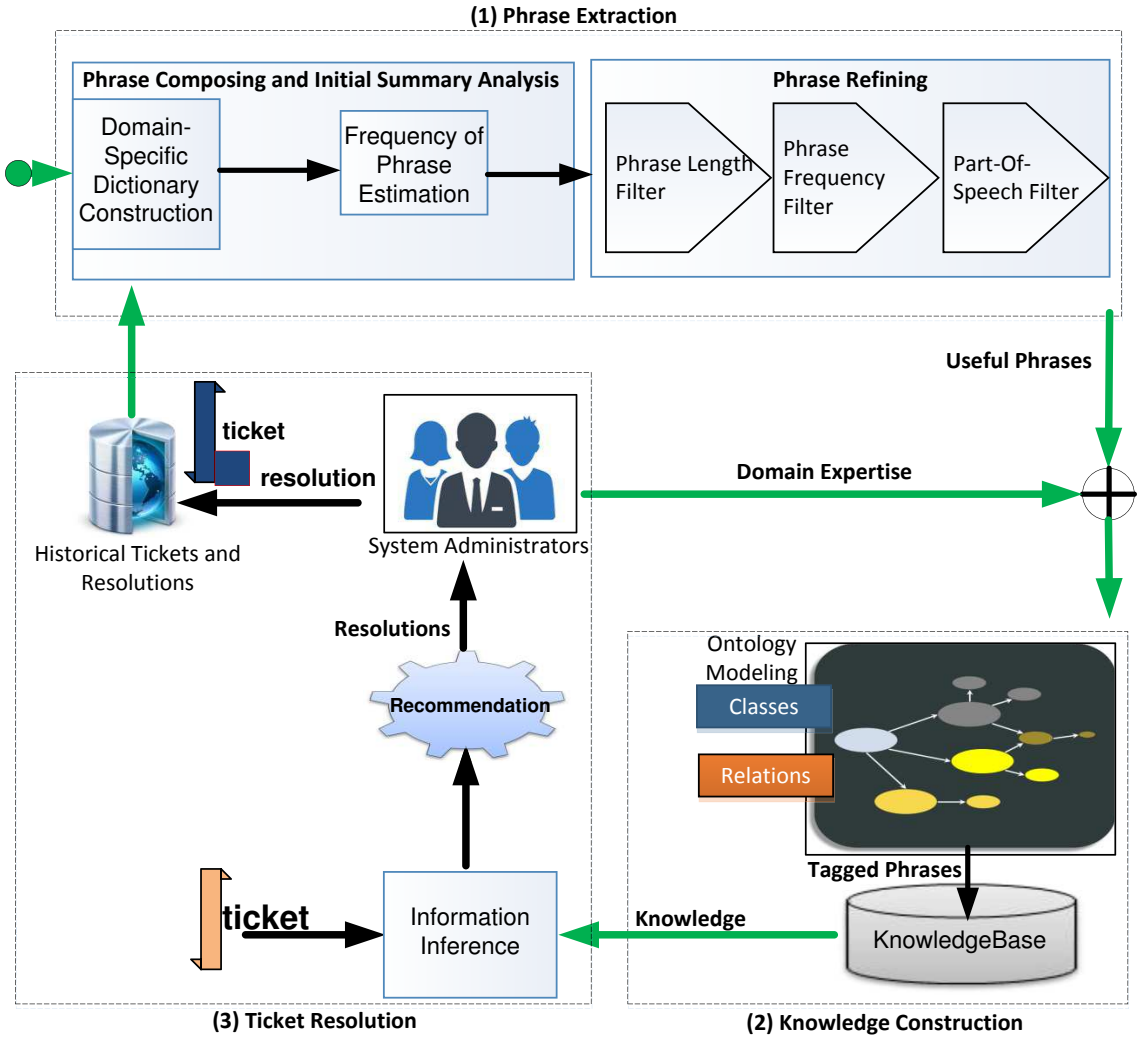

Figure 3.7: An overview of the integrated framework.

Refining component. The Phrase Composition and Initial Summary Analysis component builds phrases from the unstructured text fields of tickets and estimates the frequency for each obtained phrase. The Phrase Refining component applies filters with diverse criteria (e.g., length, frequency, etc.) to refine the extracted phrases.

In the stage of Knowledge Construction, the domain expertise (e.g., the knowledge from system administrators) is utilized for ontology modeling. As usual the ontology is composed of the classes and the relations among classes. The phrases from the Phrase Extracting stage are tagged with the classes defined in the ontology and

archived for knowledge base construction. The archived knowledge is leveraged for ticket resolving in the next stage.

The incoming tickets are resolved in the stage of Ticket Resolution. The unstructured text fields of each ticket are first tagged by the Information Inference component. Provided with the tagged ticket, the Recommendation component recommends a ranked list of the most relevant resolutions to the system administrators. The SAs can choose the most appropriate resolution. The ticket with the attached final resolution is archived into the historical ticket repository. The SAs accumulate more experience during ticket resolution. The newly obtained domain expertise can be used to enrich the knowledge base and facilitate learning. As a result, a closed feedback loop system is formed, and the knowledge base can be incrementally built.

In summary, the enriched knowledge base further facilitates the resolution recommendation, allowing the improvement of IT service management.

3.3 Design and Implementation

In this section, we explicitly describe the design and implementation for each stage.

3.3.1 Phrase Extraction Stage

This stage takes the historical tickets as input and produces useful specific domain phrases (e.g., "available disk space," "backup client connection") by analyzing the unstructured text fields. Intuitively, those phrases encompass the terms with high frequency as well as context information. To achieve this goal, we first extract frequent phrases, then filter out non-informative word combinations to keep only informative phrases. This stage consists of two main components: (1) Phrase Composing and Initial Summary Analysis, and (2) Phrase Refining.

Phrase Composing and Initial Summary Analysis

Traditionally, n-gram model is extensively applied to capture the frequently co-occurrent words in a given corpus, explored in our initial approach. However, the extraction of all possible n-grams from a large corpus is an highly time and computing power consuming task. To solve the problem, we exploit the data compression algorithm Lempel-Ziv-Welch (LZW) [Wel84] to extract the hot phrases from the massive ticket corpus.

We address two issues of LZW to achieve our goal of extracting frequent phrases as follows. First, LZW typically works at the character level, and we leverage it to the word level LZW (WLZW). Second, the algorithm only finds repeated patterns but not their frequencies.

Domain-Specific Dictionary Construction

In this part, an input text $\mathcal{T}$ = *"sql server sql server memory"* with repeated patterns is constructed to illustrate how we adopt WLZW for efficient domain-specific dictionary extraction.

Beginning with an empty dictionary, the input $\mathcal{T}$ feeds into the WLZW algorithm. We obtain a dictionary with items (e.g., *"sql,"* *"sql server"*) by reading the first two words. When WLZW reads *"sql"* again, it already exists in the dictionary. Then the algorithm continues to read the next word *"server"* and combine it with the previous word to be a new current phrase *"sql server'* as a key that also exists in the dictionary. Therefore, it keeps reading the next word *"memory"* and merges it with *"sql server,"* a new long phrase *"sql server memory"* composed and inserted into the dictionary.

The WLZW algorithm seeks the trade-off between completeness and efficiency and attempts to find the longest n-gram with a repeated prefix, indicating the importance

of the phrase. If an n-gram is not found, it adds the next word and creates an n+1-gram in the dictionary.

The analysis of the time complexity: WLZW runs in a linear time complexity of $O(n)$, where n is the length of the given text. Practically, WLZW takes less than one minute to build the domain-specific dictionary from our entire ticket resolutions.

Frequency of Phrase Estimation We use the Aho-Corasick algorithm (AC) [AC75] to locate all occurrences of keys in a dictionary built by the WLZW algorithm and to efficiently calculate the frequency of the found keywords or phrases in the given corpus. The algorithm consists of three parts:

- Build a Trie (Keyword Tree) based on the domain-specific dictionary,
- Extend the Trie into a finite state string pattern matching machine to support linear time matching,
- Fed with the given text, find all matching keywords or phrases appearing as a substring of the input text.

We provides a specific example to clarify how the AC algorithm works in our integrated framework. Assume we have a dictionary $\mathcal{D}$ comprising { *"job failed due to plc issue," "job failed due to database deadlock," "job failed due to sql error," "database connectivity," "sql server," "sql server memory"*}. Given the dictionary $\mathcal{D}$, the Aho-Corasick algorithm builds a Trie shown in Figure 3.8. The solid arrows are success transitions, while the dashed arrows are failure transitions that might lead to potentially successful matches. If matching the target word, the state of automaton transits in the direction of the arrow from the current state to the following state.

We select a real ticket resolution (e.g., *"job failed due to database connectivity"*) as the input, and demonstrate step by step how the AC algorithm finds all matching phrases from the input:

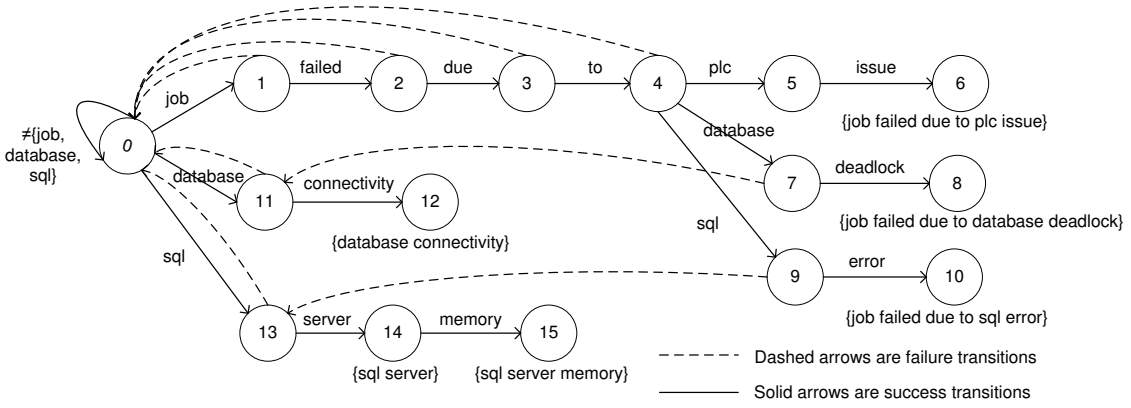

Figure 3.8: An example of a finite state string pattern matching machine.

- The automaton stays at the initial state *"State_0"* while scanning non-matching words;
- When reading the word "job," the automaton state transits from *"State_0"* to *"State_1,"* and the output of *"State_1"* is empty;
- Reading word by word, the automaton traverses success transitions (e.g., solid arrows) until it fails in *"State_7;"*
- In *"State_7,"* it transits to *"State_11"* by following a failure transition;
- With the input word "connectivity," automaton transits from *"State_11"* to *"State_12,"* and the output of *"State_12"* is *"database connectivity;"*
- As reaching the end of the word sequence, the matching substring *"database connectivity"* is output.

The analysis of the time complexity: Assume we locate occurrences of a pattern set $\mathcal{P} = \{P_1, P_2, ..., P_k\}$ in text $T[W_1, W_2, ..., W_m]$. Let $n = \sum_{i=1}^{k} |P_i|$ and z is the number of pattern occurrences in T, the AC algorithm runs in a linear time complexity of $O(n + m + z)$.

Phrase Refining

The repeated phrases have been extracted during the previous stage; however, not all of the word combinations are useful and some should be omitted from the constructed ontology. Intuitively, we should select the most frequent pattern as important. However, many of them are non-informative phrases (e.g. numbers, "no action"). We apply the following three filters to the extracted repeated phrases allowing the omission of non-informative phrases.

Phrase Length & Frequency Filters Intuitively, both length and frequency are good indicators for important phrases. Based on our experiments, we define several filtering rules for phrase length & frequent filters: (1) Length $\geq$ 10 characters; (2) Frequency ≥ 5; (3) Single-word phrases (part of a bi-gram or tri-gram); (4) containing only numbers (non-informative phrases).

With respect to the length threshold setting for Phrase Length Filter, Figure 3.9 shows that most of the useful phrases can be obtained when the length falls between 10 and 60. In practice, we keep the phrases longer than 60 as well since those long phrases indicate high frequent occurrences in the WLZW algorithm. The frequency threshold setting is validated by the system administrators considering the trade-off that lower frequency threshold can capture more informative phrases but more noises are included, while higher frequency threshold results in fewer informative phrases.

Part-Of-Speech Filter In [JK95], Justeson et al. claim that technical terms consist mostly of noun phrases containing adjectives, nouns, and occasionally prepositions. They analyze four major technical dictionaries. Subsequently, they come up with seven practical patterns defining a technical term scheme. The scheme and the corresponding Penn Treebank tagset are summarized in Table 3.1. We utilize the existing Stanford Log-linear Part-Of-Speech Tagger [TM00] to tag input phrases.

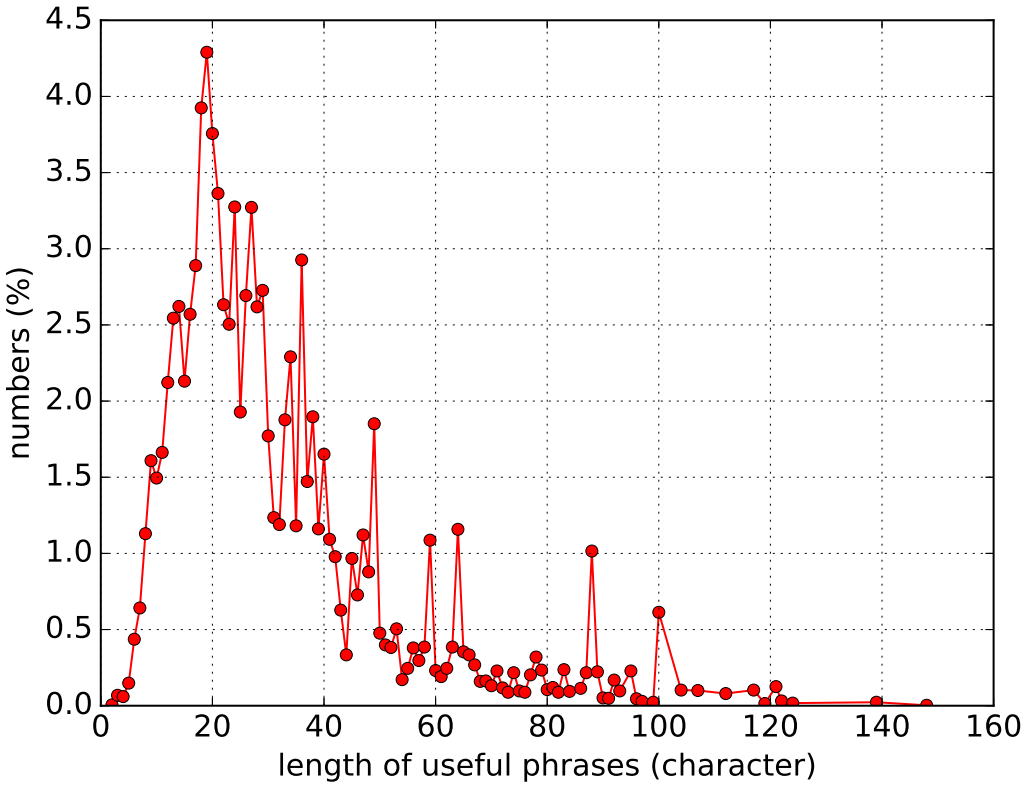

Figure 3.9: Distribution of length of useful phrases.

However, technical terms alone cannot cover all possible informative phrases since

Table 3.1: Definition of technical term's schemes.

Justeson-Katz Patterns	**Penn Treebank Entity Patterns**	**Examples in Tickets**
A N	JJ NN[P|S|PS]*	global merchant
N N	NN[P|S|PS]* NN[P|S|PS]*	database deadlock
A A N	JJ JJ NN[P|S|PS]*	available physical memory
A N N	JJ NN[P|S|PS] NN[P|S|PS]	backup client connection
N A N	NN[P|S|PS] JJ NN[P|S|PS]	load balancing activity
N N N	NN[P|S|PS] NN[P|S|PS] NN[P|S|PS]	socket connectivity error
N P N	NN[P|S|PS] IN NN[P|S|PS]	failures at sfdc
A:Adjective, **N**: Noun, **P**: Preposition		
JJ: Adjective, **NN**: singular Noun, **NNS**: plural Noun, **NNP**: singular proper Noun, **NNPS**: plural proper Noun, **IN**: Preposition		

our dictionary describes both terms and possible actions (actions may be found in the summary part of the ticket as well as in the resolution part of the ticket). We

extend the work [JK95] by including action describing domain-specific phrases - the phrases that contain verbs in different forms (e.g. past tense verb, gerund, etc.). Corresponding Penn Treebank Action Patterns are outlined in Table 3.2.

The input phrases that do not match defined patterns are eliminated.

Table 3.2: Definition of action term's schemes.

Penn Treebank Action Patterns	**Examples in Tickets**
VB[D|G|N]*	run/check, updated/corrected affecting/circumventing, given/taken
VB: base form Verb, **VBD**: past tense Verb, **VBG**: gerund Verb,**VBN**: past participle Verb,	

After applying the three filters in a pipeline, a list of candidate phrases, including entities and actions, are created for the class tagging procedure. It provides us a great benefit by reducing unqualified and unmatched potential phrases from manually unmanageable 400+K phrases to approximately 2K candidate phrases. The potential dictionary candidate phrases are ready for manual look-up by domain experts.

3.3.2 Knowledge Construction Stage

In the stage of Knowledge Construction, the SAs first develop an ontology model. This ontology model provides the semantic definition of the informative domain-specific phrases obtained during the Phrase Extracting stage.

Second, the phrases with more specific definition are tagged with the classes defined in the ontology, and finally archived for knowledge base construction. To give a concrete example, we are looking for the phrase "database deadlock" instead of just "database," since the former has more specific meaning. The archived knowledge is leveraged for the ticket resolution recommendation in the next stage.

Ontology Model

An ontology explicitly defines a common vocabulary including the formal specifications of the terms in the domain as well as the relations among them. Development of an ontology includes [NM+01b]:

- Defining classes in the ontology.
- Arranging the classes in taxonomic hierarchy.
- Defining relations amongst the classes.

Then we can construct a knowledge base by defining the instances of these classes (or facts). We build an ontology model with the help of domain experts. To verify coverage and identify capability of our ontology model, we discuss with domain experts the practical situations found in tickets and describe them in terms of the ontology's classes and relations.

Classes: a class is a deterministic concept describing a collection of objects in a given domain [NM+01b]. In our ontology model, six classes are explicitly defined in Table 3.3 to classify the important domain-specific phrases from previous stages. For example, Entity class represents all technical terms (e.g., memory fault, filesystem error). ProblemCondition class is the description of the negative state of an entity (e.g., stopped, failed).

Relations: a relation describes the interaction among the classes in our ontology model [NM+01b]. For example, the Action class can have the "TAKEN ON" interaction on Entity class, and the SupportTeam class can "WORK ON" Entity class. Note that there is no relation between Action class and Activity class. The outline of our ontology models is depicted in Figure 3.10.

Table 3.3: Classes of our ontology model.

Class	Definition	Examples
Entity	Object that can be created/destroyed/replace	memory fault; database deadlock
Action	Requires creating/destroying an entity	restart; rerun; renew
Activity	Requires interacting with an entity	check; update; clean
Incident	State known to not have a problem	false alert; false positive
ProblemCondition	Describe the condition that causes a problem	offline; abended; failed
SupportTeam	Team that works on the problem	application team; databases team

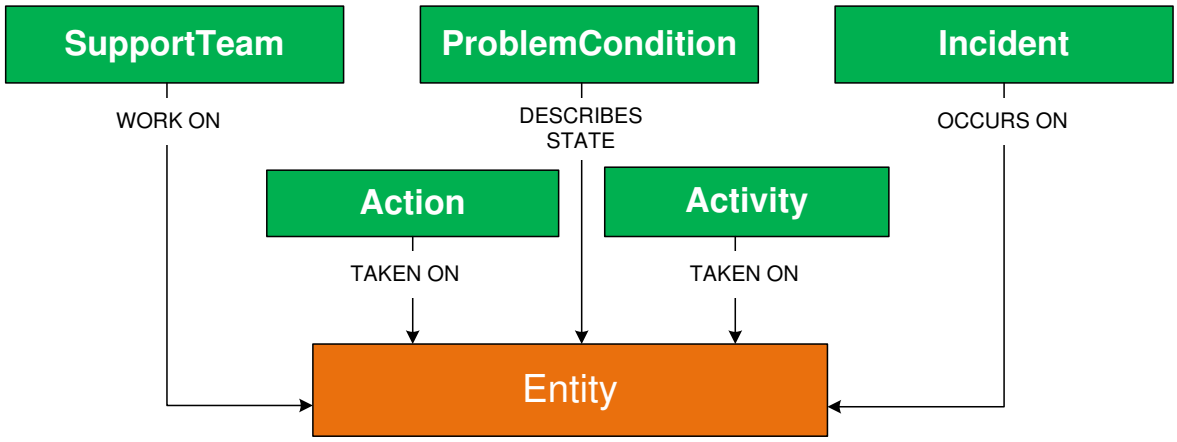

Figure 3.10: Ontology model depicting interactions amongst classes.

Knowledge Archive

Based on our ontology model, a domain expert manually tags the important keywords or phrases with their most relevant classes defined in Section 3.3.2. For example, the text snippet "certificates will be renewed" can be tagged with classes into tuples such as "[(certificates, Entity), (will, STOP WORD), (be, STOP WORD), (renewed, Action)]." Finally, we initiate our domain knowledge base with approximately 630 instances of Entity class, 240 instances of Activity class, 25 instances of Action class, 21 instances of ProblemCondition class, two instances of Incident class, and 76 instances of SupportTeam.

3.3.3 Ticket Resolution Stage

The goal of this stage is to recommend operational phrases for an incoming ticket. The incoming ticket is first processed by the Class Tagger module of Information Inference component. Taking the tagged ticket as an input, the Recommendation component provides the list of the most relevant resolutions. Finally, SAs check the recommended results. The ticket is archived into the historical ticket database, and the newly obtained domain expertise can be used to enrich the knowledge base.

Information Inference Component

The Information Inference component is used to infer problems, activities, and actions from trouble tickets by applying the constructed knowledge base and ontology model. The three key questions addressed herein are as follows: (1) how to formalize the physical words using the ontology model, (2) how to define three key concepts (e.g., problem, activity, and action) that can be extracted from the tagged ticket, (3) how to find the corresponding entity phrases associated with problem, activity or action phrases. We address them as follows:

Class Tagger Module The Class Tagger module is an index tool based on our domain knowledge base. Taking the ticket resolutions and knowledge base as the input, it outputs tagged domain keywords or phrases with the corresponding classes. The module has three steps for tagging: (1) tokenize the input into sentences; (2) construct a Trie by using ontology domain dictionary; (3) find the longest matching phrases of each sentence using the Trie and knowledge base, then map them onto the corresponding ontology classes.

For example, *"database," "deadlock,"* and *"database deadlock"* are all valid domain phrases of Entity class. But the Class Tagger module only tags the *"database*

deadlock" as Entity in a given sentence. An example of tagged ticket by the Class Tagger module is shown in Figure 3.11.

(post loading)/(Entity) (failed)/(ProblemCondition) due to (plc issue)/(Entity). (updated)/(Activity) the (gft)/(Entity) after (proper validation)/(Entity) and (processed)/(Activity) the (job)/(Entity) and (completed)/(Action) successfully.

Figure 3.11: Ticket tagged by the Class Tagger module.

Defined Concept Patterns for Inference We first define three key concepts as follows:

Problem: describes an entity in negative condition or state.

Activity: denotes the diagnostic steps on an entity.

Action: represents the fixing operation on an entity.

Using Class Tagger we obtain a total of 672+K tagged ticket resolutions and find some concept patterns in the structured corpus. For instance, ProblemCondition/Action keywords and their corresponding entities always appear in a single sentence. The structure of concepts is identified manually as shown in Table 3.4.

Table 3.4: Defined concept patterns for inference.

Concept	Pattern	Examples
Problem	Entity preceded/succeeded by ProblemCondition	(jvm) is (down)
Activity	Entity preceded/succeeded by Activity	(check) the (gft record count)
Action	Entity preceded/succeeded by Action	(restart) the (database)

Problem, Activity, and Action Extraction The derived concepts provide their patterns for information inference extraction. First, the Class Tagger module tokenizes the input into sentences and outputs a list of tagged phrases. Second, we

decide whether it is an informative snippet or not by checking if it exists in a ProblemCondition/Action list. Once ProblemConditon/Action phrase is matched in the sentence, the phrase is appended to the dictionary as a key, and all its related entities are added as the corresponding values via a neighborhood search. Each of the three key concepts has its own dictionary. Finally, we obtain the problem, activity, and action inferences. For instance, given the tagged snippet in Figure 3.11, the output is as follows:

- Problem - {failed: plc issue, post loading}
- Activity - {update: gft, proper validation; process: job}
- Action - {complete: job}

Ontology-based Resolution Recommendation Component

In our prior work [TLSG13] for automatic problem resolution, we propose a KNN-based algorithm in which the resolutions of historical tickets with top summary similarity scores to the incoming ticket summary are recommended. We use the Jaccard similarity function [SM86] to calculate the summary similarity score after tokenizing each summary into a bag of words.

Typically, Jaccard similarity function ignores the semantic information on ticket summaries. In our application, the ticket summary and resolution are highly noisy, which makes the Jaccard similarity function inappropriate. Table 3.5 shows two ticket summaries describing the same issue "database save failed." However, a low Jaccard similarity score here is due to many non-informative words.

Two extended works [ZTL$^+$15, ZLSG15b] adopt several techniques trying to alleviate the issue by grouping words into semantic topics or mapping semantically similar words closely in the same vector space. Those approaches, however, only deal with semantically similar words without handling the noise caused by the non-informative

words. Fortunately, the ontology model we constructed greatly facilitates our resolution recommendation task, as it essentially enhances our semantic understanding of the tickets and de-noises tickets by filtering the non-informative words out of the textual attributes. De-noising improves similarity allowing tokenized Jaccard similarity function to concentrate only on informative phrases.

Table 3.5: Noisy ticket summary examples.

Inside ProcessTransaction. DetermineOutcome failed. Database save failed: Tried an insert, then tried an update
CRPE3I1Server Database save failed on XXX 00:19:46 XXX /logs/websphere/wsfpp1lppwa 899CRPE3I1Server/SystemOut.log [3/20/14 0:19:33:371 MST] 0000002b SystemOut 20140320 00:19:33, 371 [WebContainer:30] [STANDARD] [DI_US:01.22] (ng.AEXP_US_ISR_Work_Txn.Action) FATAL XXX—10.16.4.4—SOAP—AEXP_US_ISR_Roads3_Pkg —AEXPUSISRWorkInquiry—ProcessInquiry

Ontology Construction in ticket summary

Ontology construction in ticket summary follows the same steps as in ticket resolution. But ticket summary delivers the problem symptoms instead of the problem resolution information. It is reasonable to assume that only problem and activity phrases present in ticket summary. Extracted activity phrases describe automatically triggered system actions such as "rerun," "restart," and so on. According to the assumption, only three types of knowledge phrases, i.e., Entity, Activity and Problem Condition, are recognized during the manual tag process.

Tokenized Similarity function

Once we extract problems from ticket summary using concept patterns of Table 3.4, the Jaccard similarity function is applied to the extracted Problem phrases. After removing the non-informative phrases in ticket summary from the process of similarity calculation, the same methodology is adopted for ticket resolution recommendation

as in the work [TLSG13]. A case study given in Section 3.4 illustrates that the revised similarity function can better capture the similarity between ticket summaries.

3.4 Experiment

In this section, we present the experimental dataset, the running environment, and the discussion of experimental results.

3.4.1 Data and Setup

Experimental tickets are collected from real production servers of the IBM Tivoli Monitoring system [urlf]. The data set covers three month time period containing $|\mathcal{D}| = 22,423$ tickets with 33 attributes corresponding to the columns of tickets table. Our integrated system is designed to compliment monitoring systems such as the IBM Tivoli Monitoring system and to automate delivery of an IT service management. The component is implemented in Java 1.8, and tested on 64-bit Windows 8.1 Enterprise residing on a machine equipped with Intel Core 2 Xeon CPU 3.4GHz and 16GB of RAM.

3.4.2 Evaluation Metrics and Evaluation Overview

Four commonly used evaluation metrics are applied in our evaluation. Let TP, TN, FP, and FN correspond to true positive, true negative, false positive, and false negative, respectively. Accuracy is computed as $\frac{TP+TN}{TP+TN+FP+FN}$. Precision is defined as $\frac{TP}{TP+FP}$, recall is defined as $\frac{TP}{TP+FN}$. The F1 score is computed as $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$.

To evaluate our integrated system, we randomly split our dataset into training and testing dataset. The training set, 90% of the entire ticket dataset, is used to build the knowledge base through our system, while the remaining are used for testing. To

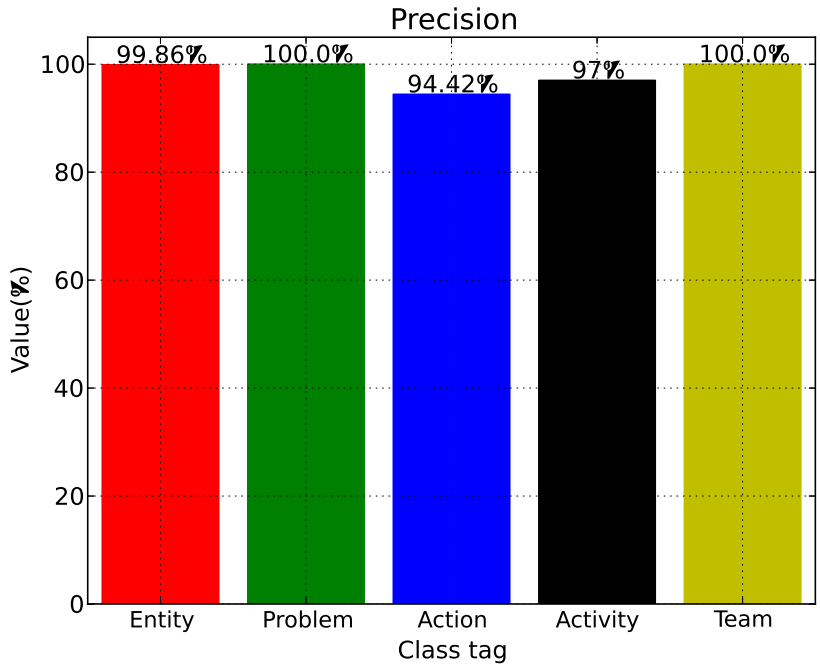

Figure 3.12: Evaluation of our integrated system in terms of precision.

build the ground truth, domain experts manually find and tag all phrase instances into six classes defined in Table 3.3. Class Tagger is applied to the testing tickets to produce tagged phrases with predefined classes. Comparing the tagged phrases with the ground truth, we obtain the performance evaluation shown in Figures 3.12 to 3.15.

The precision, recall, F1 score, and accuracy for ProblemCondition are close to 1 due to the small number of instances (e.g., failed, occurred, expired, unavailable, etc.). We also observe the precision of Entity, Action, and Activity extraction is 99.86%, 94.42%, and 97%, recall is 88.73%, 95.12%, and 93%, F1 score is 93.97%, 94.77%, and 95.1%, and accuracy is 97.05%, 97.72%, and 99.3%, respectively. The reason is that the classes of Entity, Action, and Activity contain a large amount of instances in typos and various verb forms. The Incident class is observed with similar results with ProblemCondition class, though its performance is not illustrated explicitly herein.

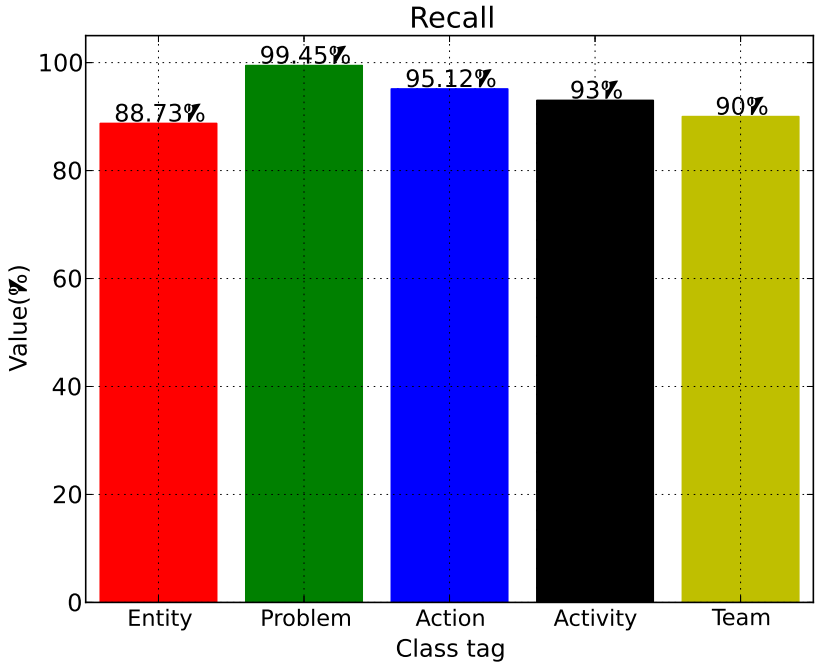

Figure 3.13: Evaluation of our integrated system in terms of recall.

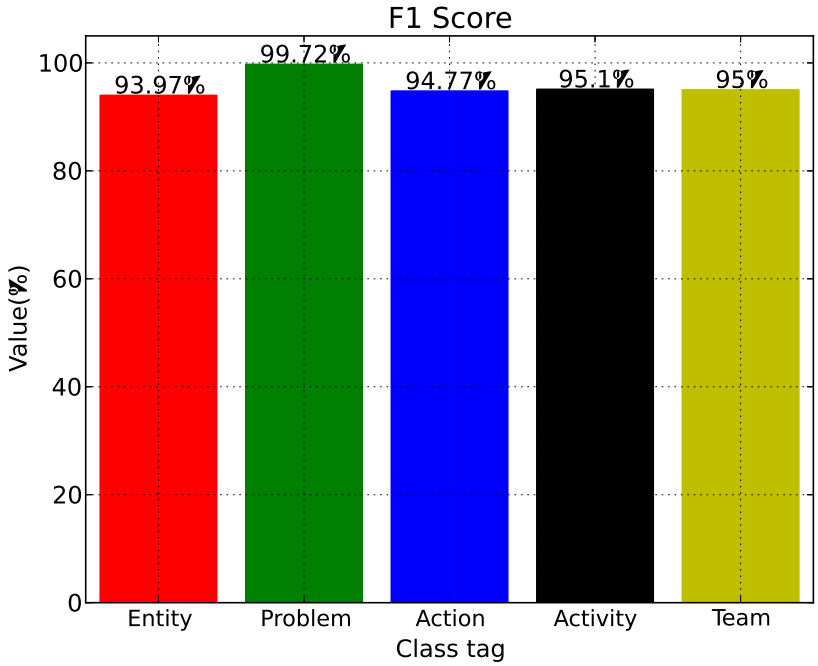

Figure 3.14: Evaluation of our integrated system in terms of f1-score.

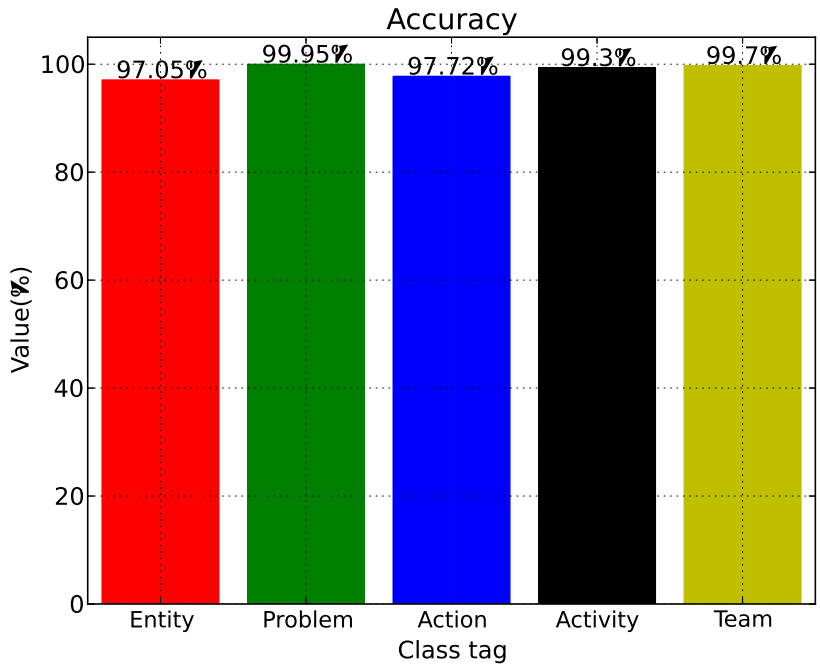

Figure 3.15: Evaluation of our integrated system in terms of accuracy.

3.4.3 Evaluating Information Inference

We also evaluate the usability and readability of our automated information inference results and compare them with traditional methods of manually analyzed tickets.

For the usability, we evaluate the extracting accuracy for concepts, i.e., Problem, Activity, and Action. Similarly, we tag the ground truth from the testing tickets and then compare it with the result tagged by Information Inference component. We evaluate the average accuracy to be 95.5%, 92.3%, and 86.2% for Problem, Activity, and Action respectively.

To evaluate readability, we focus on measuring the time-cost difference to understand a ticket with and without the Information Inference component. First, 50 tickets are randomly selected from the testing tickets and two domain experts are invited for the task of Problem, Activity, and Action identification. Then, one domain expert is required to execute the task by inspecting these tickets directly, while the

other domain expert is presented with the same task utilizing the output from the Information Inference component. We observe a significant decrease in time cost to accomplish the task from around 1000s to 100s totally.

3.4.4 Case Study: Resolution Recommendation Task

In this section, we describe a case study of our experimental results and provide the insights learned during the domain experts' manual review process.

Table 3.6: Case study for testing ticket summary "Patrol Agent is not running".

Similarity function	Top most similar summary	Associated resolutions
word level	**A1:** Patrol Agent is not running. Problem - {not running: patrol agent}	Server's uptime indicates server was unavailable. Server is available now and patrol agent connectivity present
	A2: The zpdc process is not running Problem - {not running: zpdc process}	Downstream of DB crash
	A3: The syslogd process is not running Problem - {not running: syslogd process}	No actions taken, the process is running as expected on server according to System Operations Procedures
problem level	**B1:** Patrol Agent is not running Problem - {not running: patroal agent}	Server's uptime indicates server was unavailable. Server is available now, patrol agent connectivity present
	B2: Patrol Agent Offline: Failed to reconnect to Patrol Agent on host WWPP, port 3181. Will retry in 3 timer ticks. Problem - {offline: patroal agent}	Verified connectivity. Patrol Agent connectivity test failed.
	B3: The zpdc process is not running Problem - {not running: zpdc process}	Downstream of DB crash

For the accurate evaluation one needs to fully understand the semantics of the ticket summary and resolution. That's the reason why the manual review of the recommended results by domain experts is conducted.

The recommendation is achieved based on the similarity score which can be computed by both the word level and the problem level Jaccard similarity functions shown in Table 3.6. The word level Jaccard similarity function takes the whole textual value of the ticket summary into account for similarity score computing, while the problem level Jaccard similarity function, utilizing the knowledge base constructed in our work, takes only the Problem phrases into account to obtain the similarity score.

To illustrate the difference between the two similarity functions, our task is to recommend the resolution for the ticket with summary *"Patrol Agent is not running,"* which indicates Problem *"not running: patrol agent."* As a fact confirmed by domain experts, Problem *"not running: patrol agent"* is the same as Problem *"offline: patroal agent"* occurring in **B2**, but different from Problem *"not running: zpdc process"* associated with **A2**. However, shown by Table 3.6, the recommended result based on the word level Jaccard similarity contradicts with the fact. By contrast, the recommended result according to the problem level Jaccard similarity presents the consistency with the domain expertise.

By further investigating our case study, since the entity "patrol agent" mismatches "zpdc process," domain experts assert that the resolution for the later problem contributes little to resolve the previous one. However, if the two entities are similar, such as "zpdc process" and "syslogd process," in the perspective of concept, the resolutions for the entity "zpdc process" might also apply to the entity "syslogd process."

3.5 Summary

In this chapter, we study the research problem of constructing a domain-specific knowledge base using a large number of historical tickets in the IT system. An integrated cognitive computing framework is proposed supporting incremental knowledge extraction and ontology construction. We first address the issues of efficient extraction and identification of the domain-specific phrases from noisy unstructured text fields in tickets and then construct the knowledge base with the guidance of domain experts. We conduct an empirical study that leverages a constructed knowledge base to generate ticket resolution recommendations. Our encouraging results show the effectiveness and efficiency of our integrated framework as applied to the task, and also the scalability to other critical tasks in IT service management.

CHAPTER 4

LEARN AUTOMATION INTELLIGENCE BY HIERARCHICAL MULTI-ARMED BANDIT MODEL

The increasing complexity of IT environments urgently requires the use of analytical approaches and automated problem resolution for more efficient delivery of IT services. With the purpose of automatically resolving repetitive problems in the IT system, the problem patterns are identified, and the corresponding scripted resolutions (i.e., automations) are written by human engineers. These automations enable the system automatically solve these repetitively happened issues, making the operational fixing time from months to minutes and reducing human errors. However, a traditional automation system still needs human interaction when an unexpected event occurred, which is a challenge must be solved.

In this chapter, we model the automation recommendation procedure of IT automation services as a contextual bandit problem, where arms are dependent in the form of hierarchies. Intuitively, different automations in IT automation services, designed to automatically solve the corresponding ticket problems, can be organized into a hierarchy by domain experts according to the types of ticket problems. We introduce novel hierarchical multi-armed bandit algorithms leveraging the hierarchies, which can match the coarse-to-fine feature space of arms. Empirical experiments on a real large-scale ticket dataset have demonstrated substantial improvements over the conventional bandit algorithms. Also, a case study of dealing with the well-known cold-start problem is conducted to show the merits of our proposed model clearly.

4.1 Introduction

4.1.1 Background

Facing the rapid changes in the economic environment, business enterprises constantly evaluate their competitive position in the market and attempt to come up with innovative ways to gain a competitive advantage. Value-creating activities cannot be accomplished without stable and continuous delivery of IT services. The growing complexity of IT environments dictates an extensive use of analytics combined with automation. Incident management is one of the most critical processes in IT service management as it resolves incidents and restores provisioned services.

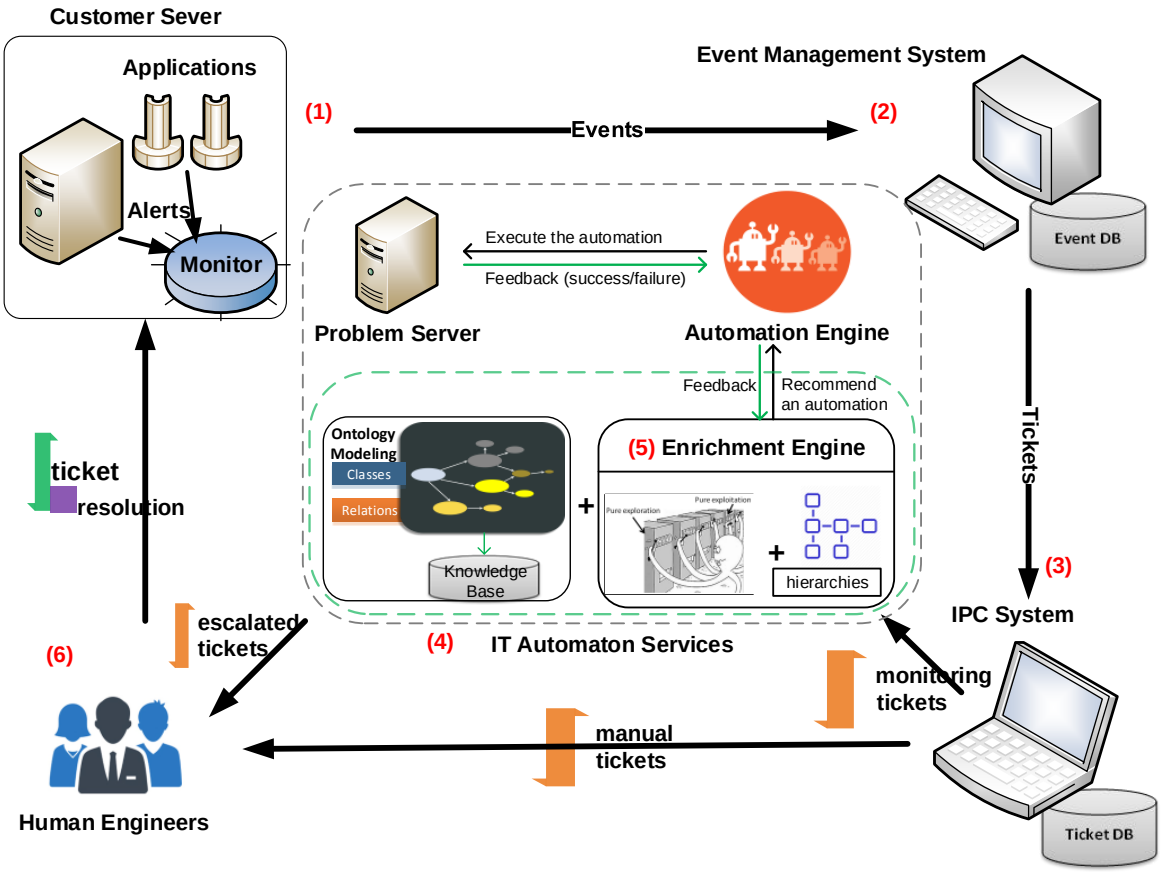

Figure 4.1: The overview of ITAS-integrated IT service management workflow.

A typical workflow of IT service management is illustrated in Figure 4.1. It usually includes six steps: (1) As an anomaly detected, an event is generated and the monitoring system emits the event if it persists beyond a predefined duration. (2) Events from an entire IT environment are consolidated in an enterprise event management system, which upon results of quick analysis, determines whether to create an alert and subsequently an incident ticket. (3) Tickets are collected by an IPC (Incident, Problem, and Change) system [ZLSG14]. (4) A monitoring ticket, identified by IT automation services for potential automation (i.e., scripted resolution) based on the ticket description. In case the issue could not be completely resolved, this ticket is then escalated to human engineers. (5) In order to improve the performance of IT automation services and reduce human efforts for escalated tickets, the workflow incorporates an enrichment engine that uses data mining techniques (e.g., classification and clustering) for continuous enhancement of IT automation services. Additionally, the information is added to a knowledge base, which is used by the IT automation services as well as in resolution recommendation for tickets escalated to a human. (6) Manually created and escalated tickets are forwarded to human engineers for problem determination, diagnosis, and resolution, which is a very labor-intensive process.

4.1.2 Motivation

In today's economic climate, IT service provider is expected to focus on innovation and assisting customers in their core business areas. Time the experts spend on fixing operational issues has to be minimized. With the increasing complexity and scalability of IT service, it has become an urgent challenge to fix operational issues regardless of the problem severity. Even the simplest issues, such as user password expiration or a CPU becoming high because of a particular process, can take several hours to be identified and fixed, and as a result, can severely cause degraded per-

formance. In order to solve these problems before they become critical, enterprise IT Automation Services [urle] has been introduced into IT service management as an engine for automated corrective actions (i.e., scripted resolutions) and closure of incident records.

Figure 4.2 shows an example of an IT service management ticket that was automatically generated by a monitoring system, and successfully fixed by the automation engine. The summary and monitoring class (i.e., an alert key) of the ticket provide an initial symptom description, which is used for automation service to identify existing automation or lack thereof. If the problem is resolved by the recommended automation, the value of "AUTORESOVLED" will be marked non-zero. To improve the efficiency of the recommending strategies of the automation engine, it is essential to understand how the symptoms could be mapped to the corresponding scripted resolutions. This is the initial motivation for our study. Based on preliminary studies [ZLSG14, WZZ+17b, ZXB+17], we have identified three key challenges in virtual engineering technology.

ALERT_KEY	cpc_cpuutil_gntw_win_v3		**AUTOMATON_NAME**		CPC:WIN:GEN:R:W:System Load Handler		
OPEN_DTTM	**CLIENT_ID**	**HOSTNAME**	**ORIGINAL SEVERITY**	**OSTYPE**	**COMPONET**	**SUBCOMP OMET**	**AUTO RESOVLED**
2016-04-30 12:43:07	136	LEXSBWS01 VH	2	WIN	WINDOWS	CPU	1
TICKET SUMMARY	CPU Workload High. CPU 1, busy 99% time.		**TICKET RESOLUTION**		The CPU Utilization was quite reduced, hence closing the ticket.		

Figure 4.2: A sample ticket is logged in IT service management with its corresponding automaton.

Challenge 1 *How do we appropriately solve the well-known cold-start problem in IT automation services?*

Most recommender systems suffer from a cold-start problem. This problem is critical since every system could encounter a significant number of users/items that are com-

pletely new to the system with no historical records at all. The cold-start problem makes recommender systems ineffective unless additional information about users/items is collected [SPUP02, CZC+15], which is a crucial problem for automation engine as well, since it cannot make any effective recommendation that translates into significant human efforts. Multi-armed bandit algorithm can address the cold-start problem, which balances the tradeoff between *exploration* and *exploitation*, hence, maximizing the opportunity for fixing the tickets, while gathering new information for improving the goodness of the ticket and automation matching.

Challenge 2 *How do we utilize the interactive feedback to adaptively optimize the recommending strategies of the enterprise automation engine to enable a quick problem determination by IT automation services?*

The automation engine (see Figure 4.1) automatically takes action based on the contextual information of the ticket and observes the execution feedback (e.g., success or failure) from the problem server. The current strategies of the automation engine do not take advantage of these interactive information for continuous improvement. Based on the aforementioned discussion, we present an online learning problem of recommending an appropriate automation and constantly adapting the up-to-date feedback given the context of the incoming ticket. This can be naturally modeled as a contextual multi-armed bandit problem, which has been widely applied into various interactive recommender systems [LCLS10, ZZW13, ZWML16]. To the best of authors' knowledge, it is the first study to formulate the strategies of the automation recommendation in IT automation services as a contextual bandit problem.

Challenge 3 *How do we efficiently improve the performance of recommendation using the automation hierarchies of IT automation services?*

Domain experts usually define the taxonomy (i.e., hierarchy) of the IT problems explicitly (see Figure 4.3). Correspondingly, the scripted resolutions (i.e., automations) also contain the underlying hierarchical problems' structure.

For example, a ticket is generated due to a failure of the DB2 database. The root cause may be database deadlock, high usage or other issues. Intuitively, if the problem was initially categorized as a database problem, the automated ticket resolutions have a much higher probability to fix this problem, than if it hasn't been categorized as such, and all other categories (e.g., file system and networking) are now taken into consideration. We formulate this as a contextual bandit problem with dependent arms organized hierarchically, which can match the feature spaces from a coarse level first, and then be refined to the next lower level of taxonomy. The existing bandit algorithms can only explore the flat feature spaces by assuming the arms are independent.

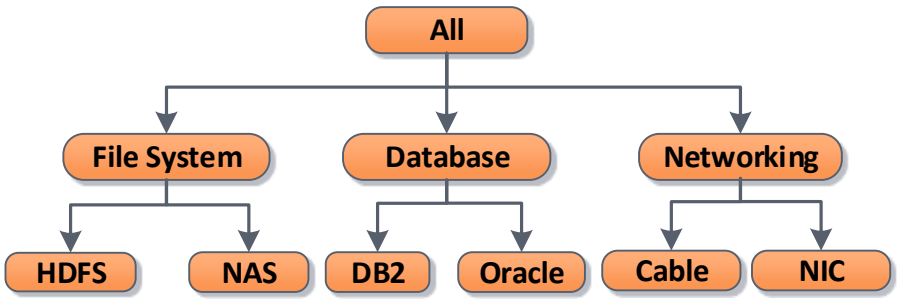

Figure 4.3: An example of taxonomy in IT tickets.

4.1.3 Contribution

To the best of our knowledge, this is the first work to formulate the automation recommendation in IT automation services as a multi-armed bandit problem by considering the dependencies among arms in the form of hierarchies. We demonstrate this approach on the automation recommendation for IT service management. The

contribution mainly focuses on proposing hierarchical multi-armed bandit algorithms to overcome the aforementioned three key challenges. The key features of our contribution include:

- A new online learning approach, designed to (1) solve the cold-start problem, and (2) continuously recommend an appropriate automation for the incoming ticket and adapt based on the feedback to improve the goodness of match between the problem and automation in IT automation services.
- Utilization of the hierarchies, integrated into bandit algorithms to model the dependencies among arms.

The effectiveness and efficiency of our proposed methods are verified on a large dataset of tickets from IBM Global Services.

The remainder of this chapter is organized as follows. In Section 4.2, we give the mathematical formalization of the problem. The solution to the problem is provided in Section 4.3. Section 4.4 describes comparative experiments and an empirical case study conducted over the real ticket data, which demonstrate the efficacy of the proposed algorithms. Finally, Section 4.5 summarizes and concludes this chapter.

4.2 Problem Formulation

In this section, we provide a mathematical formulation of the problem and describe a new contextual multi-armed bandit model, which can utilize a taxonomy defined by domain experts explicitly depicting the dependencies among arms. A glossary of notations mentioned in this work is summarized in Table 4.1.

Table 4.1: Important Notations

Notation	Description
$a^{(i)}$	the i-th arm.
$\mathcal{A}$	the set of arms, $\mathcal{A} = \{a^{(1)}, ..., a^{(K)}\}$.
$\mathcal{H}$	the hierarchy (taxonomy) defined by domain experts.
$\mathcal{X}$	d -dimensional context feature space.
$\mathbf{x}_t$	the context at time t.
$r_{k,t}$	the reward (payoff) of pulling the arm $a^{(k)}$ at time t.
$\hat{r}_{k,t}$	the predicted reward (payoff) for the arm $a^{(k)}$ at time t.
π	the policy for pulling arm sequentially.
R_π	the cumulative reward of the policy π.
$\mathbf{S}_{\pi,t}$	the sequence of $(\mathbf{x}_i, \pi(\mathbf{x}_i), r_{\pi(\mathbf{x}_i)})$ observed until time $t = 1, ..., T$.
θ_k	the coefficients predicting reward of the arm $a^{(k)}$.
σ_k^2	the reward prediction variance for arm $a^{(k)}$.
α, β	the parameters of the distribution of σ_k^2.
$\mu_\theta, \mathbf{\Sigma}_\theta$	the parameters of the distribution of θ.

4.2.1 Basic Concepts and Terminologies

Let $\mathcal{A} = \{a^{(1)}, ..., a^{(K)}\}$ denote a set of automations (i.e., scripted resolutions) feasible in IT automation system, where K is the number of the automations. Every time a ticket is reported, the online IT automation recommendation process selects an automation $a^{(i)} \in \mathcal{A}$ using contextual information (i.e., the symptom description in the ticket) and recommends it as a possible resolution for the ticket. Specifically, the contextual information for the reported ticket at time t is represented as a feature vector $\mathbf{x}_t \in \mathcal{X}$, where $\mathcal{X}$ denotes the d-dimensional feature space. After recommending

an IT automation $a^{(k)}$ at time t, its corresponding feedback is received, indicating whether the ticket has been successfully resolved or not.

We formalize the online IT automation recommendation process as a contextual multi-armed bandit problem where automations are constantly recommended and the underlying recommendation model is instantly updated based on the feedback collected over time. The graphic model representation for contextual multi-armed bandit problem is presented in Figure 4.4.

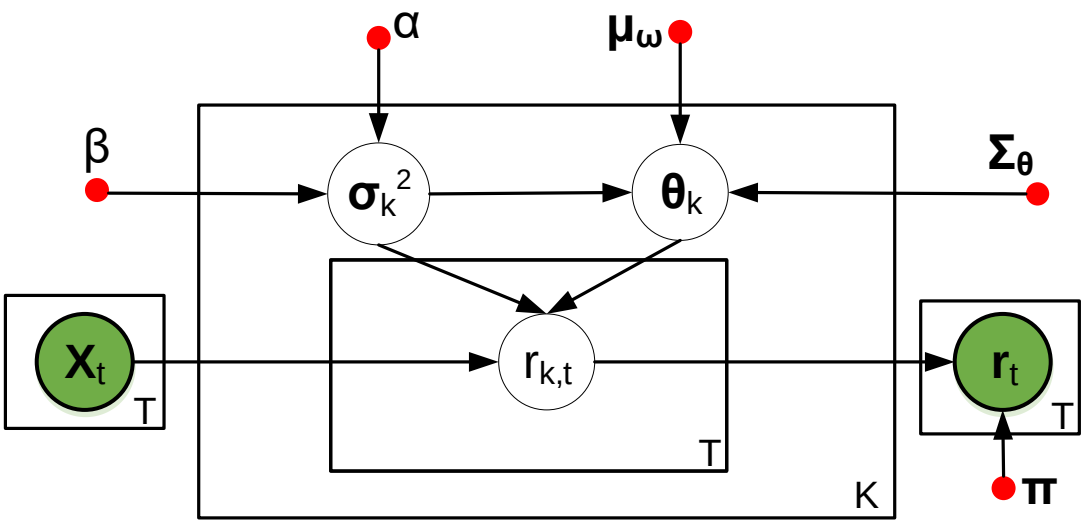

Figure 4.4: Graphical model representation for bandit problem. Random variable is denoted as a circle. The circle with green color filled means the corresponding random variable is observed. Red dot indicates a hyper parameter.

In general, a contextual multi-armed problem involves a series of decisions over a finite but possibly unknown time horizon T. In our formalization, each automation corresponds to an arm. Pulling an arm indicates its corresponding automation is being recommended, and the feedback (e.g., success or failure) received after pulling the corresponding arm is used to compute the reward.

In the contextual multi-armed bandit setting, at each time $t = [1, T]$, a policy π makes a decision for selecting an automation $\pi(\mathbf{x}_t) \in \mathcal{A}$ to perform an action according

to the contextual vector $\mathbf{x}_t$ of the current ticket. Let $r_{k,t}$ denote the reward for recommending an automation $a^{(k)}$ at time t, whose value is drawn from an unknown distribution determined by the context $\mathbf{x}_t$ presented to automation $a^{(k)}$. The total reward received by the policy π after T iterations is

$$R_\pi = \sum_{t=1}^{T} r_{\pi(\mathbf{x}_t)}. \tag{4.1}$$

The optimal policy π^* is defined as the one with maximum accumulated expected reward after T iterations,

$$\pi^* = \arg\max_{\pi} E(R_\pi) = \arg\max_{\pi} \sum_{t=1}^{T} E(r_{\pi(\mathbf{x}_t)}|t). \tag{4.2}$$

Our goal is to identify a good policy for maximizing the total reward. Herein we use reward instead of regret to express the objective function, since maximization of the cumulative reward is equivalent to minimization of regret during the T iterations [ZZW13].

Before selecting the optimal automation at time t, a policy π is updated to refine a model for reward prediction of each automation according to the historical observations $S_{\pi,t-1} = \{(\mathbf{x}_i, \pi(\mathbf{x}_i), r_{\pi(\mathbf{x}_i)})|i = [1, t-1]\}$. The reward prediction helps to ensure that the policy π includes decisions to increase the total reward. The reward $r_{k,t}$ is typically modeled as a linear combination of the feature vector $\mathbf{x}_t$ given as follows:

$$r_{k,t} = \mathbf{x}_t^T \theta_k + \xi_k, \tag{4.3}$$

where θ_k is a d-dimensional coefficient vector, and ξ_k denotes an observation noise, a zero-mean Gaussian noise with variance σ_k^2, i.e., $\xi_k \sim \mathcal{N}(0, \sigma_k^2)$. Then,

$$r_{k,t} \sim \mathcal{N}(\mathbf{x}_t^T \theta_k, \sigma_k^2), \tag{4.4}$$

and our objective function in Equation 4.2 can be reformulated as:

$$\pi^* = \arg\max_{\pi} \sum_{t=1}^{T} E_{\theta_{\pi(\mathbf{x}_t)}}(\mathbf{x}_t^T \theta_{\pi(\mathbf{x}_t)}|t). \tag{4.5}$$

To address the aforementioned problem, contextual multi-armed bandit algorithms have been proposed to balance the tradeoff between exploration and exploitation for arm selection, including ϵ-greedy, Thompson sampling, LinUCB, etc.

Thompson sampling is one of the earliest heuristic methods to address the contextual bandit problems, belonging to the probability matching family [AG13]. The main idea is to allocate the pulling chance according to the probability that an arm produces the maximum expected reward given the context $\mathbf{x}_t$ at time t. Particularly, Thompson sampling method learns and maintains the posterior distribution of the parameters in the reward prediction model for each arm. At every time t, Thompson sampling first samples the model parameters from its posterior distribution learnt at time $t-1$. The sampled parameters together with the contextual information $\mathbf{x}_t$ are used for reward prediction. The arm with maximum predicted reward is then selected to pull. Based on the feedback after pulling at time t, the posterior distribution of the model parameters for the selected arm at time t is updated and ready for arm selection at time $t+1$.

LinUCB [LCLS10], an extension of the UCB algorithm [Aue02], is another contextual bandit algorithm. It pulls the arm with the largest score computed by combining both reward expectation and deviation, which are computed in light of the reward prediction model.

Although different multi-armed bandit algorithms have been proposed and extensively adopted in diverse real applications, most of them do not take the dependencies between arms into account. In the IT environment, the automations (i.e., arms) are organized with its taxonomy, i.e., a hierarchical structure. The following section will introduce our approach to make use of the arm dependencies in the bandit settings for IT automation recommendation optimization.

4.2.2 Automation Hierarchy

In IT automation services, the automations can be classified with a pre-defined taxonomy. It allows us to reformulate the problem as a bandit model with the arm dependencies described by a tree-structured hierarchy.

Let $\mathcal{H}$ denote the taxonomy, which contains a set of nodes (i.e., arms) organized in a tree-structured hierarchy. Given a node $a^{(i)} \in \mathcal{H}$, $pa(a^{(i)})$ and $ch(a^{(i)})$ are used to represent the parent and children sets, respectively. Accordingly, we have Property 1.

Property 1 *If $pa(a^{(i)}) = \emptyset$, node $a^{(i)}$ is assumed to be the root node. If $ch(a^{(i)}) = \emptyset$, then $a^{(i)}$ is a leaf node, which represents an automation. Otherwise, $a^{(i)}$ is a category node when $ch(a^{(i)}) \neq \emptyset$.*

Since the goal is to recommend an automation for ticket resolving and only a leaf node of $\mathcal{H}$ represents an automation, the recommendation process cannot be completed until a leaf node is selected at each time t. Therefore, the multi-armed bandit problem for IT automation recommendation is reduced to select a path of $\mathcal{H}$ from root to a leaf node, where multiple arms along the path are sequentially selected with respect to the contextual vector $\mathbf{x}_t$ at time t.

Let $pth(a^{(i)})$ be a set of nodes, consisting of all the nodes along the path from root node to $a^{(i)}$ in $\mathcal{H}$. Further, assume $\pi_{\mathcal{H}}(\mathbf{x}_t|t)$ to be the path selected by policy π in light of the contextual information $\mathbf{x}_t$ at time t. Hence, we can have Property 2 for every arm selection policy π.

Property 2 *Given the contextual information $\mathbf{x}_t$ at time t, if a policy π selects a node $a^{(i)}$ in the hierarchy $\mathcal{H}$ and receives positive feedback (i.e., success), the policy π receives positive feedback as well by selecting the nodes in $pth(a^{(i)})$.*

Let $r_{\pi_{\mathcal{H}}(\mathbf{x}_t|t)}$ denote the reward obtained by the policy π after selecting the multiple arms along the path $\pi_{\mathcal{H}}(\mathbf{x}_t|t)$ at time t. The reward is computed as follows,

$$r_{\pi_{\mathcal{H}}(\mathbf{x}_t|t)} = \sum_{a^{(i)} \in \pi_{\mathcal{H}}(\mathbf{x}_t|t), ch(a^{(i)}) \neq \emptyset} r_{\pi(\mathbf{x}_t|ch(a^{(i)}))}, \tag{4.6}$$

where $\pi(\mathbf{x}_t|ch(a^{(i)}))$ represents the arm selected from the children of $a^{(i)}$, given the contextual information $\mathbf{x}_t$.

Therefore, after T iterations, the total reward received by the policy π is computed as below,

$$R_{\pi_{\mathcal{H}}} = \sum_{t=1}^{T} r_{\pi_{\mathcal{H}}(\mathbf{x}_t|t)}. \tag{4.7}$$

The optimal policy π^* with respect to $\mathcal{H}$ is determined by

$$\pi^* = \arg\max_{\pi} E(R_{\pi_{\mathcal{H}}}) = \arg\max_{\pi} \sum_{t=1}^{T} E(r_{\pi_{\mathcal{H}}(\mathbf{x}_t)}|t). \tag{4.8}$$

The reward prediction for each arm is conducted by Equation (4.4), and then the optimal policy can be equivalently determined by

$$\pi^* = \arg\max_{\pi} \sum_{t=1}^{T} (\sum_{\substack{a^{(i)} \in \pi_{\mathcal{H}}(\mathbf{x}_t|t), \\ ch(a^{(i)}) \neq \emptyset}} E_{\theta_{\pi(\mathbf{x}_t|ch(a^{(i)}))}} (\mathbf{x}_t^T \theta_{\pi(\mathbf{x}_t|ch(a^{(i)}))}|t)) \tag{4.9}$$

Both Thompson sampling and LinUCB mentioned above will be incorporated into our new learning models that leverage the hierarchies defined by domain experts. In such settings, bandit algorithms can achieve faster convergence by exploring feature space hierarchically.

4.3 Solution & Algorithm

In this section, we propose the HMAB (Hierarchical Multi-Armed Bandit) algorithms for exploiting the dependencies among arms organized hierarchically. As presented in Equation (4.4), the reward $r_{k,t}$ depends on random variable $\mathbf{x}_t$, θ_k and σ_k^2. We assume θ_k and σ_k^2 follow a conjugate prior distribution, Normal Inverse Gamma (abbr., $\mathcal{NIG}$)

distribution. σ_k^2 is drawn from the Inverse Gamma (abbr., $\mathcal{IG}$) distribution shown in Equation (4.10).

$$p(\sigma_k^2|\alpha_k, \beta_k) \sim \mathcal{IG}(\alpha_k, \beta_k), \tag{4.10}$$

where α_k and β_k are the predefined hyper parameters for the $\mathcal{IG}$ distribution. Given σ_k^2, the coefficient vector θ_k is generated by a Gaussian prior distribution with the hyper parameter μ_{θ_k} and $\mathbf{\Sigma}_{\theta_k}$:

$$p(\theta_k|\mu_{\theta_k}, \mathbf{\Sigma}_{\theta_k}, \sigma_k^2) \sim \mathcal{N}(\mu_{\theta_k}, \sigma_k^2\mathbf{\Sigma}_{\theta_k}), \tag{4.11}$$

At each time t, a policy π will select a path $\pi_{\mathcal{H}}(\mathbf{x}_t|t)$ from $\mathcal{H}$ according to the context $\mathbf{x}_t$. Assuming $a^{(p)} \in \pi_{\mathcal{H}}(\mathbf{x}_t|t)$ is the leaf node (i.e., an automaton), then we have $pth(a^{(p)}) = \pi_{\mathcal{H}}(\mathbf{x}_t|t)$. After recommending the automation $a^{(p)}$, a reward $r_{p,t}$ is obtained. Since the reward $r_{p,t}$ is shared by all the arms along the path $pth(a^{(p)})$, a set of triples $F = \{(\mathbf{x}_t, a^{(k)}, r_{k,t})|a^{(k)} \in pth(a^{(k)}), r_{k,t} = r_{p,t}\}$ are acquired. A new sequence $S_{\pi,t}$ is generated by incorporating the triple set F into $S_{\pi,t-1}$. The posterior distribution for every $a^{(k)} \in pth(a^{(k)})$ needs to be updated with the new feedback sequence $S_{\pi,t}$. The posterior distribution of θ_k and σ_k^2 given $S_{\pi,t}$ is a $\mathcal{NIG}$ distribution with the hyper parameter μ_{θ_k}, $\mathbf{\Sigma}_{\theta_k}$, α_k and β_k. These hyper parameters at time t are updated based on their values at time $t-1$:

$$\begin{aligned}
\mathbf{\Sigma}_{\theta_{k,t}} &= (\mathbf{\Sigma}_{\theta_{k,t-1}}^{-1} + \mathbf{x}_t\mathbf{x}_t^T)^{-1} \\
\mu_{\theta_{k,t}} &= \mathbf{\Sigma}_{\theta_{k,t}}(\mathbf{\Sigma}_{\theta_{k,t-1}}^{-1}\mu_{\theta_{k,t-1}} + \mathbf{x}_t r_{k,t}) \\
\alpha_{k,t} &= \alpha_{k,t-1} + \frac{1}{2} \\
\beta_{k,t} &= \beta_{k,t-1} + \frac{1}{2}[r_{k,t}^2 + \mu_{\theta_{k,t-1}}^T\mathbf{\Sigma}_{\theta_{k,t-1}}^{-1}\mu_{\theta_{k,t-1}} - \mu_{\theta_{k,t}}^T\mathbf{\Sigma}_{\theta_{k,t}}^{-1}\mu_{\theta_{k,t}}]
\end{aligned} \tag{4.12}$$

Note that the posterior distribution of θ_k and σ_k^2 at time t is considered as the prior distribution of time $t+1$. On the basis of the aforementioned inference of the leaf node $a^{(k)}$, we propose HMAB algorithms presented in Algorithm 1 developing different strategies including HMAB-TS($\mathcal{H}$, α, β) and HMAB-LinUCB($\mathcal{H}$, λ).

Algorithm 1 The algorithms for HMAB model

procedure MAIN($\mathcal{H}, \pi, \lambda$) ▷ Main entry, π is the policy.
  for $t \leftarrow 1, T$ **do**
    Initialize parameters of $a^{(m)} \in \mathcal{H}$ to α_m, β_m, $\mathbf{\Sigma}_{\theta_m} = \mathbf{I}_d$, $\mu_{\theta_m} = \mathbf{0}_{d\times 1}$.
    Get contextual vector $\mathbf{x}_t \in \mathcal{X}$.
    for each path P of $\mathcal{H}$ **do**
      Compute the reward of P using Equation (4.6), by calling EVAL($\mathbf{x}_t, a^{(k)}, \pi$) for each arm $a^{(k)} \in P$.
    end for
    Choose the path P^* with maximum reward.
    Recommend the automation $a^{(*)}$ (leaf node of P^*).
    Receive reward $r_{*,t}$ by pulling arm $a^{(*)}$.
    UPDATE($\mathbf{x}_t, P^*, r_{*,t}, \pi$)
  end for
end procedure

procedure EVAL($\mathbf{x}_t$, $a^{(k)}$, π) ▷ Get a score for $a^{(k)}$, given $\mathbf{x}_t$.
  if π is TS **then**
    Sample $\sigma^2_{k,t}$ according to Equation (4.10).
    Sample $\theta_{k,t}$ according to Equation (4.11).
    return $\hat{r}_{k,t} = \mathbf{x}_t^T \theta_{k,t}$.
  end if
  if π is LinUCB **then**
    return $\hat{r}_{k,t} = \mathbf{x}_t^T \mu_{\theta_{k,t-1}} + \frac{\lambda}{\sigma_{k,t-1}} \sqrt{\mathbf{x}_t^T \mathbf{\Sigma}^{-1}_{\theta_{k,t-1}} \mathbf{x}_t}$
  end if
end procedure

procedure UPDATE($\mathbf{x}_t, P, r_t, \pi$) ▷ Update the inference.P is the path in $\mathcal{H}$, r_t is the reward.
  for each arm $a^{(k)} \in P$ **do**
    Update $\alpha_{k,t}$, $\beta_{k,t}$, $\mathbf{\Sigma}_{\theta_{k,t}}$, $\mu_{\theta_{k,t}}$ using Equation (4.12).
  end for
end procedure

Online inference of our hierarchical bandit problem starts with MAIN procedure. As a ticket $\mathbf{x}_t$ arrives at time t, the EVAL procedure computes a score for each arm of different levels. In each level, the arm with the maximum score is selected to be pulled. After receiving reward by pulling an arm, the new feedback is used to update the HMAB algorithms by the UPDATE procedure.

4.4 Experiment Setup

To demonstrate the efficiency of our proposed algorithms, we conduct a large scale experimental study over a real ticket dataset from IBM Global Services. First, we outline the general implementation of the baseline algorithms for comparison. Second, we describe the dataset and evaluation method. Finally, we discuss the comparative experimental results of the proposed and baseline algorithms, and present a case study to demonstrate the effectiveness of HMAB algorithms.

4.4.1 Baseline Algorithms

In the experiments, we demonstrate the performance of our methods by comparing the following baseline algorithms:

1. `Random`: a random item recommended to the targeted user without considering the contextual information.
2. `EpsGreedy`(ϵ): a random arm with probability ϵ selected, as well as the arm of the largest predicted reward $\hat{r}_{k,t}$ with probability $1-\epsilon$, where ϵ is a predefined parameter.
3. `TS`(α, β): Thompson sampling described in Section 4.2.1, randomly draws the coefficients from the posterior distribution, and selects the item with the largest estimated payoff according to Equation (4.4). Both α and β are hyper parameters. We initial α and β with the same value.
4. `LinUCB`(λ): the parameter λ is used to calculate the score, a linear combination of the expectation and deviation of the reward. The arm with the largest score is selected. When $\lambda = 0$, it is equivalent to the `Exploit` policy.

Our methods proposed in this chapter include:

1. `HAMB-EpsGreedy`($\mathcal{H}$, ϵ): a random arm with probability ϵ is selected, and the arm of the highest estimated reward $\hat{r}_{k,t}$ with probability $1-\epsilon$ with respect to the hierarchy $\mathcal{H}$, which is a predefined parameter as well as ϵ.
2. `HMAB-TS`($\mathcal{H}, \alpha, \beta$): it denotes our proposed hierarchical multi-armed bandit with `Thompson sampling` outlined in Algorithm 1. $\mathcal{H}$ is the taxonomy defined by domain experts. α and β are hyper parameters.
3. `HMAB-LinUCB`($\mathcal{H}, \lambda$): it represents our proposed algorithm based on `LinUCB` presented in Algorithm 1. Similarly, $\mathcal{H}$ is the hierarchy depicting the dependencies among arms. And the parameter λ is given with the same use in `LinUCB`.

4.4.2 Dataset Description

Experimental tickets are collected by IBM Tivoli Monitoring system [urlf]. This dataset covers from July 2016 to March 2017 with the size of $|\mathcal{D}| = 116,429$. Statistically, it contains 62 automations (e.g., NFS Automation, Process CPU Spike Automation, and Database Inactive Automation) recommended by the automation engine to fix the corresponding problems. The execution feedback including success, failure and escalation, indicates whether the problem has been resolved or needs to be escalated to human engineers. These collected feedback can be utilized to improve the accuracy of recommended results. Thereby, the problem of automation recommendation can be regarded as an instance of the contextual bandit problem. As we mentioned above, an arm is an automation, a pull is to recommend an automation for an incoming ticket, the context is the information vector of ticket's description, and the reward is the feedback on the result of the execution of recommended automation on the problem server. An automation hierarchy $\mathcal{H}$ shown in Figure 4.5 with three layers constructed by domain experts is introduced to present the dependencies

among automations. Moreover, each record is stamped with the open time of the ticket.

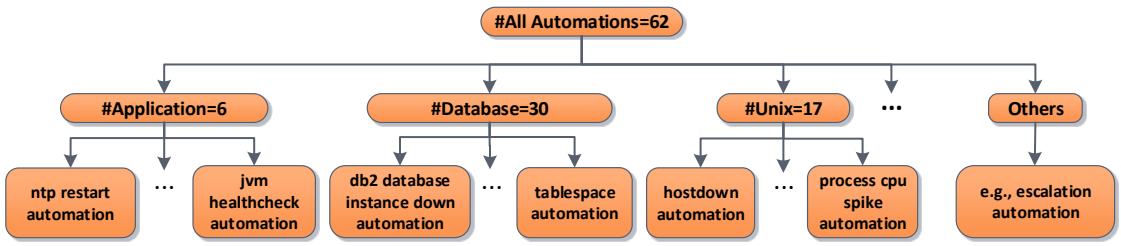

Figure 4.5: An automation hierarchy defined by domain experts.

We now discuss how to construct ticket features for the experiments. To reduce the noise of the data, the domain experts only selected the categorical attributes (e.g., ALERT_ KEY, CLIENT_ ID, SEVERITY and OSTYPE) with high representative information of tickets. These categorical information of a ticket is encoded as a binary vector [LCL+12]. In addition, we augmented a constant feature with value 1 for all vectors. Therefore, each ticket is represented as a binary feature vector $\mathbf{x}$ of dimension 1,182.

Evaluation Method

We apply the *replayer* method to evaluate our proposed algorithms on the aforementioned dataset. The *replayer* method is first introduced in [LCL+12], which provides an unbiased offline evaluation via the historical logs. The main idea of *replayer* is to replay each user visit to the algorithm under evaluation. If the recommended item by the testing algorithm is identical to the one in the historical log, this visit is considered as an impression of this item to the user. The ratio between the number of user clicks and the number of impressions is referred to as Click-through rate (CTR). The work in [LCL+12] shows that the CTR estimated by the *replayer* method approaches the real CTR of the deployed online system. In this problem, a user is a ticket, and an item is an automation. A user visit means a ticket comes into IT automation services,

and a user click indicates the ticket has been successfully solved by the recommended automation.

Relative Success Rate Optimization for Online Automation Recommendation

In this section, we discuss the performance evaluation for each proposed algorithm on the dataset. The averaged reward (i.e., the overall success rate of the corresponding automations) is considered as the metric in the experiments. We define it as the total reward divided by the total number of times a given automation has been recommended (i.e., $\frac{1}{n}\sum_{t=1}^{n} r_t$). It is obvious that the higher the success rate, the better the performance of the algorithm. To avoid the leakage of business-sensitive information, the relative success rate is reported, which is the overall success rate of an algorithm divided by the overall success rate of random selection.

In contrast to the baseline algorithms outlined in Section 4.4.1, the corresponding HMABs configured with different parameter settings achieve much better performance on the dataset shown in Figure 4.6, Figure 4.7 and Figure 4.8, respectively. To be clarified, we set the parameter $\lambda > 1$ of LinUCB and HMAB-LinUCB in the experiments deliberately to reveal the merits of HMAB-LinUCB because their performance are almost equal with $\lambda < 1$. By observing the results, we find that HMAB-LinUCB has the best performance compared with other algorithms. Through these substantial experiments, we conclude that the proposed algorithms outperformed the strong baselines with the assumption that arms are independent.

A Comparative Case Study

In order to better illustrate the merits of the proposed algorithms, we present a case study on the recommendation for an escalated ticket in IT automation services.

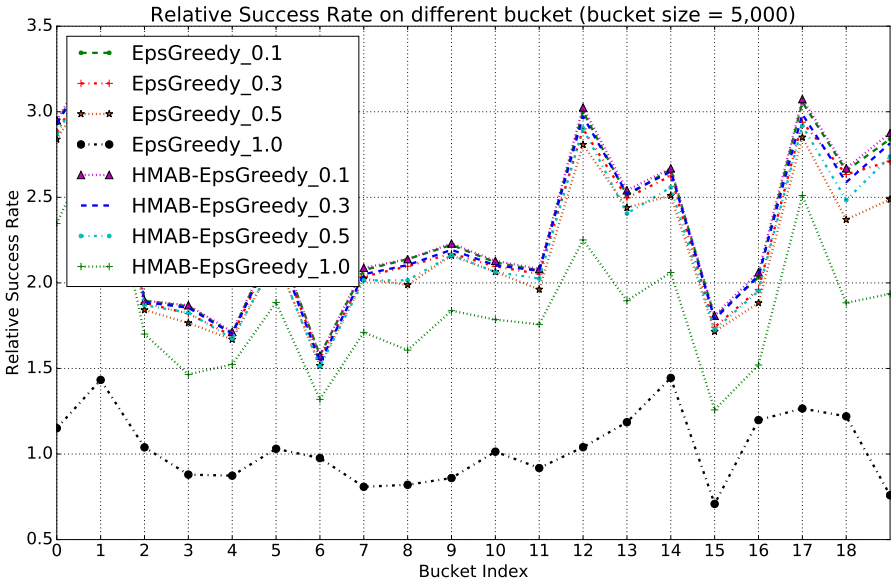

Figure 4.6: The Relative Success Rate of EpsGreedy and HMAB-EpsGreedy on the dataset is given along each time bucket with diverse parameter settings.

As mentioned above, the recommendation for an escalated ticket can be regarded as a cold-start problem due to the lack of the corresponding automations. In other words, there is no historical records for resolving this ticket. Note that both our proposed HMABs and conventional MABs are able to deal with the cold-start problem by *exploration*. To compare their performance, we calculate the distribution of the recommended automations over different categories (e.g., database, unix, and application). Figure 4.10 presents an escalated ticket, which records a database problem. Such a problem has been repeatedly reported over time in the dataset. Since this ticket reports a database problem, intuitively the automations in the database category should have a high chance of being recommended. The category distributions of our proposed HMABs and conventional MABs are provided in Figure 4.9, as well as the baseline category distribution, which is the prior category distribution obtained from all the automations of the hierarchy. From Figure 4.9, we observe that 1) compared with TS, HMAB-TS explores more automations from the database category;

and 2) in HMAB-TS the database category has the highest percentage among all the automation categories. This shows that our proposed HMABs can achieve better performance by making use of the predefined hierarchy.

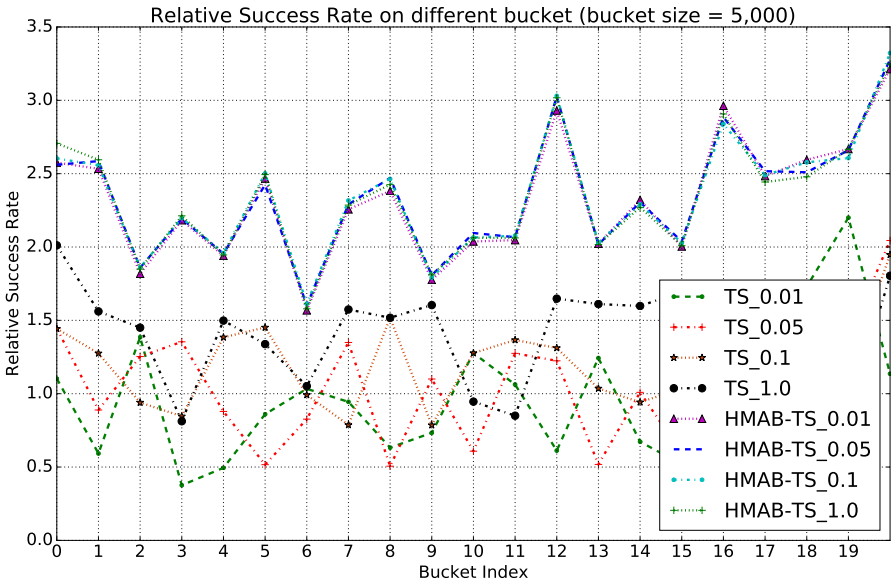

Figure 4.7: The Relative Success Rate of TS and HMAB-TS on the dataset is given along each time bucket with diverse parameter settings.

To further illustrate the effectiveness of HMABs, we provide the detailed results of recommended automations for the escalated tickets. As shown in Figure 4.10, automations from the database category (e.g., database instance down automation, db2 database inactive automation) are frequently recommended according to the context of the ticket, which clearly indicate the issue is due to the inactive database. By checking the recommended results, domain experts figure out the **database instance down automation**, one of the top recommended automations, can successfully fix such a cold-start ticket problem, which clearly demonstrate the effectiveness of our proposed algorithms.

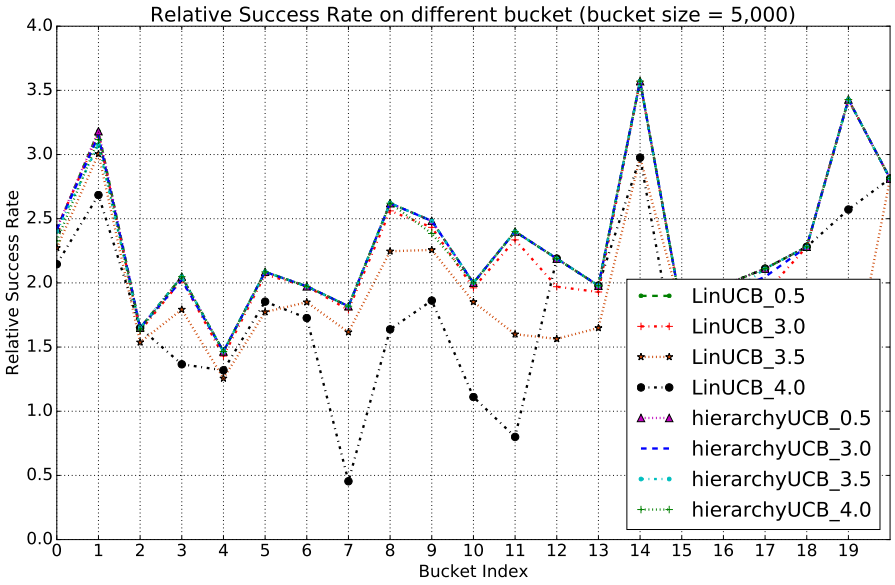

Figure 4.8: The Relative Success Rate of LinUCB and HMAB-LinUCB on the dataset is given along each time bucket with diverse parameter settings.

4.5 Summary

In this chapter, we propose a novel parametric model (HMAB) to formulate the automation recommendation in IT automation services as a contextual multi-armed bandit problem, where the arms are organized in the form of a taxonomy. To show the effectiveness of our proposed solutions, empirical experiments are conducted on a real ticket dataset compared with conventional bandit algorithms, which assume that the arms are independent. In a case study of solving a cold-start problem, our proposed algorithms show a better performance due to usage of the hierarchy.

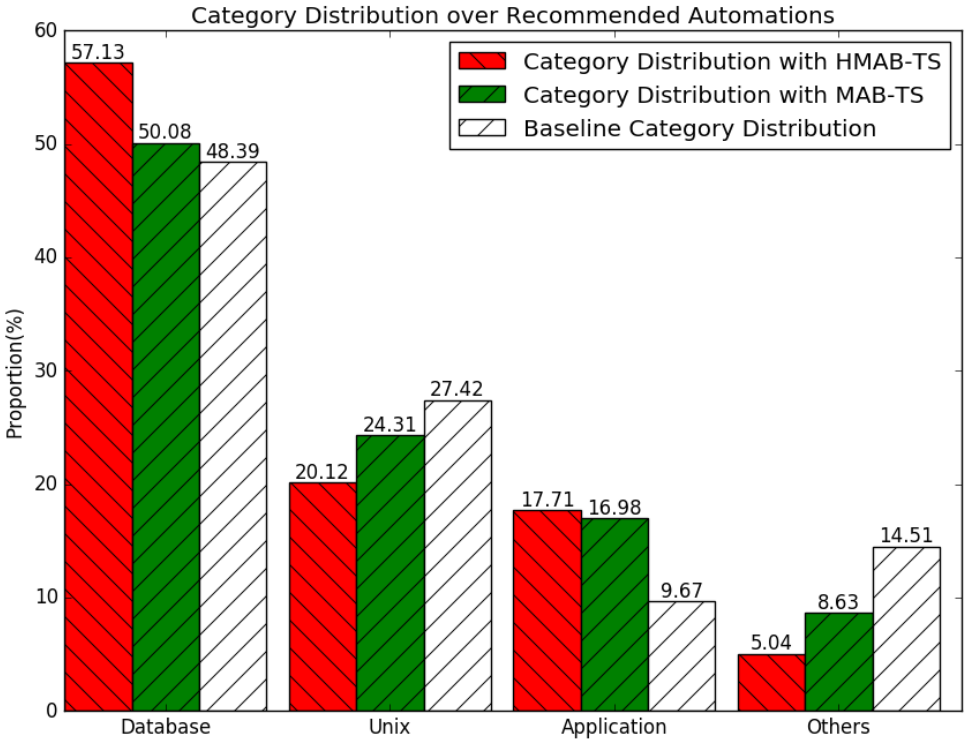

Figure 4.9: The comparison of category distribution on the recommended automations.

ALERT_KEY	ac2_dbinact_grzc_std	**AUTOMATON NAME**	Escalation Handler
TICKET SUMMARY	Database fin91dmo status is inactive.	**TICKET RESOLUTION**	The database is down. It has been restarted, hence closing the ticket.

RECOMMENDED CATEGORY	**(%)**	**RECOMMENDED AUTOMATON**
DATABASE	**57.13**	(1) **database** instance **down** automation; (2) db2 **database inactive** automation; (3) mysql **database offline** automation.
UNIX	20.12	(1) asm space check diskgroup **dbautomation;** (2) **hostdown** automation; (3) certification **expiration** automation.
APPLICATION	17.71	(1) ntp **restart** automation; (2) mq manager **down** automation.
OTHERS	5.04	(1) system load automation; (2) others.

Figure 4.10: The *exploration* by HMAB-TS of a cold-start ticket case.

CHAPTER 5

LEARN AUTOMATION INTELLIGENCE BY INTERACTIVE COLLABORATIVE TOPIC REGRESSION MODEL

The reality of IT environments is such that not all automations are properly set in a hierarchical structure due to the lack of sufficient information and some may fall into the "Others" category (see Figure 5.1). Furthermore, as a result of the imperfection of log information, many tickets are logged with an error code only with no detailed symptoms. This makes inferring a proper automation more challenging. By observing the rich historical data, we find different ticket problems (see Figure 5.2) that can be resolved by the same automation, while a single ticket problem may be resolved by different automations, which can be treated as an interactive collaborative filtering problem [ZZW13]. In this chapter, we consider such a context-free automation recommendation in IT service management as a real-world application of online interactive recommender systems, which can adaptively learn the preferences of each ticket problem on automations.

Online interactive recommender systems strive to promptly suggest users appropriate items (e.g., movies, news articles) according to the current context including both user and item content information. However, such contextual information is often unavailable in practice, where only the users' interaction data on items can be utilized by recommender systems. The lack of interaction records, especially for new users and items, inflames the performance of recommendation further. To address these issues, both collaborative filtering, one of the most popular recommendation techniques relying on the interaction data only, and bandit mechanisms, capable of achieving the balance between exploitation and exploration, are adopted into an online interactive recommendation setting assuming independent items (i.e., arms). This assumption rarely holds in reality, since the real-world items tend to be cor-

related with each other. we study online interactive collaborative filtering problems by considering the dependencies among items. We explicitly formulate item dependencies as the clusters of arms in the bandit setting, where the arms within a single cluster share the similar latent topics. In light of topic modeling techniques, we come up with a novel generative model to generate the items from their underlying topics. Furthermore, an efficient particle-learning based online algorithm is developed for inferring both latent parameters and states of our model by taking advantage of the fully adaptive inference strategy of particle learning techniques. Additionally, our inferred model can be naturally integrated with existing multi-armed selection strategies in an interactive collaborative filtering setting.

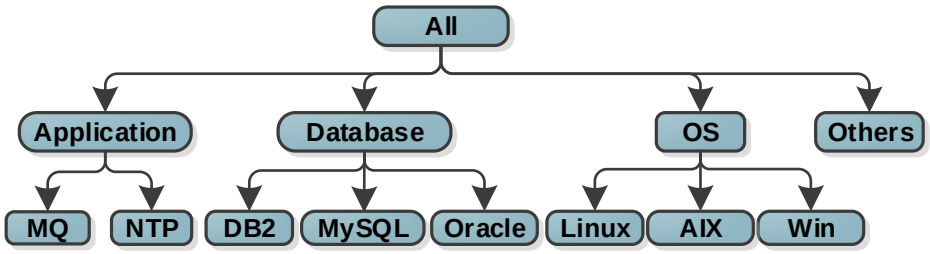

Figure 5.1: An example of taxonomy in IT tickets with "Others" category.

Ticket Problem 1

ALERT_KEY	xxx_cpusum_xuxc_aix		AUTOMATON_NAME		Process CPU Spike Automation		
OPEN_DTTM	**CLIENT_ID**	**HOSTNAME**	**ORIGINAL SEVERITY**	**OSTYPE**	**COMPONENT**	**SUBCOMPONENT**	**AUTO RESOVLED**
1456383421000	90	XXX	4	AIX	UNIX	UNKNOWN	1
TICKET SUMMARY	XXX CPU Utilization is very high, workloads affected			**TICKET RESOLUTION**	Alert in question has recovered, hence closing the ticket.		

Ticket Problem 2

ALERT_KEY	xxx_prccpu_rlzc_std		AUTOMATON_NAME		Process CPU Spike Automation		
OPEN_DTTM	**CLIENT_ID**	**HOSTNAME**	**ORIGINAL SEVERITY**	**OSTYPE**	**COMPONENT**	**SUBCOMPONENT**	**AUTO RESOVLED**
1454900281000	52	XXX	4	UNKNOWN	LINUX	Process	1
TICKET SUMMARY	XXX Process using high cpu (97.30%) nsrdmpix			**TICKET RESOLUTION**	Alert in question has recovered, hence closing the ticket.		

Figure 5.2: Two different ticket problems in IT service management.

5.1 Introduction

The overwhelming amount of data requires an efficient online interactive recommendation system where online users constantly interact with the system, and user feedback is instantly collected to improve recommendation performance. Online interactive recommender systems are challenged to immediately recommend the most proper items (e.g., movies, news articles) to users based on the current user and item content information aiming to continuously maximize users' satisfaction over a long run. To achieve this goal, it becomes a critical task for such recommender systems to constantly track user preferences and recommend interesting items from a large item repository.

In the process of identifying the appropriate match between user preferences and target items, the systems encounter difficulties due to several existing practical challenges. One challenge is the well-known *cold-start* problem since a significant number of users/items might be completely new to the system, that is, they may have no consumption historical records at all. This problem makes recommender systems ineffective unless additional information including both items and users is collected [ZWML16], [CZC+15]. The second challenge is that most recommender systems typically assume the entire set of contextual features with respect to both users and items can be accessed to infer users' preference. Due to a number of reasons (e.g., privacy or sampling constraints), it is challenging to obtain all relevant features ahead of time, thus rendering many factors unobservable to recommendation algorithms.

In the first challenge, an *exploration* or *exploitation* dilemma [BU17] is identified in the aforementioned setting. A tradeoff between two competing goals needs to be considered in recommender systems: maximizing user satisfaction using their consumption history, while gathering new information for improving the goodness of match between user preferences and items [LCLS10]. This dilemma is typically

formulated as a multi-armed bandit problem where each arm corresponds to an item. The recommendation algorithm determines the strategies for selecting an arm to pull according to the contextual information at each trial. Pulling an arm indicates that the corresponding item is recommended. When an item matches the user preference (e.g., a recommended news article or movie is consumed), a reward is obtained; otherwise, no reward is provided. The reward information is fed back to optimize the strategies. The optimal strategy is to pull the arm with the maximum expected reward based on the historical interaction on each trial, and then to maximize the total accumulated rewards for the whole series of trials.

Collaborative filtering (CF) is widely applied in recommender systems to address the second challenge [SFHS07],[BL$^+$07],[KBV09] . CF has gained its popularity due to its advantage over other recommendation techniques, where CF requires no extra information about items or users for recommendation but only users' historical ratings on items [KBK$^+$15, HKBR99]. Further, considering both aforementioned challenges simultaneously aggravates the difficulties when recommending items. Recently, an online interactive collaborative filtering system has been suggested [ZZW13, KBK$^+$15] adopting both techniques, multi-armed bandit and collaborative filtering. Typically, one collaborative filtering task is formulated as a matrix factorization problem. Matrix factorization derives latent features for both users and items from the historical interaction records. It assumes that a user's preference (i.e., rating) on a given item can be predicted considering the item and user latent feature vectors. Based on this assumption, multi-armed bandit policies make use of the predicted reward (i.e., user preference) for arm (i.e., item) selection. The feedback occurring between the current user and arm is used to update their latent vectors, without impacting the inference of other arms' latent vectors assuming arms are independent from each other. However, the assumption about the independency among arms rarely holds in real-world

applications. For example, in the movie recommendation scenario, each movie corresponds to an arm. The dependent arms (i.e., movies) typically share similar latent topics (e.g., science fiction movies, action movies, etc.), and are likely to receive similar rewards (i.e., ratings or feedback) from users. Intuitively, the dependencies among arms can be utilized for reward prediction improvement and further facilitated the maximization of users' satisfaction in a long run.

In this chapter, we introduce an interactive collaborative topic regression model that utilizes bandit algorithms with dependent arms to recommend appropriate items for target users. A sequential online inference method is proposed to learn the latent parameters and infer the latent states. We adopt a generative process based on topic model to explicitly formulate the arm dependencies as the clusters on arms, where dependent arms are assumed to be generated from the same cluster. Every time an arm is pulled, the feedback is not only used for inferring the involved user and item latent vectors, but it is also employed to update the latent parameters with respect to the arm's cluster. The latent cluster parameters further help with reward prediction for other arms in the same cluster. The fully adaptive online inference strategy of particle learning [CJLP10] allows our model to effectively capture arm dependencies. In addition, the learnt parameters can be naturally integrated into existing multi-arm selection strategies, such as *UCB* and *Thompson sampling*. We conduct empirical studies on three real-world applications, movie recommendation, news recommendations, and ticket automation recommendation. The experimental results demonstrate the effectiveness of our proposed approach.

The rest of this chapter is organized as follows. We formulate the underlying problem in Section 5.2. The solution to the problem is presented in Section 5.3. Extensive evaluation results are reported in Section 5.4. Finally, Section 5.5 summary this chapter.

5.2 Problem Formulation

In this section, we provide a mathematical formulation of the interactive collaborative filtering (ICF) problem into a bandit setting. Then we introduce a new generative model describing the dependency among arms (i.e. items). A glossary of notations mentioned in this chapter is summarized in Table 5.1.

5.2.1 Basic Concepts and Terminologies

Assume that there are M users and N items in the system. The preferences of the users for the items are recorded by a partially observable matrix $\mathbf{R} = \{r_{m,n}\} \in \mathcal{R}^{M \times N}$, where the rating score $r_{m,n}$ indicates how user m would like item n. The basic collaborative filtering task is to predict the unknown rating score in light of the observed rating scores in $\mathbf{R}$. However, it is very challenging to fulfill the task in practice due to the high dimensionality and sparsity of the rating matrix. Matrix factorization addresses this challenge by mapping each user m and item n to the latent feature vectors $\mathbf{p}_m \in \mathcal{R}^K$ and $\mathbf{q}_n \in \mathcal{R}^K$ in a shared low K-dimension space (typically, $K \ll M, N$). It assumes that the rating $r_{m,n}$ can be predicted by

$$y_{m,n} = \mathbf{p}_m^\intercal \mathbf{q}_n. \tag{5.1}$$

Therefore, the latent features $\{\mathbf{p}_m\}$ and $\{\mathbf{q}_n\}$ can be learned by minimizing the prediction error for all observed ratings in $\mathbf{R}$, while each unobserved rating value can be estimated using Equation (5.1) with its corresponding latent features learned by matrix factorization. In practice, since the feedback (i.e., rating scores) from users is received over time, the system is required to address the collaborative filtering problem in an interactive mode, which is referred to as an interactive recommender system.

Table 5.1: Important notations.

Notation	Description
M, N	number of rows (users) and columns (items).
$\mathbf{R} \in \mathcal{R}^{M \times N}$	the rating matrix.
$\mathbb{S}(t)$	the sequence of $(n(t-1), r_{m,n(t-1)})$ observed until time t.
$n(t)$	the recommended item index in the t-th iteration.
$r_{m,t}$	the rating (reward) of the m-th user by pulling the given item in the t-th iteration.
$y_{m,t}$	the predicted rating for the m-th user over given item in the t-th iteration.
π	the policy to recommend items sequentially.
R_π	the cumulative rating (reward) of the policy π.
K	the number of topics and the number of dimensions for latent vectors.
$\mathbf{p}_m \in \mathcal{R}^K$	the latent feature vector of the m-th user.
$\mathbf{q}_n \in \mathcal{R}^K$	the latent feature vector of the n-th item.
$\mathbf{\Phi}_k \in \mathcal{R}^N$	the item distribution of the k-th topic.
$\mathcal{P}_{m,n(t-1)}$	the set of particles for item $n(t-1)$ given user m at time $t-1$.
$z_{m,t}$	the latent topic of the m-th user in the t-th iteration.
$x_{m,t}$	the selected item of the m-th user in the t-th iteration.
λ	Dirichlet priors over topics for topic model.
η	Dirichlet priors over items for topic model.
σ_n^2	the variance of rating prediction.
α, β	the hyper parameters determine the distribution of σ_n^2.
$\mu_{\mathbf{q}}$, $\mathbf{\Sigma}_{\mathbf{q}}$	the hyper parameters determine the Gaussian distribution of $\mathbf{q}_n$.
ξ	the observation noise of the rating.

In an interactive recommender system, user m constantly arrives to interact with the system over time. At each time $t = [1, \ldots, T]$, the system, according to the observed rating history, recommends an item $n(t)$ to the corresponding user m. After consuming item $n(t)$, the feedback (i.e., rating) $r_{m,n(t)}$ from user m is collected by the system and further utilized to update the model for the next item delivery. The interactive recommendation process involves a series of decisions over a finite but possibly unknown time horizon T. Accordingly, such an interactive recommendation process is modeled as a multi-armed bandit problem, where each item corresponds to an arm. Pulling an arm indicates that its corresponding item is being recommended and the rating score is considered as the reward received after pulling the corresponding arm.

Let $\mathbb{S}(t)$ be the available information at time t collected by the system for the target user m,

$$\mathbb{S}(t) = \{(n(1), r_{m,n(1)}), \ldots, (n(t-1), r_{m,n(t-1)})\}. \tag{5.2}$$

A policy π is defined as a function and used to select an arm based on the current cumulative information $\mathbb{S}(t)$,

$$n(t) = \pi(\mathbb{S}(t)). \tag{5.3}$$

The total rewards received by the policy π after T iterations is

$$R_\pi = \sum_{t=1}^{T} r_{m,\pi(\mathbb{S}(t))}. \tag{5.4}$$

The optimal policy π^* is defined as the one with maximum accumulated expected reward after T iterations,

$$\pi^* = \arg\max_\pi \mathbb{E}(R_\pi) = \arg\max_\pi \sum_{t=1}^{T} \mathbb{E}(r_{m,\pi(\mathbb{S}(t))}|t). \tag{5.5}$$

Therefore, our goal is to identify an optimal policy for maximizing the total rewards. Herein we use reward instead of regret to express the objective function, since maximization of the cumulative rewards is equivalent to minimization of regret during the

T iterations [ZZW13]. Before selecting one arm at time t, a policy π typically learns a model to predict the reward for every arm according to the historical accumulated information $\mathbb{S}(t)$. The reward prediction helps the policy π make decisions to increase the total rewards.

In the latent factor model [MS08, SM08], the rating is estimated by a product of user and item feature vectors $\mathbf{p}_m$ and $\mathbf{q}_n$ in Equation (5.1). From the probabilistic perspective, PMF introduces an observation noise ξ, a zero-mean Gaussian noise with variance σ^2 (i.e., $\xi \sim \mathcal{N}(0, \sigma^2)$), to the rating prediction function given in Equation (5.1). The derived rating prediction is as follows:

$$r_{m,n} = \mathbf{p}_m^\intercal \mathbf{q}_n + \xi. \tag{5.6}$$

In this setting, Equation (5.5) can be re-formulated as:

$$\pi^* = \arg\max_{\pi} \sum_{t=1}^{T} \mathbb{E}_{\mathbf{p}_m, \mathbf{q}_{\pi(\mathbb{S}(t))}} (\mathbf{p}_m^\intercal \mathbf{q}_{\pi(\mathbb{S}(t))} | t). \tag{5.7}$$

Consequently, the goal of an interactive recommender system is reduced to the optimization of the objective function in Equation (5.7).

Thompson Sampling, one of earliest heuristics for the bandit problem [CL11], belongs to the probability matching family. Its main idea is to randomly allocate the pulling chance according to the probability that an arm gives the largest expected reward at a particular time t. Based on the objective function in Equation (5.7), the probability of pulling arm n can be expressed as follows:

$$\begin{aligned} p(n(t) = n) = \int \mathbb{I}[\mathbb{E}(r_{m,n} | \mathbf{p}_m, \mathbf{q}_n) = \max_i \mathbb{E}(r_{m,i} | \mathbf{p}_m, \mathbf{q}_i)] \\ p(\mathbf{p}_m, \mathbf{q}_n | t) d\mathbf{p}_m d\mathbf{q}_n. \end{aligned} \tag{5.8}$$

At each time t, Thompson sampling samples both the user and item feature vectors together from their corresponding distributions, and then selects the item that leads to the largest reward expectation. Therefore, using Thompson sampling, the item

selection function can be defined as:

$$n(t) = \arg\max_{n} (\tilde{\mathbf{p}}_m^{\intercal} \tilde{\mathbf{q}}_n | t), \tag{5.9}$$

where $\tilde{\mathbf{p}}_m$ and $\tilde{\mathbf{q}}_n$ denote the sampled feature vectors for user m and item n, respectively.

To accomplish Thompson sampling, it is critical to model the random variable $\mathbf{p}_m$ and $\mathbf{q}_n$ using distributions, where the latent feature vectors can be easily sampled and the feedback at every time can be reasonably integrated. Most of the previous studies suppose a Gaussian prior for both user and item feature vectors with an assumption that items are independent from each other [ZZW13, KBK$^+$15]. However, this assumption rarely holds in real applications. In the following section, we explicitly formulate the dependent arms with a generative model.

5.2.2 Modeling the Arm Dependency

Based on the fact that similar items (i.e., arms) are likely to receive similar feedback (i.e., rewards), we assume that a dependency exists among similar items. The dependencies among items can be further leveraged to improve the users' preferences inference on a particular item even if the item has little historical interaction data in the system. The challenge here lies in how to sequentially infer the arms' dependencies as well as the users' preferences simultaneously, providing the feedback over time.

In our work, the arms' dependencies are expressed in the form of the clusters of arms, where the dependent arms fall into the same one. In order to explore the dependencies in the bandit setting, Latent Dirichlet Allocation (LDA) [BNJ03], a generative statistic model for topic modeling, is adopted to construct the arms' clus-

ters. We propose the ICTR (Interactive Collaborative Topic Regression) model to infer the clusters of arms as well as the arm selection.

The main idea of our model is to treat an item n as a word, while consider a user m as a document. All the items rated by a user indicate the hidden preferences of the user, analogous to the scenario in topic modeling where the words contained in a document imply its latent topics. Specifically, let K be the number of latent aspects (i.e., topics or clusters) the users concern when consuming items. We assume that $\mathbf{p}_m \in \mathcal{R}^K$ corresponds to the latent vector for user m, where the k-th component of $\mathbf{p}_m$ indicates the user's preference over the k-th aspect of items. Further, $\mathbf{q}_n \in \mathcal{R}^K$ is supposed to be the latent vector for item n, and the k-th component value of $\mathbf{q}_n$ represents that it belongs to the k-th cluster. The rating score $r_{m,n}$, given by user m after consuming item n, is assumed to be the inner product of $\mathbf{p}_m$ and $\mathbf{q}_n$. By linking to the topic model, a generative process for user ratings is accordingly introduced and presented in Figure 5.3.

Based on the above description, the user latent vector $\mathbf{p}_m$ is assumed to follow a Dirichlet prior distribution with a predefined hyper parameter λ, shown in Equation (5.10).

$$\mathbf{p}_m|\lambda \sim Dir(\lambda). \tag{5.10}$$

As presented in Equation (5.6), we denote σ^2 as the variance of the noise for reward prediction and assume σ_n^2 is drawn from the Inverse Gamma ($\mathcal{IG}$) distribution shown in the following.

$$p(\sigma_n^2|\alpha,\beta) = \mathcal{IG}(\alpha,\beta), \tag{5.11}$$

where α and β are predefined hyper parameters for $\mathcal{IG}$ distribution.

Given σ_n^2, the item latent vector $\mathbf{q}_n$ is generated by a Gaussian prior distribution as follows:

$$\mathbf{q}_n|\mu_{\mathbf{q}}, \mathbf{\Sigma}_{\mathbf{q}}, \sigma_n^2 \sim \mathcal{N}(\mu_{\mathbf{q}}, \sigma_n^2\mathbf{\Sigma}_{\mathbf{q}}), \tag{5.12}$$

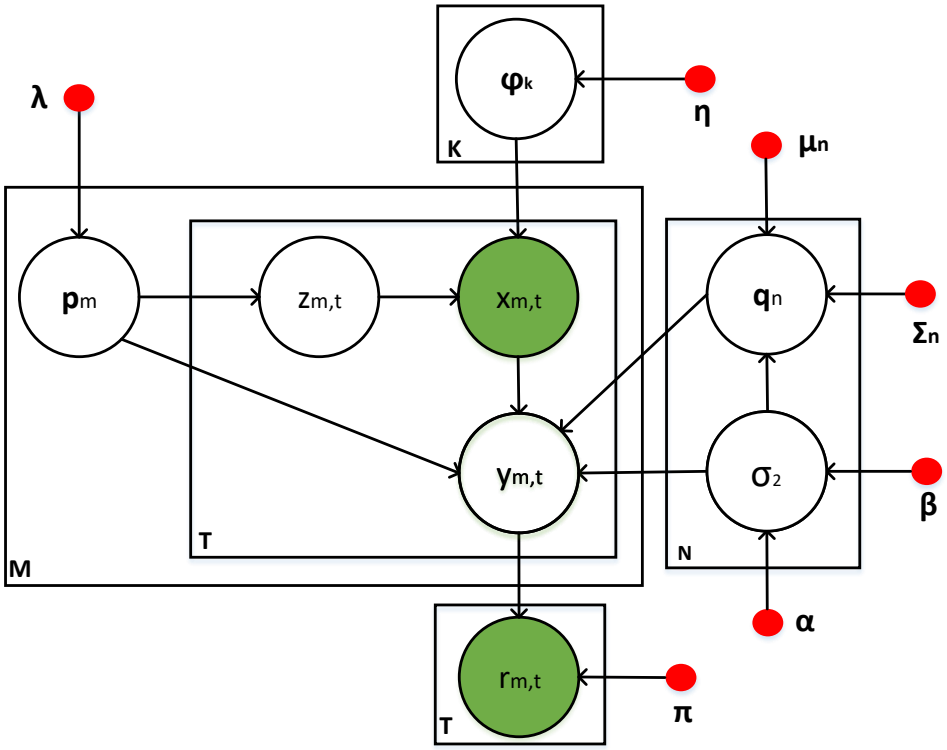

Figure 5.3: The graphic model for the ICTR model. Random variable is denoted as a circle. The circle with filled color denotes the observed random variable. Red dot represents a hyper parameter.

where $\mu_{\mathbf{q}}$ and $\mathbf{\Sigma}_{\mathbf{q}}$ are predefined hyper parameters.

Further, let

$$\mathbf{\Phi}_k \in \mathcal{R}^N$$

be the item distribution for topic k. Similar to $\mathbf{p}_m$, Dirichlet distribution is specified as the prior of $\mathbf{\Phi}_k$ presented in Equation (5.13).

$$\mathbf{\Phi}_k|\eta \sim Dir(\eta), \tag{5.13}$$

where $\eta \in \mathcal{R}^N$ is the hyper parameter.

When user m arrives to interact with the system at time t, one of K topics, denoted as $z_{m,t}$, is first selected according to the user's latent preference $\mathbf{p}_m$, indicating that user m shows interest in the topic $z_{m,t}$ at this moment. Accordingly, $z_{m,t}$ is supposed

to follow a multinomial distribution governed by $\mathbf{p}_m$ as follows,

$$z_{m,t}|\mathbf{p}_m \sim Mult(\mathbf{p}_m). \tag{5.14}$$

W.L.O.G, we assume

$$z_{m,t} = k$$

, and then the item distribution for topic k (i.e., $\mathbf{\Phi}_k$) is for generating item $x_{m,t}$ recommended to user m at time t. We assume the random variable $x_{m,t}$ follows the multinominal distribution ruled by $\mathbf{\Phi}_k$, i.e.,

$$x_{m,t}|\mathbf{\Phi}_k \sim Mult(\mathbf{\Phi}_k). \tag{5.15}$$

W.L.O.G, item n is assumed to be selected by user m at time t (i.e., $x_{m,t} = n$) where the latent vector corresponding to item n is $\mathbf{q}_n$. Let $y_{m,t}$ be the predicted reward (i.e., rating), given by user m at time t. The predicted reward $y_{m,t}$ can be inferred by

$$y_{m,t} \sim \mathcal{N}(\mathbf{p}_m^\intercal \mathbf{q}_n, \sigma_n^2). \tag{5.16}$$

By Equation (5.16), the rewards of different items are predicted. Based on the predicted rewards, the policy π selects an item and recommends it to user m, considering the tradeoff between exploitation and exploration. After consuming the recommended item, the system receives the actual reward $r_{m,t}$ from user m. The objective of the model is to maximize the expected accumulative rewards in a long run as described in Equation (5.5).

In this section, taking the clusters of arms into account, we formally introduced our ICTR model, which integrates matrix factorization with topic modeling in the bandit setting. We develop our solution to infer ICTR model from a Bayesian perspective in the following section.

5.3 Methodology and Solution

In this section, we present the methodology for online inferences of ICTR model.

The posterior distribution inference involves five random variables, i.e., $\mathbf{p}_m$, $z_{m,t}$, $\mathbf{\Phi}_k$, $\mathbf{q}_n$, and σ_n^2. According to the graphical model in Figure 5.3, the five random variables belong to two categories: parameter random variable and latent state random variable. $\mathbf{\Phi}_k$, $\mathbf{p}_m$, $\mathbf{q}_n$, and σ_n^2 are parameter random variables since they are assumed to be fixed but unknown, and their values do not change with time. Instead, $z_{m,t}$ is referred to as a latent state random variable since it is not observable and its value is time dependent. After pulling arm $n(t)$, where

$$n(t) = x_{m,t}$$

according to Equation (5.15) at time t, a reward is observed as $r_{m,t}$. Thus, $x_{m,t}$ and $r_{m,t}$ are referred to as observed random variables.

Our goal is to infer both latent parameter variables and latent state random variables to sequentially fit the observed data at time $t-1$, and predict the rewards for arm selection with respect to the incoming user at time t. However, since the inference of our model cannot be conducted by a simple closed-form solution, we adopt the sequential sampling-based inference strategy that is widely used in sequential Monte Carlo sampling [SDdFG13], particle filtering [DKZ+03], and particle learning [CJLP10] to learn the distribution of both parameter and state random variables. Specifically, particle learning that allows both state filtering and sequential parameter learning simultaneously is a perfect solution to our proposed model inference. In order to develop the solution based on particle learning, we first define a particle as follows.

Definition 5.3.1 (Particle) *A particle for predicting the reward $y_{m,t}$ is a container that maintains the current status information for both user m and item $x_{m,t}$. The*

status information comprises of random variables such as $\mathbf{p}_m$, σ_n^2, $\mathbf{\Phi}_k$, $\mathbf{q}_n$, *and* $z_{m,t}$, *as well as the hyper parameters of their corresponding distributions, such as* λ, α, β, η, $\mu_{\mathbf{q}}$ *and* $\mathbf{\Sigma}_{\mathbf{q}}$.

In particle learning, each particle corresponds to a sample for modeling inference status information. At each time stamp, particles are re-sampled according to their fitness to the current observable data. Then, the re-sampled particles are propagated to new particles and obtain the status information for the next time stamp. In the following subsections, we develop our solution based on particle learning.

5.3.1 Re-sample Particles with Weights

At time $t-1$, a fixed-size set of particles is maintained for the reward prediction for each arm $n(t-1)$ given user m. We denote the particle set at time $t-1$ as $\mathcal{P}_{m,n(t-1)}$ and assume the number of particles in $\mathcal{P}_{m,n(t-1)}$ is B. Let $\mathcal{P}^{(i)}_{m,n(t-1)}$ be the i^{th} particles given both user m and item $n(t-1)$ at time $t-1$, where $1 \leq i \leq B$. Each particle $\mathcal{P}^{(i)}_{m,n(t-1)}$ has a weight, denoted as $\rho^{(i)}$, indicating its fitness for the new observed data at time t. Note that

$$\sum_{i=1}^{B} \rho^{(i)} = 1$$

. The fitness of each particle $\mathcal{P}^{(i)}_{m,n(t-1)}$ is defined as the likelihood of the observed data $x_{m,t}$ and $r_{m,t}$. Therefore,

$$\rho^{(i)} \propto p(x_{m,t}, r_{m,t} | \mathcal{P}^{(i)}_{m,n(t-1)}). \tag{5.17}$$

Further, $y_{m,t}$ is the predicted value of $r_{m,t}$. The distribution of $y_{m,t}$, determined by $\mathbf{p}_m$, $\mathbf{q}_n$, $z_{m,t}$, $\mathbf{\Phi}_k$, and σ_n^2, has been described in Section 5.2.2.

Therefore, we can compute $\rho^{(i)}$ as proportional to the density value given

$$y_{m,t} = r_{m,t}$$

and

$$x_{m,t} = n.$$

Thus, we obtain

$$\rho^{(i)} \propto \sum_{z_{m,t}=1}^{K} \{\mathcal{N}(r_{m,t}|(\mathbf{p}_m^{\intercal}\mathbf{q}_n, \sigma_n^2) \bullet p(z_{m,t} = k, x_{m,t} = n|\mathcal{P}^{(i)}_{m,n(t-1)})\},$$

where

$$\begin{aligned} & p(z_{m,t} = k, x_{m,t} = n|\mathcal{P}^{(i)}_{m,n(t-1)}) \\ & = \iint_{\mathbf{p}_m, \mathbf{\Phi}_k} p(z_{m,t} = k, x_{m,t} = n, \mathbf{p}_m, \mathbf{\Phi}_k|\lambda, \eta) d\mathbf{p}_m d\mathbf{\Phi}_k \\ & = \int_{\mathbf{p}_m} Mult(z_{m,t} = k|\mathbf{p}_m) Dir(\mathbf{p}_m|\lambda) d\mathbf{p}_m \\ & \bullet \int_{\mathbf{\Phi}_k} Mult(x_{m,t} = n|\mathbf{\Phi}_k) Dir(\mathbf{\Phi}_k|\eta) d\mathbf{\Phi}_k \\ & = E(\mathbf{p}_{m,k}|\lambda) \bullet E(\mathbf{\Phi}_{k,n}|\eta). \end{aligned} \tag{5.18}$$

Thus, we have:

$$\rho^{(i)} \propto \sum_{z_{m,t}=1}^{K} \{\mathcal{N}(\mathbf{r}_{m,t}|(\mathbf{p}_m^{\intercal}\mathbf{q}_n, \sigma_n^2) \bullet E(\mathbf{p}_{m,k}|\lambda) \bullet E(\mathbf{\Phi}_{k,n}|\eta)\}, \tag{5.19}$$

where $E(\mathbf{p}_{m,k}|\lambda)$ and $E(\mathbf{\Phi}_{k,n}|\eta)$ represent the conditional expectations of $\mathbf{p}_{m,k}$ and $\mathbf{\Phi}_{k,n}$ given the observed reward λ and η of $\mathcal{P}^{(i)}_{m,n(t-1)}$. The expectations can be inferred by

$$E(\mathbf{p}_{m,k}|\lambda) = \frac{\lambda_k}{\sum_{k=1}^{K} \lambda_k}$$

and

$$E(\mathbf{\Phi}_{k,n}|\eta) = \frac{\eta_{k,n}}{\sum_{n=1}^{N} \eta_{k,n}}.$$

Before updating any parameters, a re-sampling process is conducted. We replace the particle set $\mathcal{P}_{m,n(t-1)}$ with a new set $\mathcal{P}_{m,n(t)}$, where $\mathcal{P}_{m,n(t)}$ is generated from $\mathcal{P}_{m,n(t-1)}$ using sampling with replacement based on the weights of particles. Then sequential parameter updating is based on $\mathcal{P}_{m,n(t)}$.

5.3.2 Latent State Inference

Provided with the new observation $x_{m,t}$ and $r_{m,t}$ at time t, the random state $z_{m,t}$ can be one of K topics and the posterior distribution of $z_{m,t}$ is shown as follows:

$$z_{m,t}|x_{m,t}, r_{m,t}, \mathcal{P}^{(i)}_{m,n(t-1)} \sim Mult(\theta), \tag{5.20}$$

where $\theta \in \mathcal{R}^K$ is the parameter specifying the multinominal distribution. According to Equation (5.18), since

$$p(z_{m,t}|x_{m,t}, r_{m,t}, \lambda, \eta) \propto p(z_{m,t}, x_{m,t}|r_{m,t}, \lambda, \eta),$$

θ can be computed by

$$\theta_k \propto E(\mathbf{p}_{m,k}|r_{m,t}, \lambda) \bullet E(\mathbf{\Phi}_{k,n}|r_{m,t}, \eta)$$

. Further, $E(\mathbf{p}_{m,k}|r_{m,t}, \lambda)$ and $E(\mathbf{\Phi}_{k,n}|r_{m,t}, \eta)$ can be obtained as follows,

$$\begin{aligned} E(\mathbf{p}_{m,k}|r_{m,t}, \lambda) &= \frac{\mathcal{I}(z_{m,t} = k)r_{m,t} + \lambda_k}{\sum_{k=1}^{K}[\mathcal{I}(z_{m,t} = k)r_{m,t} + \lambda_k]}, \\ E(\mathbf{\Phi}_{k,n}|r_{m,t}, \eta) &= \frac{\mathcal{I}(x_{m,t} = n)r_{m,t} + \eta_{k,n}}{\sum_{n=1}^{N}[\mathcal{I}(x_{m,t} = n)r_{m,t} + \eta_{k,n}]}. \end{aligned} \tag{5.21}$$

where $\mathcal{I}(\bullet)$, an indicator function, returns 1 when the input boolean expression is true, otherwise returns 0. Specifically, if $r_{m,t} \in \{0, 1\}$, the value of $r_{m,t}$ indicates whether $x_{m,t}$ should be included in the preferred item list of user m. If $r_{m,t} \in [0, +\infty)$, the value of $r_{m,t}$ implies how user m likes item $x_{m,t}$. Therefore, our solution can effectively handle the non-negative rating score at different scales.

5.3.3 Parameter Statistics Inference

At time $t-1$, the sufficient statistics for the parameter random variables ($\mathbf{q}_n$, σ_n^2, $\mathbf{p}_m$, $\mathbf{\Phi}_k$) are ($\mu_{\mathbf{q}}$, $\mathbf{\Sigma}_{\mathbf{q}}$, α, β, λ, η). Assume $\mu'_{\mathbf{q}}$, $\mathbf{\Sigma}'_{\mathbf{q}}$, α', β', λ', and η' are the sufficient

statistics at time t, which are updated on the basis of both the sufficient statistics at time $t-1$ and the new observation data (i.e, $x_{m,t}$ and $r_{m,t}$). The sufficient statistics for parameters are updated as follows:

$$\begin{aligned}
&\mathbf{\Sigma}'_{\mathbf{q}_n} = (\mathbf{\Sigma}_{\mathbf{q}_n}^{-1} + \mathbf{p}_m \mathbf{p}_m^{\intercal})^{-1}, \\
&\mu'_{\mathbf{q}_n} = \mathbf{\Sigma}'_{\mathbf{q}_n}(\mathbf{\Sigma}_{\mathbf{q}_n}^{-1}\mu_{\mathbf{q}_n} + \mathbf{p}_m r_{m,t}), \\
&\alpha' = \alpha + \frac{1}{2}, \\
&\beta' = \beta + \frac{1}{2}(\mu_{\mathbf{q}_n}^{\intercal}\mathbf{\Sigma}_{\mathbf{q}_n}^{-1}\mu_{\mathbf{q}_n} + r_{m,t}^{\intercal} r_{m,t} - \mu_{\mathbf{q}_n}'^{\intercal}\mathbf{\Sigma}_{\mathbf{q}_n}'^{-1}\mu'_{\mathbf{q}_n}), \\
&\lambda'_k = \mathcal{I}(z_{m,t} = k) r_{m,t} + \lambda_k, \quad \eta'_{k,n} = \mathcal{I}(x_{m,t} = n) r_{m,t} + \eta_{k,n}.
\end{aligned} \tag{5.22}$$

At time t, the sampling process for the parameter random variables σ_n^2, $\mathbf{q}_n$, $\mathbf{p}_m$ and $\mathbf{\Phi}_k$ is summarized as below:

$$\begin{aligned}
&\sigma_n^2 \sim \mathcal{IG}(\alpha', \beta'), \\
&\mathbf{q}_n | \sigma_n^2 \sim \mathcal{N}(\mu'_{\mathbf{q}_n}, \sigma_n^2 \mathbf{\Sigma}'_{\mathbf{q}_n}), \\
&\mathbf{p}_m \sim Dir(\lambda'), \\
&\mathbf{\Phi}_k \sim Dir(\eta').
\end{aligned} \tag{5.23}$$

5.3.4 Integration with Policies

In our ICTR model, when user m arrives at time t, reward $r_{m,t}$ is unknown since it is not observed until one of arms $x_{m,t}$ is pulled. Without observed $x_{m,t}$ and $r_{m,t}$, the particle re-sampling, latent state inference, and parameter statistics inference for time t cannot be conducted. Therefore, we utilize the latent vectors $\mathbf{p}_m$ and $\mathbf{q}_n$, sampled from their corresponding posterior distributions by Equation (5.23) at time $t-1$, to predict the reward for each arm. In this section, two policies based on Thompson sampling and UCB for ICF are integrated with our model.

In the model, every item has B independent particles given user m. Each particle i contains its latent variables and parameters, and produces an independent reward prediction $r_{m,t}^{(i)}$. Specifically, according to Thompson sampling discussed in Section 5.2.1,

we predict the reward of pulling arm n with the average value of rewards from B particles. The policy based on Thompson sampling selects an arm $n(t)$ based on the following equation,

$$n(t) = \arg\max_n (\bar{r}_{m,n}), \tag{5.24}$$

where $\bar{r}_{m,n}$ denotes the average reward, i.e.,

$$\bar{r}_{m,n} = \frac{1}{B}\sum_{i=1}^{B} \mathbf{p}_m^{(i)\intercal}\mathbf{q}_n^{(i)}.$$

Moreover, UCB policy selects an arm based on the upper bound of the predicted reward. Assuming that

$$r_{m,t}^{(i)} \sim \mathcal{N}(\mathbf{p}_m^{(i)\intercal}\mathbf{q}_n^{(i)}, \sigma^{(i)2}),$$

the UCB-based policy is developed by the mean and variance of predicted reward, i.e.,

$$n(t) = \arg\max_n (\bar{r}_{m,n} + \gamma\sqrt{\nu}), \tag{5.25}$$

where $\gamma \geq 0$ is a predefined threshold and the variance is expressed as

$$\nu = \frac{1}{B}\sum_{i}^{B} \sigma^{(i)2}.$$

5.3.5 Algorithm

Putting all the aforementioned inference together, an algorithm for ICTR model is provided below.

Online inference for ICF problem starts with MAIN procedure presented in Algorithm 2. As user m arrives at time t, EVAL procedure computes a score for each arm, where we define the score as the average reward. The arm with the highest score is selected to be pulled. After receiving a reward by pulling an arm, the new feedback is used to update ICTR model by UPDATE procedure. Especially in UPDATE procedure, we use the *resample-propagate* strategy in particle learning [CJLP10]

rather than the *propagate-resample* strategy in particle filtering [DKZ+03]. With the *resample-propagate* strategy, particles are re-sampled by taking $\rho^{(i)}$ as the i^{th} particle's weight, where the $\rho^{(i)}$ indicates the fitness of the observation at time t given the particle at time $t-1$. The *resample-propagate* strategy is considered as an optimal and fully adapted strategy avoiding an importance sampling step.

Algorithm 2 The algorithms for `ICTR` model

procedure MAIN(B) ▷ Main entry.
  Initialize B particles, i.e., $\mathcal{P}^{(1)}_{m,n(0)} ... \mathcal{P}^{(B)}_{m,n(0)}$.
  for $t \leftarrow 1, T$ **do**
    User m arrives for item recommendation.
    $n(t) = \arg\max_{n=1,N} \text{EVAL}(m, n)$ by Equation (5.24) or Equation (5.25).
    Receive $r_{m,t}$ by rating item $n(t)$.
    UPDATE(m, $n(t)$, $r_{m,t}$).
  end for
end procedure

procedure EVAL(m, n) ▷ Get a rating score for item n, given user m.
  for $i \leftarrow 1, B$ **do** ▷ Iterate on each particle.
    Get the user latent vector $\mathbf{p}_m^{(i)}$.
    Get the item latent vector $\mathbf{q}_n^{(i)}$.
    Predict i^{th} reward $r_{m,t}^{(i)}$.
  end for
  Compute the average reward as the final reward $r_{m,t}$.
  return the score.
end procedure

procedure UPDATE(m, $n(t)$, $r_{m,t}$) ▷ Update the inference.
  for $i \leftarrow 1, B$ **do** ▷ Compute weights for each particle.
    Compute weight $\rho^{(i)}$ of particle $\mathcal{P}^{(i)}_{m,n(t)}$ by Equation (5.17).
  end for
  Re-sample $\mathcal{P}'_{m,n(t)}$ from $\mathcal{P}_{m,n(t)}$ according to the weights $\rho^{(i)}$s.
  for $i \leftarrow 1, B$ **do** ▷ Update statistics for each particle.
    Update the sufficient statistics for $z_{m,t}$ by Equation (5.21).
    Sample $z_{m,t}$ according to Equation (5.20).
    Update the statistics for $\mathbf{q}_n$, σ_n^2, $\mathbf{p}_m$, $\mathbf{\Phi}_k$ by Equation (5.22).
    Sample $\mathbf{q}_n$, σ_n^2, $\mathbf{p}_m$, $\mathbf{\Phi}_k$ by Equation (5.23).
  end for
end procedure

In addition, existing algorithms [ZZW13, KBK+15] consider all the arms independently, while our model takes the clusters of arms into account by learning the topic-related random variables (e.g., Φ_k), which are shared among all the arms.

5.4 Empirical Study

To demonstrate the efficiency of our proposed algorithm, we conduct our experimental study over two popular real-world dataset: Yahoo! Today News and MovieLens (10M). First, we outline the general implementation of the baselines. Second, we start with a brief description of the datasets and evaluation method. Finally, we show and discuss the comparative experimental results of both the proposed algorithms and the baselines, and a case study on movie topic distribution analysis of MovieLens (10M). Methods in [GLZ14, WWGW16, WWW17, ZB16] are excluded from the baselines since our work is orthogonal to those methods.

5.4.1 Baseline Algorithms

In the experiment, we demonstrate the performance of our methods by comparing them with the following baseline algorithms:

1. `Random`: it randomly selects an item recommending to the target user.
2. ϵ-`greedy`(ϵ): it randomly selects an item with probability ϵ and selects the item of the largest predicted reward with probability $1 - \epsilon$, where ϵ is a predefined parameter.
3. `UCB`(λ): it picks item $j(t)$ with the highest rewards at time t as follows:

$$j(t) = \arg \max_{j=1,\ldots,N} (\hat{\mu_i} + \lambda \sqrt{\frac{2ln(t)}{n_{i(t)}}})$$

where $n_{i(t)}$ is the number of times that item n_i has been recommended until time t.

4. `TS`$(S_i(t), F_i(t))$: Thompson sampling described in Section 5.2.1, randomly draws the expected reward from the Beta posterior distribution, and selects the item with the largest predicted reward.

$$S_i(t)/F_i(t)$$

is the number of positive/negative feedback on item i until time t.

5. `PTS`$(d,\ p)$: particle Thompson sampling for matrix factorization approximates the posterior of latent feature vectors by updating a set of particles. Here d is the dimension of latent feature vector and p is the number of particles.

Our methods proposed in this chapter include:

1. `ICTRTS`$(d,\ p)$: it denotes our proposed interactive collaborative topic regression model with `TS`. Here d is the dimension of latent feature vector and p is the number of particles.

2. `ICTRUCB`$(d,\ p,\ \gamma)$: it indicates our proposed model with `UCB`. Similar to `UCB`, γ is given. Here d is the dimension of latent feature vector and p represents the number of particles.

All algorithms are implemented using Java 1.8. All empirical experiments are running on Linux 2.6.32. The server is equipped with Intel(R) Xeon(R) CPU with 24 cores running at speed of 2.50GHZ. The total volume of memory is 158GB.

5.4.2 Datasets Description

We use two real-world datasets shown in Table 5.2 to evaluate our proposed algorithms.

Table 5.2: Description of datasets.

Dataset	Yahoo News	MovieLens (10M)	Automation
#users or #alert keys	226,710	71,567	1,091
#items or #automations	652	10,681	62
#ratings	280,410,150	10,000,054	332,211

Yahoo! Today News: The core task of personalized news recommendation is to display appropriate news articles on the web page for users. The system often takes the user's instant feedback into account to improve the prediction of his/her preferences, where the user feedback is about whether he/she clicks the recommended article or not. Here, we formulate the personalized news recommendation problem as an instance of bandit problem, where each arm corresponds to a news article. The experimental dataset is a collection based on a sample of anonymized user interaction on the news feeds published by Yahoo! Research Lab[1]. The dataset contains 15 days' visit events of user-news item interaction data by randomly selecting news articles for recommendation. Besides, user's information (e.g., demographic information) is provided for each visit event and represented as the user identification, where users with the same information are identified as one user. In our experiments, the visit events of the first day are utilized for selecting proper parameters of ICTR model, while two million of the remaining are for the evaluation. Each interactive record in the historical logs consists of user ID, news article ID, rating feedback and a timestamp.

MovieLens (10M): Online movie recommender service aims to maximize the customer's satisfaction by recommending proper movies to target users according to their preferences. Specifically, several movies are selected out of a movie set and displayed to users, and then users' feedback on displayed movies are collected for improving the user satisfaction. Thereby, the problem of movie recommendation can

[1]http://webscope.sandbox.yahoo.com/catalog.php

be formulated as a bandit problem where an arm is a movie, a pull is regarded as a movie selection, and the reward is indicated by the user's rating on the recommended movie. In our experiments, each rating associates user ID, movie ID, and a timestamp. In order to use the *replayer* evaluation method, we assume the rating data is produced by the users when the movies are randomly recommended. The rating score in the dataset ranges from 1 to 5. Additionally, we choose the top-N (N=100) popular movies to form a movie set, from which one movie is recommended to a user by algorithms in every trial.

Automation: The dataset is collected by IBM Tivoli Monitoring system [urlf] from July 2016 to March 2017, which contains 332,211 historical records. After filtering out those unqualified ones for recommendation algorithms, $116,429$ records are available for empirical studies. The dataset contains 1,091 alert keys (e.g., c-pusum_xuxc_aix, prccpu_rlzc_std) and 62 automations (e.g., NFS automation, process CPU spike automation) in total. The execution feedback (i.e., reward or rating) indicating whether the ticket has been resolved by an automation or needs to be escalated to human engineers, is collected and utilized for our proposed model inference. Each record is stamped with the reporting time of the ticket.

5.4.3 Evaluation Method and Metrics

The evaluation methods for traditional non-interactive recommender systems assume the independence among the items at different time stamps once the offline model is built. In an online interactive recommender system, the recommended items at previous time stamps are used to update the recommendation model, and then further effect the recommendation items at current time stamp.

We apply the *replayer* method to evaluate our proposed algorithms on the aforementioned two datasets. The *replayer* method, first introduced in [LCL+12], provides

an unbiased offline evaluation for multi-armed bandit algorithms via historical logs, where the logs are assumed to be generated by random recommendation. The main idea of *replayer* is to replay each user visit in the historical logs to the algorithm under evaluation. If the recommended item by the testing algorithm is identical to the one in the historical log, this visit is considered as a match between the historical recommendation and the testing recommendation algorithm. The *replayer* method only counts those matched visits in for the accumulated reward computation. Since the recommendation algorithms may result in different numbers of matched visits, the average reward (i.e., the accumulated rewards divided by the number of matched visits) is adopted for evaluation.

Particularly, in the scenario of news article recommendation, a matched visit corresponds to an impression, and a reward of one is obtained by a click, so the average reward also represents the average CTR (Click Through Rate). In the scenario of movie recommendation, we set the reward of one if the rating score of the recommended movie is no less than four, indicating that the user likes the recommended movie. If the rating is less than four, a reward value of zero is obtained. Thus, the average reward in this scenario indicates the success rate of movie recommendation. To sum it up, in our setting, the reward is one if the recommended article (movie) is clicked (liked), otherwise it is zero.

Table 5.3: Evaluation metric computation for *replayer*.

		Item in Random Recommendation Logs	
		Clicked/Liked	Not Clicked/Not Liked
Recommended Item	Matched	TP	FP
	Not Matched	N/A	N/A

Consider a matched visit shown in Table 5.3. If the item in the logs is clicked or liked by a user, the recommended item is referred to as a true positive (TP), otherwise it is referred to as a false positive (FP). The average reward is computed

as $\frac{TP*1+FP*0}{TP+FP} = \frac{TP}{TP+FP}$, corresponding to the formula for the precision for matched visits. However, for the unmatched visits, we can not determine whether an item is false negative or true negative since no ground truth is provided. Therefore, the computation of the average reward in this case as the recall, relying on the false negative, is not feasible.

5.4.4 Recommendation Evaluation

In this section we first conduct the *replayer* evaluation method for each algorithm with different parameter settings. The aforementioned average reward is used as the performance metric in the experiments.

All baseline algorithms are configured with different parameter settings provided in Table 5.4. The settings of all algorithms with the highest average reward are highlighted in bold. Our algorithm `ICTRUCB(2,10,1.0)` achieves the best performance among all algorithms on Yahoo! Today News, and the performance comparisons among different algorithms along different time buckets are illustrated in Figure 5.4. For MovieLens (10M), `ICTRTS(3,10)` outperforms all others and the corresponding performance comparisons are shown in Figure 5.5. `ICTRTS(5,3)` has the best performance during recommending the most matched automation presented in Figure 5.6.

Our proposed algorithms outperform the baseline algorithms using independent arms because ICTR model can leverage the dependencies among items by clustering items (arms) using items' latent aspects. The feedback received after recommending an item is not only used to update the model parameters related to this item, but also utilized to refine the parameters for the item's cluster. As a result, the updated cluster parameters further influence the model's parameter inference for other items within the same cluster. The effect of the clustering is illustrated in more details in the next section.

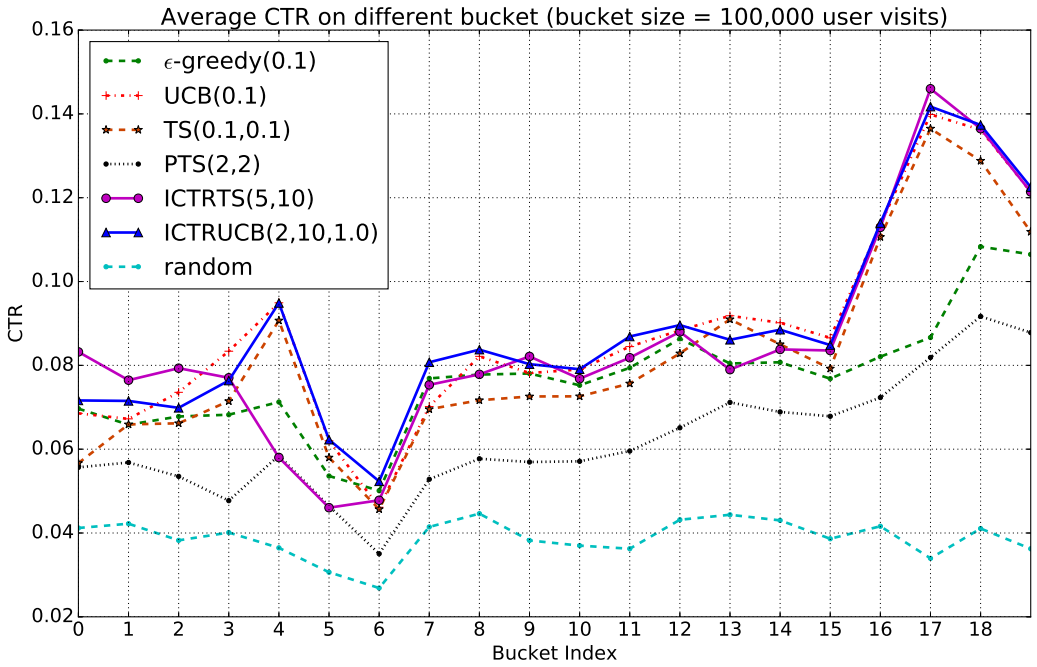

Figure 5.4: The average CTR of Yahoo! Today News data is given along each time bucket. All algorithms shown here are configured with their best parameter settings.

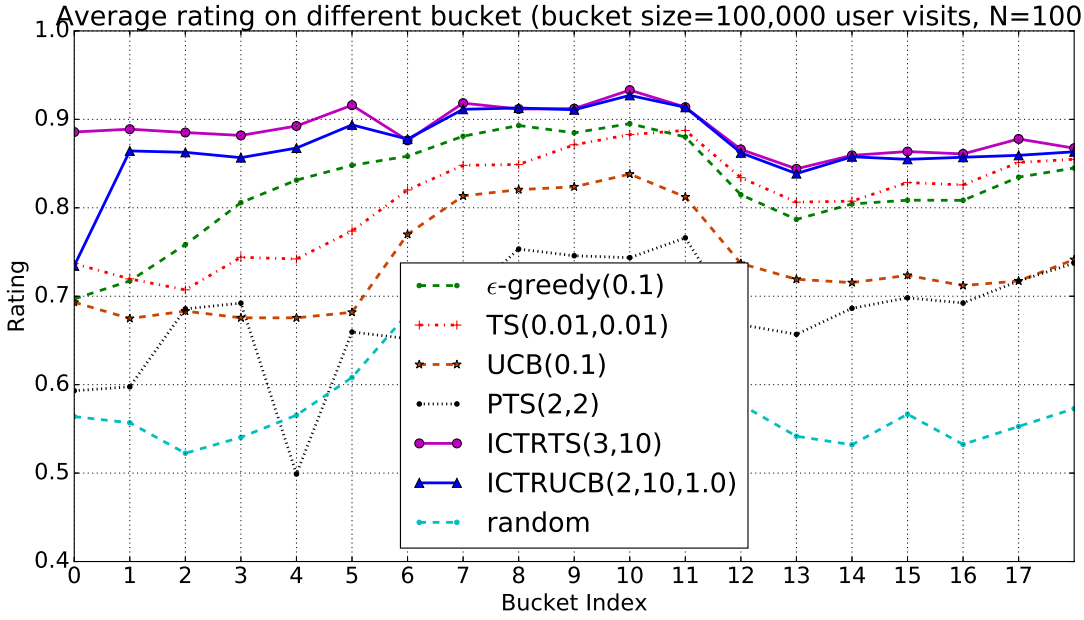

Figure 5.5: The average rating of MovieLens (10M) data is given along each time bucket. All algorithms shown here are configured with their best parameter settings.

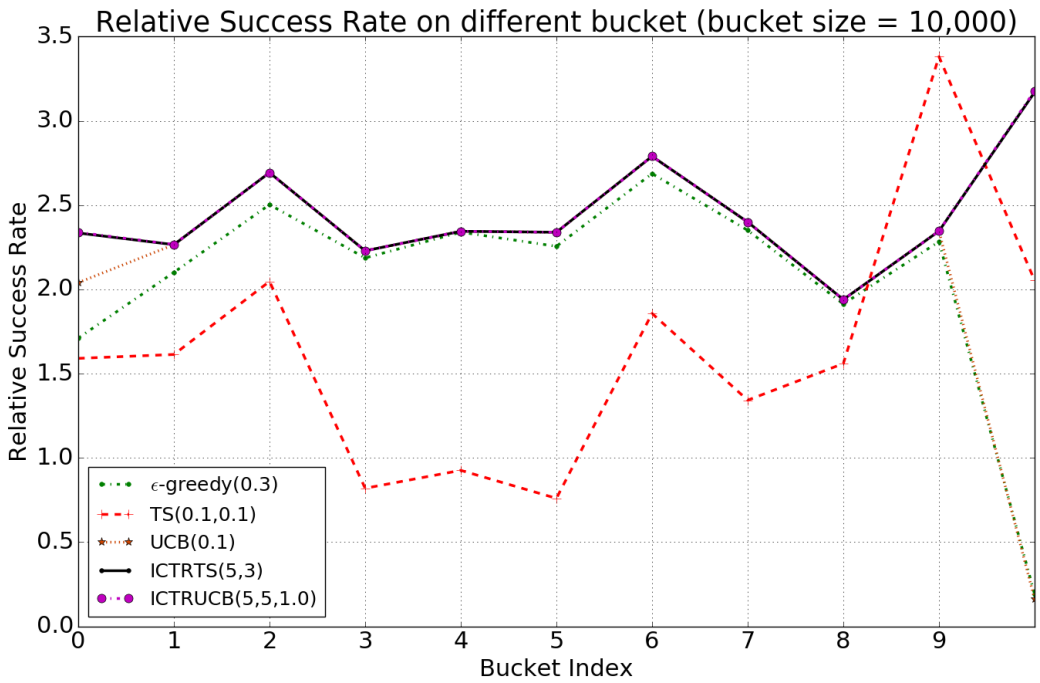

Figure 5.6: The relative average rating of Automation data is given along each time bucket. All algorithms shown here are configured with their best parameter settings.

5.4.5 A Case Study: Topic Distribution Analysis on MovieLens (10M)

We conduct an experiment to demonstrate that our model can effectively capture the dependency between items, i.e., finding the latent topics among movies and clustering similar movies together. In this experiment, top-N (N=8) popular movies are selected and topic number (K=2) is set for our model. After millions of training iterations, the learned latent movie feature vectors will represent each movie's topic distribution over the two latent topics, in which the i-th dimension of the feature vector encodes the probability that the movie belongs to the i-th movie topic cluster. We separately choose four movies with the highest value of the first element and the second element of these latent feature vectors, and list their IDs, names, and movie types in Table 5.5,

Table 5.4: Average CTR/rating on two real world datasets.

Algorithm	**Yahoo! Today News**				**MovieLens (10M)**			
	mean	std	min	max	mean	std	min	max
ϵ-greedy(0.01)	0.06916	0.00312	0.06476	0.07166	0.70205	0.06340	0.60752	0.78934
ϵ-greedy(0.1)	**0.07566**	0.00079	0.07509	0.07678	**0.82038**	0.01437	0.79435	0.83551
ϵ-greedy(0.3)	0.07006	0.00261	0.06776	0.07372	0.80447	0.01516	0.77982	0.82458
ϵ-greedy(1.0)	0.03913	0.00051	0.03842	0.03961	0.60337	0.00380	0.59854	0.60823
UCB(0.01)	0.05240	0.00942	0.04146	0.06975	0.62133	0.10001	0.45296	0.73369
UCB(0.1)	**0.08515**	0.00021	0.08478	0.08544	**0.73537**	0.07110	0.66198	0.85632
UCB(0.5)	0.05815	0.00059	0.05710	0.05893	0.71478	0.00294	0.63623	0.64298
UCB(1.0)	0.04895	0.00036	0.04831	0.04932	0.63909	0.00278	0.60324	0.61296
TS(0.01,0.01)	0.07853	0.00058	0.07759	0.07921	**0.83585**	0.00397	0.82927	0.84177
TS(0.1,0.1)	**0.07941**	0.00040	0.07869	0.07988	0.83267	0.00625	0.82242	0.84001
TS(0.5,0.5)	0.07914	0.00106	0.07747	0.08041	0.82988	0.00833	0.81887	0.84114
TS(1.0,1.0)	0.07937	0.00079	0.07788	0.08044	0.83493	0.00798	0.82383	0.84477
PTS(2,2)	**0.06069**	0.00575	0.05075	0.06470	**0.70484**	0.03062	0.64792	0.74610
PTS(2,10)	0.05699	0.00410	0.05130	0.06208	0.65046	0.01124	0.63586	0.66977
PTS(5,10)	0.05778	0.00275	0.05589	0.06251	0.63777	0.00811	0.62971	0.65181
PTS(5,20)	0.05726	0.00438	0.05096	0.06321	0.62289	0.00714	0.61250	0.63567
PTS(10,20)	0.05490	0.00271	0.05179	0.05839	0.61819	0.01044	0.60662	0.63818
ICTRTS(2,5)	0.06888	0.00483	0.06369	0.07671	0.70386	0.15772	0.48652	0.85596
ICTRTS(2,10)	0.06712	0.01873	0.03731	0.08487	0.56643	0.10242	0.42974	0.67630
ICTRTS(3,10)	0.06953	0.00783	0.05857	0.07804	**0.88512**	0.00052	0.88438	0.88553
ICTRTS(5,10)	**0.08321**	0.08236	0.08492	0.06292	0.55748	0.14168	0.38715	0.73404
ICTRTS(7,10)	0.05066	0.00885	0.04229	0.06423	0.517826	0.07120	0.42297	0.59454
ICTRTS(7,20)	0.04925	0.00223	0.04672	0.05285	0.61414	0.12186	0.44685	0.73365
ICTRUCB(2,10,0.01)	0.06673	0.01233	0.04588	0.08112	0.44650	0.06689	0.38678	0.53991
ICTRUCB(2,10,1.0)	**0.08597**	0.00056	0.08521	0.08675	**0.86411**	0.01528	0.85059	0.88547
ICTRUCB(3,10,0.05)	0.07250	0.00426	0.06799	0.07694	0.54757	0.13265	0.43665	0.73407
ICTRUCB(3,10,1.0)	0.08196	0.00296	0.07766	0.08530	0.57805	0.08716	0.46453	0.67641
ICTRUCB(5,10,0.01)	0.07009	0.00722	0.06411	0.08244	0.62282	0.02572	0.59322	0.65594
ICTRUCB(5,10,1.0)	0.08329	0.00140	0.08098	0.08481	0.80038	0.24095	0.29625	0.88554

which clearly proves our assumption that the model is able to capture the dependency between items and cluster similar movies together.

Table 5.5: Movie topic distribution of MovieLens (10M).

Topic Cluster I			**Topic Cluster II**		
MovieId	MovieName	MovieType	MovieId	MovieName	MovieType
32	12 Monkeys	Sci-Fi,Thriller	344	Pet Detective	Comedy
50	Usual Suspects	Crime,Mystery,Thriller	588	Aladdin	Children,Animation,Comedy
590	Dances with wolves	Adventure,Drama,Western	595	Beauty and the Beast	Animation,Children,Musical
592	Batman	Action,Crime,Sci-Fi,Thriller	2857	Yellow Submarine	Adventure,Animation,Comedy,Musical

5.4.6 A Case Study: Identify the Category of a New Automation

Another case study is carried out with an attempt to identify the category of an automation named "process missing" using ICTR model. Based on our previous discussion, ICTRTS $(5, 3)$ can achieve the best performance on this dataset, where the dimension of the latent feature vector is 5 and the number of particles is 3. Through iteratively running on the rating dataset, ICTRTS $(5, 3)$ fully learns the latent feature vector of each automation as well as performs the recommendation. Four automations are randomly selected from the automation pool and listed in Figure 5.7. Different from the automation "process missing", the other four automations can be easily figured out their correct categories according to their names. We compute the Euclidean distances between the latent feature vector of "process missing" and the ones of the other four categorized automations. We consider it as an automation related to "WINDOWS" category as the minimum distance indicates in Figure 5.7. The correctness of categorization is verified by the domain experts, which demonstrates ICTR's capability of discovering the automation categories by clustering the learned latent features.

UNCATEGORIZED AUTOMATION	process missing	
CATEGORIZED AUTOMATION	**CATEGORY**	**EUCLIDEAN DISTANCE**
(1) db2 percent db connection executing is to high automation	DATABASE	1.086
(1) process cpu spike automation	UNIX	1.014
(2) swap automation		0.858
(1) windows service automation	**WINDOWS***	**0.565***

Figure 5.7: An example of categorizing an automation.

5.4.7 Time Cost

The cumulative time cost of each algorithm on Yahoo! Today News data and MovieLens (10M) datasets is presented in Figure 5.8 and Figure 5.9, where all algorithms are configured with their best parameter settings, since the size of Automation data is not large enough. Our proposed algorithms have higher running time since they needs to learn the latent features for arms. However, the computational complexity of both `ICTRUCB(1,1,1.0)` and `ICTRTS(1,1)` is comparable to the baselines'. We also evaluate the time costs of `ICTRTS` and `ICTRUCB` with different number of particles and latent feature vector dimensions on the two datasets (see Figure 5.10 and Figure 5.11). It shows that the time cost grows linearly with the number of particles and dimensions of latent feature vector.

The observations can be summarized as follows: (1) MovieLens (10M) requires much more time than Yahoo! Today New due to a larger amount of items and users. (2) In general, `UCB`-based algorithms (e.g., `ICTRUCB`, `UCB`) are faster than `TS`-based ones (e.g., `ICTRTS`, `PTS`) since the `TS`-based algorithms highly depend on the sampling process.

5.5 Summary

In this chapter, we propose an interactive collaborative topic regression model that adopts a generative process based on topic model to explicitly formulate the arm dependencies as the clusters on arms, where dependent arms are assumed to be generated from the same cluster. Every time an arm is pulled, the feedback is not only used for inferring the involved user and item latent vectors, but also employed to update the latent parameters with respect to the arm's cluster. The latent cluster parameters further help with the reward prediction for other arms in the same cluster.

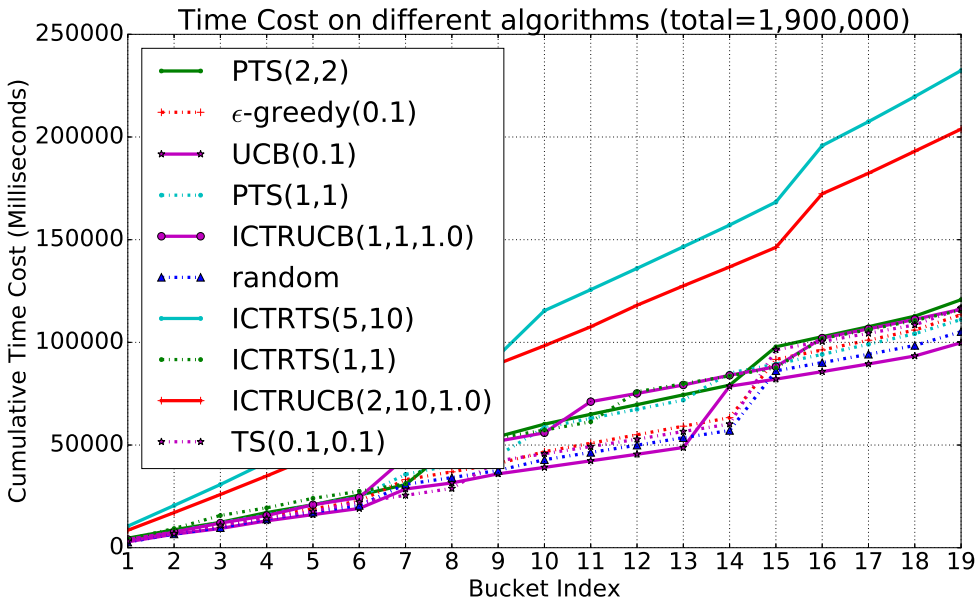

Figure 5.8: Cumulative time cost of Yahoo! Today News is given along each time bucket.

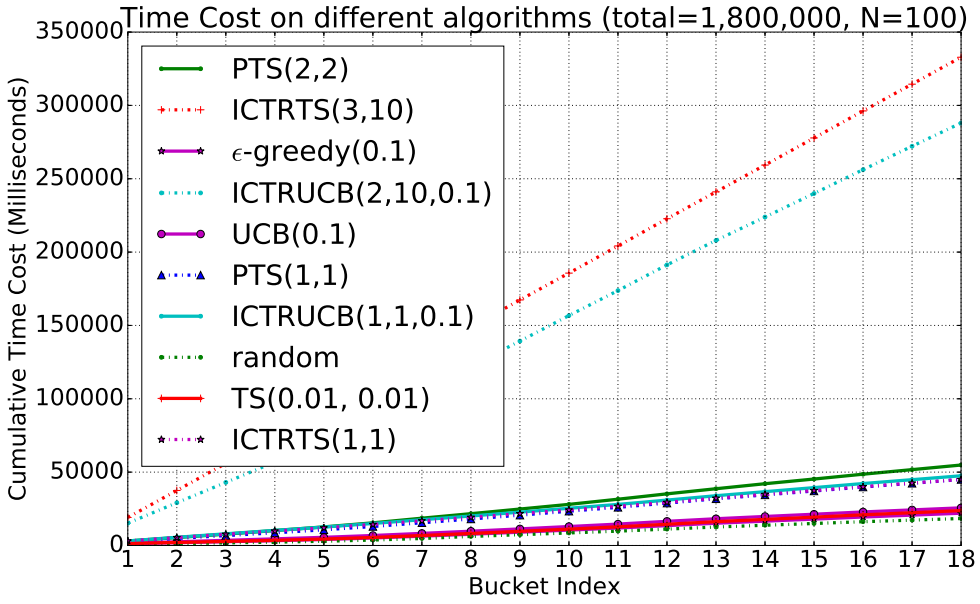

Figure 5.9: Cumulative time cost of MovieLens (10M) is given along each time bucket.

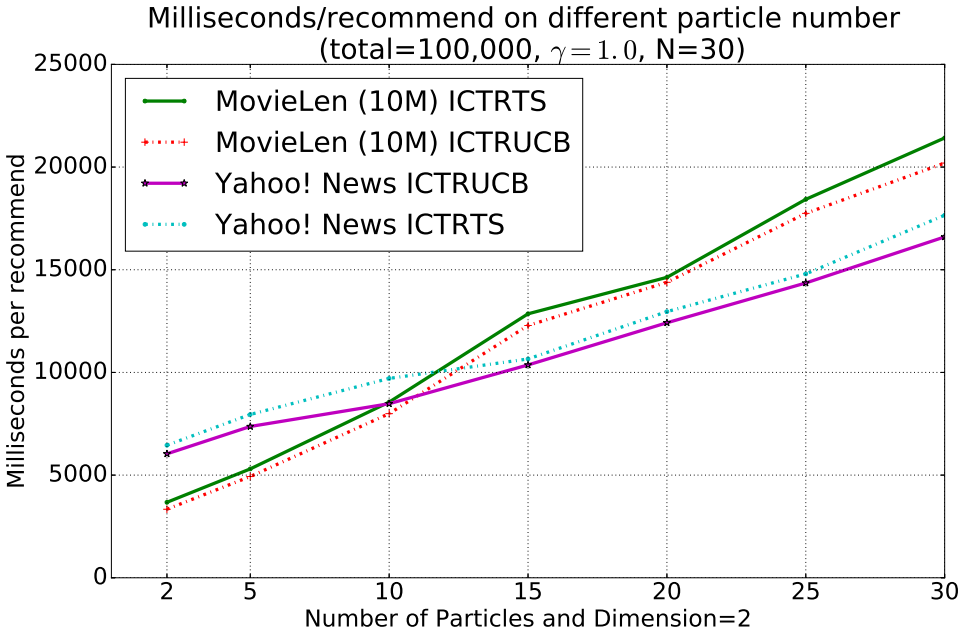

Figure 5.10: Time cost is given with different number of particles.

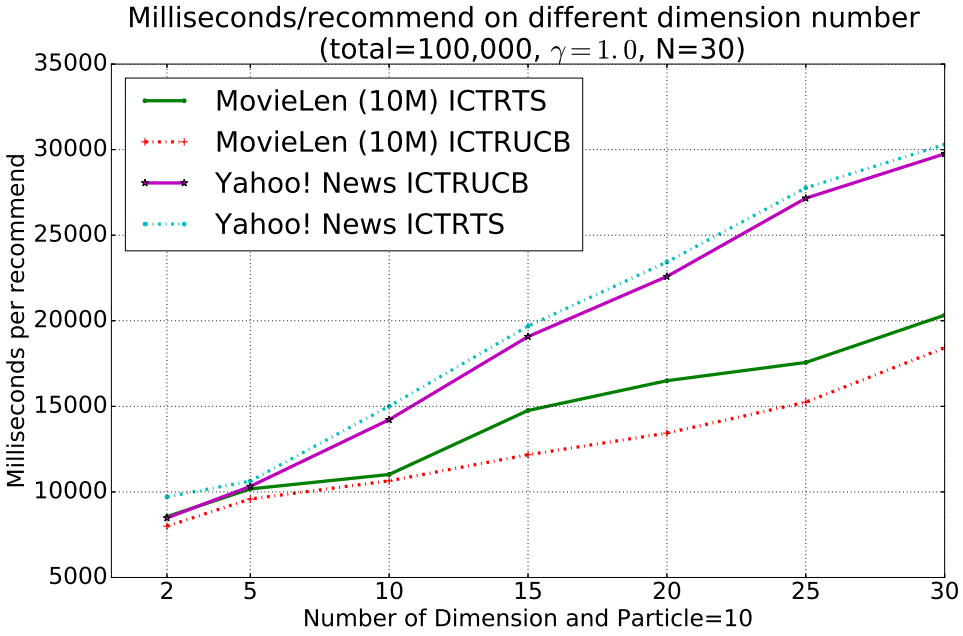

Figure 5.11: Time cost is given with different number of latent feature vector dimensions.

We conduct empirical studies on three real-world applications and the experimental results demonstrate the effectiveness of our proposed approach.

We also inspected the current procedures in IT automation services and extend the ICTR model to solve the problems in real IT environment, where implicit automation dependencies can be effectively exploited as well as promptly suggest the most matched automations for resolving a ticket problem. Empirical studies in real IT environment are conducted to show the advantages of our solutions. In addition, we would like to provide a comprehensive regret analysis [GLK+17] of our model in the future work.

CHAPTER 6

THE FUTURE OF AI-BASED SERVICE MANAGEMENT

In the previous chapters, my efforts are aiming at empowering the automated IT service management processes certain AI capability. Constructing the domain knowledge base would make it capable of learning past experiences from both human engineers and virtual engineers. For example, updating its knowledge database and ultimately improving the performance that how it reacts to any issues, even some of them are completely new to the system. Introducing multi-armed bandit model into IT automation services would enable it automatically learn the underlying mapping function between ticket problem symptoms and resolutions from virtual engineers through the up-to-date feedback from the problem servers. In recent years, deep learning, a sub-field of machine learning, has become so successful in many research areas such as computer vision and natural language processing [ZYS17]. The amazing performance on image recognition and language translation and its attractive property of learning feature representations from scratch make it a glaring star garnered considerable interests. In this chapter, in order to provide more complex and intelligent solutions, I propose two future potential research directions for automatic service management: deep bandit model and script generation machine.

6.1 Deep Bandit Collaborative Filtering Model

Deep learning techniques have made tremendous success in many application domains including information retrieval and recommender systems in the past few decades. In Chapter 4 and Chapter 5, our proposed bandit models have successfully solved the challenges existed in the current IT automation services system. However, both context-based and context-free automation recommendation model is based on the

assumption that there exists a linear regression mapping function between problem symptom and its corresponding automation (i.e. scripted resolution). Deep neural networks are capable of modeling the non-linear mapping function in data using nonlinear activations such as relu, sigmoid, tanh and etc [ZYS17], which makes it possible to capture the more complex and intricate user-item interaction patterns in recommender systems. It is a natural way to construct a dual neural network introducing the nonlinear transformation to model the interaction between users and items. In our system, a ticket problem can be treated as a user and an automation as an item.

In [HLZ+17, ZYS17], neural collaborative filtering (NCF) model is introduced. Figure 6.1 shows the architecture of NCF model. Let $\mathbf{p}_m$ and $\mathbf{q}_n$ denote the contextual information of user m and item n. The predictive reward $\hat{r}_{m,n}$ function can be defined in the following:

$$\hat{r}_{m,n} = f(\mathbf{P}^T \cdot \mathbf{p}_m, \mathbf{Q}^T \cdot \mathbf{q}_n | \mathbf{P}, \mathbf{Q}, \theta)$$

where $f(\cdot)$ represents the nonlinear activations and θ indicates the all parameters of this network. It is convenient to come up with a more general model that leverages both linearity of matrix factorization and non-linearity of deep neural network to improve recommendation performance. However, NCF is not in an online mode.

Deep reinforcement learning [MKS+15] has made a significant improvement on the applications such as AlphaGo and Atari games, which makes the agent can well adopt the surrounding environment through learning the strategies from the immediate feedback from outside after taking action. Deep bandit model [RTS18] is developed to balance exploration and exploitation in complex domains. Facing the challenges of the automation recommendation, it is promising to come up with deep bandit collaborative filtering model that could utilize both the advantages of two models.

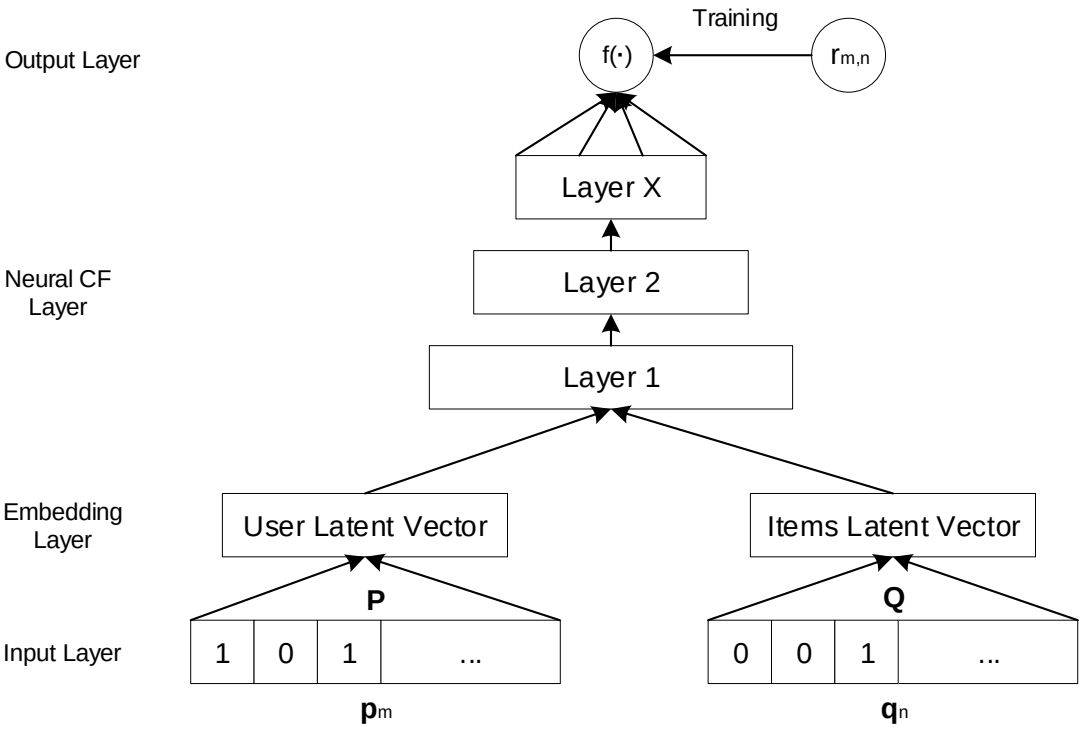

Figure 6.1: Neural Collaborative Filtering.

6.2 Script Generation Machine

As we mentioned in Chapter 4, incident tickets that manually created by customers and escalated from IT automation services will be directly forward to human engineers, which leads to a labor-intensive and error-prone process for problem determination, diagnosis, and resolution. Therefore, a more intelligent idea is proposed. Is it possible to automatically generate the proper scripted resolutions and recommend them to human engineers for further improvement when a new ticket problem is arriving in the system?

Indeed, neural machine translation [BCB14] provides a possible way. We can consider the scripted resolutions (i.e., automations) are written in a machine language. In the IT automation services system, thousands of pairs of ticket resolutions written in the natural human language and the form of scripts can be used to train a neural

machine translation model, which is a recent popular end-to-end learning approach for automated translation [WSC+16]. In Figure 6.2, a simple example is presented to demonstrate how to translate the natural human language into Linux commands using a seq2seq model. Some related work [LWP+17, ZXS17] have shown the feasibility of our proposal.

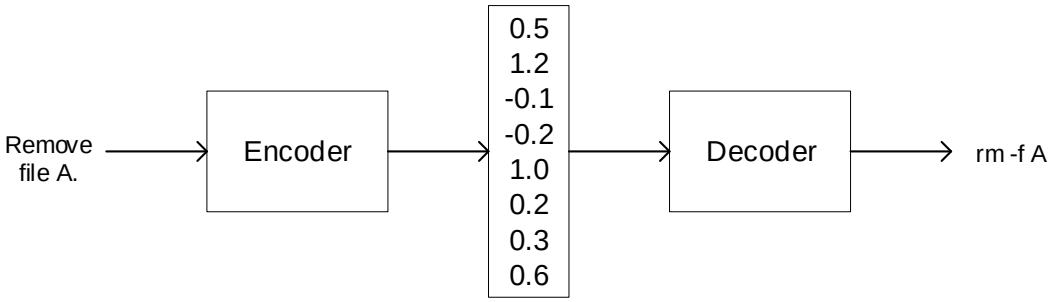

Figure 6.2: Encoder-decoder architecture for neural machine translation. An encoder learns the vector representation from an source sentence. A decoder is used to produce the translation.

6.3 Summary

In this chapter, I highlighted potential research directions by extending to deep neural network research area. Its capability and property will greatly enhance the performance of automatic processes and finally achieve the ultimate goal of fully maximizing the automation of subroutine procedures such as problem detection, determination, and resolution without any human interaction.

BIBLIOGRAPHY

[ABCM09] Michal Aharon, Gilad Barash, Ira Cohen, and Eli Mordechai. One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs. In *Machine Learning and Knowledge Discovery in Databases*, pages 227–243. Springer, 2009.

[ABD+07] Naga Ayachitula, Melissa Buco, Yixin Diao, Surendra Maheswaran, Raju Pavuluri, Larisa Shwartz, and Chris Ward. It service management automation-a hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 574–581. IEEE, 2007.

[AC75] Alfred V Aho and Margaret J Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[ACBFS02] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[ACE09] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Explore/exploit schemes for web content optimization. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 1–10. IEEE, 2009.

[AG13] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

[Ahn08] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[AK08] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

[Aue02] Peter Auer. Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[BBG12] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Gançarski. A contextual-bandit algorithm for mobile context-aware recommender system. In *International Conference on Neural Information Processing*, pages 324–331. Springer, 2012.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[BL+07] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[BP17] Krisztian Buza and Ladislav Peska. Aladin: A new approach for drug–target interaction prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 322–337. Springer, 2017.

[BSW+08] C. Bartsch, L. Shwartz, C. Ward, G.Ya. Grabarnik, and M.J. Buco. Decomposition of it service processes and alternative service identification using ontologies. In *NOMS*, pages 714–717. IEEE, 2008.

[BU17] Andrea Barraza-Urbina. The exploration-exploitation trade-off in interactive recommender systems. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 431–435. ACM, 2017.

[CJLP10] Carlos Carvalho, Michael S Johannes, Hedibert F Lopes, and Nick Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

[CL11] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[CZC+15] Shiyu Chang, Jiayu Zhou, Pirooz Chubak, Junling Hu, and Thomas S Huang. A space alignment method for cold-start tv show recommendations. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3373–3379. AAAI Press, 2015.

[DDFG01] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[DGA00] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[DHR08] Mohamed Yehia Dahab, Hesham A Hassan, and Ahmed Rafea. Textontoex: Automatic ontology construction from natural english text. *Expert Systems with Applications*, 34(2):1474–1480, 2008.

[DJL09] Yixin Diao, Hani Jamjoom, and David Loewenstern. Rule-based problem classification in it service management. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 221–228. IEEE, 2009.

[DKZ+03] Petar M Djurić, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *Signal Processing Magazine, IEEE*, 20(5):19–38, 2003.

[DLT+13] Omkar Deshpande, Digvijay S Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *Proceedings SIGMOD*, pages 1209–1220. ACM, 2013.

[GLK+17] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. On context-dependent clustering of bandits. In *International Conference on Machine Learning*, pages 1253–1262, 2017.

[GLZ14] Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 757–765, 2014.

[GSS93] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113. IET, 1993.

[Hal62] John H Halton. Sequential monte carlo. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 58, pages 57–78. Cambridge Univ Press, 1962.

[HKBR99] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[HNL+17] Daniel N Hill, Houssam Nassif, Yi Liu, Anand Iyer, and SVN Vishwanathan. An efficient bandit algorithm for realtime multivariate optimization. In *SIGKDD*, pages 1813–1821. ACM, 2017.

[IGFH10] Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. Ontology-based news recommendation. In *EDBT/ICDT*, page 16. ACM, 2010.

[JK95] John S Justeson and Slava M Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, 1(01):9–27, 1995.

[JS04] Rong Jin and Luo Si. A bayesian approach toward active learning for collaborative filtering. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 278–285. AUAI Press, 2004.

[KBK+15] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in Neural Information Processing Systems*, pages 1297–1305, 2015.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[KDF+] Kristof Kloeckner, John Davis, Nicholas C Fuller, Giovanni Lanfranchi, Stefan Pappe, Amit Paradkar, Larisa Shwartz, Maheswaran Surendra,

and Dorothea Wiesmann. Transforming the it services lifecycle with ai technologies.

[KMSS03] Leonid Kalinichenko, Michele Missikoff, Federica Schiappelli, and Nikolay Skvortsov. Ontological modeling. In *Proc. of the 5th Russian Conference on Digital Libraries RCDL2003, St.-Petersburg, Russia*, pages 7–13, 2003.

[KWI+11] Cristina Kadar, Dorothea Wiesmann, Jose Iria, Dirk Husemann, and Mario Lucic. Automatic classification of change requests for improved it service quality. In *SRII Global Conference (SRII), 2011 Annual*, pages 430–439. IEEE, 2011.

[LCL+12] Lihong Li, Wei Chu, John Langford, Taesup Moon, and Xuanhui Wang. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. *JMLR*, 26:19–36, 2012.

[LCLS10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[LKKW07] Chang-Shing Lee, Yuan-Fang Kao, Yau-Hwang Kuo, and Mei-Hui Wang. Automated ontology construction for unstructured text documents. *Data & Knowledge Engineering*, 60(3):547–566, 2007.

[LPG02] G di Lucca, M Di Penta, and Sara Gradara. An approach to classify software maintenance requests. In *Software Maintenance, 2002. Proceedings. International Conference on*, pages 93–102. IEEE, 2002.

[LWP+17] Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, and Michael D Ernst. Program synthesis from natural language using recurrent neural networks. *University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01*, 2017.

[LZ07] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2007.

[LZJ+17] Tao Li, Chunqiu Zeng, Yexi Jiang, Wubai Zhou, Liang Tang, Zheng Liu, and Yue Huang. Data-driven techniques in computing system management. *ACM Computing Surveys (CSUR)*, 50(3):45, 2017.

[LZZ+16] Tao Li, Wubai Zhou, Chunqiu Zeng, Qing Wang, Qifeng Zhou, Dingding Wang, Jia Xu, Yue Huang, Wentao Wang, Minjing Zhang, et al. Didap: An efficient disaster information delivery and analysis platform in disaster management. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1593–1602. ACM, 2016.

[LZZ+17] Tao Li, Chunqiu Zeng, Wubai Zhou, Wei Xue, Yue Huang, Zheng Liu, Qifeng Zhou, Bin Xia, Qing Wang, Wentao Wang, et al. Fiu-miner (a fast, integrated, and user-friendly system for data mining) and its applications. *Knowledge and Information Systems*, 52(2):411–443, 2017.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[MN+95] Kavi Mahesh, Sergei Nirenburg, et al. A situated ontology for practical nlp. In *IJCAI*, volume 19, page 21. Citeseer, 1995.

[MRTM12] Dhruv Kumar Mahajan, Rajeev Rastogi, Charu Tiwari, and Adway Mitra. Logucb: an explore-exploit algorithm for comments recommendation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 6–15. ACM, 2012.

[MS08] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264. ACM, 2008.

[MSGL09] Patricia Marcu, Larisa Shwartz, Genady Grabarnik, and David Loewenstern. Managing faults in the service delivery process of service provider coalitions. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 65–72. IEEE, 2009.

[NM+01a] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[NM+01b] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[PACJ07] Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for taxonomies: A model-based approach. In *Pro-*

ceedings of the 2007 SIAM International Conference on Data Mining, pages 216–227. SIAM, 2007.

[PBK17] Ladislav Peska, Krisztian Buza, and Júlia Koller. Drug-target interaction prediction: A bayesian ranking approach. *Computer methods and programs in biomedicine*, 152:15–21, 2017.

[PCA07] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, pages 721–728. ACM, 2007.

[PO07] Sandeep Pandey and Christopher Olston. Handling advertisements of unknown quality in search advertising. In *Advances in neural information processing systems*, pages 1065–1072, 2007.

[RAC+02] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.

[RESK15] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015.

[RKR08] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100, 2008.

[RR18] Alex Ratner and Christopher Ré. Knowledge base construction in the machine-learning era. *Queue*, 16(3):50, 2018.

[RTS18] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown. In *International Conference on Learning Representations*, 2018.

[S+00] John F Sowa et al. *Knowledge representation: logical, philosophical, and computational foundations*, volume 13. Brooks/Cole Pacific Grove, CA, 2000.

[SDdFG13] Adrian Smith, Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[SFHS07] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[SM86] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[SPUP02] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[STvdS16] Linqi Song, Cem Tekin, and Mihaela van der Schaar. Online learning in large-scale contextual recommender systems. *IEEE Transactions on Services Computing*, 9(3):433–445, 2016.

[TJL+15] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 323–332. ACM, 2015.

[TJLL14] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 73–80. ACM, 2014.

[TLS12] Liang Tang, Tao Li, and Larisa Shwartz. Discovering lag intervals for temporal dependencies. In *Proceedings of the 18th ACM SIGKDD in-*

ternational conference on Knowledge discovery and data mining, pages 633–641. ACM, 2012.

[TLS+13] Liang Tang, Tao Li, Larisa Shwartz, Florian Pinel, and Genady Ya Grabarnik. An integrated framework for optimizing automatic monitoring systems in large it infrastructures. In *SIGKDD*, pages 1249–1257. ACM, 2013.

[TLSG13] Liang Tang, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Recommending resolutions for problems identified by monitoring. In *IFIP/IEEE IM*, pages 134–142. IEEE, 2013.

[TM00] Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *SIGDAT*, pages 63–70. ACL, 2000.

[Tok10] Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210. Springer, 2010.

[urla] AI Impacts IT Service management. `https://www.servicedeskshow.com/feature/how-ai-will-impact-it-service-management`.

[urlb] HP OpenView : Network and Systems Management Products. `http://www8.hp.com/us/en/software/enterprise-software.html`.

[urlc] IBM Cognitive Computing. `http://research.ibm.com/artificial-intelligence/`.

[urld] IBM Enterprise IT Automation Services. `http://www.redbooks.ibm.com/redpapers/pdfs/redp5363.pdf`.

[urle] IBM Enterprise IT Automation Services. `www.redbooks.ibm.com/redpapers/pdfs/redp5363.pdf`.

[urlf] IBM Tivoli : Integrated Service Management. `http://ibm.com/software/tivoli/`.

[urlg] ITIL. `http://www.itlibrary.org/`.

[Wel84] Terry A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[WHLE17] Xin Wang, Steven CH Hoi, Chenghao Liu, and Martin Ester. Interactive social recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 357–366. ACM, 2017.

[WHS17] Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. Large-scale collaborative ranking in near-linear time. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 515–524. ACM, 2017.

[WLI+18] Qing Wang, Tao Li, SS Iyengar, Larisa Shwartz, and Genady Ya Grabarnik. Online it ticket automation recommendation using hierarchical multi-armed bandit algorithms. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 657–665. SIAM, 2018.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[WWGW16] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538. ACM, 2016.

[WWW16] Huazheng Wang, Qingyun Wu, and Hongning Wang. Learning hidden features for contextual bandits. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1633–1642. ACM, 2016.

[WWW17] Huazheng Wang, Qingyun Wu, and Hongning Wang. Factorization bandits for interactive recommendation. In *AAAI*, pages 2695–2702, 2017.

[WZZ+17a] Qing Wang, Chunqiu Zeng, Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *arXiv preprint arXiv:1708.03058*, 2017.

[WZZ+17b] Qing Wang, Wubai Zhou, Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Constructing the knowledge base for cognitive it service management. In *Services Computing (SCC), 2017 IEEE International Conference on*, pages 410–417. IEEE, 2017.

[XHF+09] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the 22nd ACM SIGOPS*, pages 117–132. ACM, 2009.

[YHG12] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. Hierarchical exploration for accelerating contextual bandits. *arXiv preprint arXiv:1206.6454*, 2012.

[ZB16] Li Zhou and Emma Brunskill. Latent contextual bandits and their application to personalized recommendations for new users. *arXiv preprint arXiv:1604.06743*, 2016.

[ZL15] Chunqiu Zeng and Tao Li. Event pattern mining. *Event Mining: Algorithms and Applications*, pages 71–121, 2015.

[ZLSG14] Chunqiu Zeng, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Hierarchical multi-label classification over ticket data using contextual loss. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE, 2014.

[ZLSG15a] Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Recommending ticket resolution using feature adaptation. In *CNSM*, pages 15–21. IEEE, 2015.

[ZLSG15b] Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Recommending ticket resolution using feature adaptation. In *CNSM*, pages 15–21. IEEE, 2015.

[ZTL+14] Chunqiu Zeng, Liang Tang, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Mining temporal lag from fluctuating events for correlation and root cause analysis. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 19–27. IEEE, 2014.

[ZTL+15] Wubai Zhou, Liang Tang, Tao Li, Larisa Shwartz, and Genady Ya Grabarnik. Resolution recommendation for event tickets in service management. In *IFIP/IEEE IM*, pages 287–295. IEEE, 2015.

[ZTZ+17] Chunqiu Zeng, Liang Tang, Wubai Zhou, Tao Li, Larisa Shwartz, Genady Grabarnik, et al. An integrated framework for mining temporal logs from fluctuating events. *IEEE Transactions on Services Computing*, 2017.

[ZWML16] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2025–2034. ACM, 2016.

[ZWW+16] Chunqiu Zeng, Qing Wang, Wentao Wang, Tao Li, and Larisa Shwartz. Online inference for time-varying temporal dependency discovery from time series. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1281–1290. IEEE, 2016.

[ZXB+17] Wubai Zhou, Wei Xue, Ramesh Baral, Qing Wang, Chunqiu Zeng, Tao Li, Jian Xu, Zheng Liu, Larisa Shwartz, and Genady Ya Grabarnik. Star: A system for ticket analysis and resolution. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2181–2190. ACM, 2017.

[ZXS17] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[ZYS17] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[ZYZ11] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324. ACM, 2011.

[ZZL+17] Chunqiu Zeng, Wubai Zhou, Tao Li, Larisa Shwartz, and Genady Y Grabarnik. Knowledge guided hierarchical multi-label classification over ticket data. *IEEE Transactions on Network and Service Management*, 2017.

[ZZW13] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on*

Conference on information and knowledge management, pages 1411–1420. ACM, 2013.

VITA

QING WANG

2014-Present	Ph.D., Computer Science Florida International University, Miami, Florida
2013	M.S., Computer Science Xidian University, Xi'an, P.R. China
2009	B.A., Computer Science Zhengzhou University, Zhengzhou, P.R. China

PUBLICATIONS

Hongjun Li, Biao Cai, Shaojie Qiao, Qing Wang, Yan Wang, *ExTCKNN: Expanding Tree-based Continuous K Nearest Neighbor Query in Road Networks with Traffic Rules*, In IEEE Access, 2018.

Qing Wang, Chunqiu Zeng, S. S. Iyengar, Tao Li, Larisa Shwartz, Genady Y. Grabarnik, *AISTAR: An Intelligent Integrated System for Online IT Ticket Automation Recommendation*, In *Proceedings of the 6th annual IEEE International Conference on Big Data (IEEE Big Data)*, 2018.

Qing Wang, Chunqiu Zeng, Wubai Zhou, Tao Li, S. S. Iyengar, Larisa Shwartz, Genady Y. Grabarnik, *Online Interactive Collaborative Filtering Using Multi-armed Bandit with Dependent Arms*, In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2018.

Qing Wang, S. S. Iyengar, Tao Li, Larisa Shwartz, Genady Ya. Graharnik, *Online IT automation recommendation Using Hierarchical Multi-armed Bandit Algorithms*, *SIAM International Conference on Data Mining (SDM)*, 2018.

Wubai Zhou, Wei Xue, Ramesh Baral, Qing Wang, Chunqiu Zeng, Tao Li, Jian Xu, Zheng Liu, Larisa Shwartz, Genady Ya.Grabarnik, *STAR: A System for Ticket Analysis and Resolution*, In *Proceedings of the 23nd annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2017.

Wei Xue, Wubai Zhou, Tao Li, Qing Wang, *MTNA: A Neural Multi-Task Model for Aspect Category Classification and Aspect Term Extraction on Restaurant Reviews*, In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)*, 2017.

Qing Wang, Wubai Zhou, Chunqiu Zeng, Tao Li, Larisa Shwartz, Genady Ya.Grabarnik, *Constructing the Knowledge Base for Cognitive IT Service Management*, In *Proceedings of the 14th IEEE International Conference on Services Computing (IEEE SCC)*, 2017.

Chunqiu Zeng, Qing Wang, Wentao Wang, Tao Li, Larisa Shwartz, *Online Inference for Time-Varying Temporal Dependency Discovery from Time Series*, in *Proceedings of the 4th annual IEEE International Conference on Big Data (IEEE Big Data)*, 2016.

Tao Li, Wubai Zhou, Chunqiu Zeng, Qing Wang, Qifeng Zhou, Dingding Wang, Jia Xu, Yue Huang, Wentao Wang, Minjing Zhang, Steve Luis, Shu-Ching Chen, Naphtali Rishe, *DI-DAP: An Efficient Disaster Information Delivery and Analysis Platform in Disaster Management*, in *Proceedings of the 25th ACM Conference on Information and Knowledge Management (CIKM)*, 2016.

Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, Tao Li, *Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit*, in *Proceedings of the 22nd annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2016.

Tao Li, Chunqiu Zeng, Wubai Zhou, Wei Xue, Yue Huang, Zheng Liu, Qifeng Zhou, Bin Xia, Qing Wang, Wentao Wang, Xiaolong Zhu, *FIU-Miner (A Fast, Integrated, and User-Friendly System for Data Mining) and Its Applications*, In *Knowledge and Information Systems (KAIS)*, 2016.