COMPUTATIONAL USER INTENT MODELING

BY

HONGNING WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor ChengXiang Zhai, Chair
Professor Jiawei Han
Professor Dan Roth
Dr. Evgeniy Gabrilovich, Google

Abstract

User modeling is essential for any information service system (e.g., search engines, recommender systems, and computational advertising) to optimize its service to the end users. The level of user understanding directly determines the upper bound of optimality that such a system can achieve when assisting its users. Unfortunately, due to the limited support in current human-computer interaction interfaces, users are restricted to express their complex information needs via simple keyword queries or some predefined categories, which are too shallow to capture users' higher-level latent intents that influence their decisions and preferences. As a result, there is great demand to build effective computational models to analyze users' generated data and their behavior patterns when they interact with such systems, and understand users' underlying intents so as to enable the systems to provide optimal and personalized services for each individual user.

This dissertation aims at developing general and effective computational methods for user modeling based on two specific types of user-generated data. First, a novel opinionated text mining problem called Latent Aspect Rating Analysis (LARA) is proposed and studied. Clearly distinct from all previous works in opinion analysis that mostly focus on integrated entity-level opinions, LARA for the first time reveals individual users' latent sentiment preference at the level of topical aspects in an unsupervised manner. A prototype system called ReviewMiner has been developed based on the techniques proposed in the LARA work. Second, users' interaction patterns recorded in search engine logs (e.g., their issued queries and clicked documents) are explored for understanding their longitudinal information seeking behaviors. Various important problems related to users' search behaviors have been addressed, including long-term search task identification, personalized ranking model adaptation prediction and task-level search satisfaction.

To my wife and my parents for all their love.

Acknowledgments

I would like to thank all the people and agencies who give me tremendous support and help to make this thesis happen.

First and foremost, I would like to express my sincere thanks and appreciation to my advisor Prof. ChengXiang Zhai, for his generous time and devotion on supervision and guidance in the past five years. In many ways, Prof. Zhai has set up a great example of authentic researcher for me: his keen insights and vision, his passion on research and life, his patience in mentoring students, and his diligence. I always feel lucky to have Prof. Zhai as my Ph.D. advisor, for all the insightful discussions, bright ideas, earnest encouragements, and strongest supports.

I would like to thank my thesis committee members, Prof. Jiawei Han, Prof. Dan Roth, and Dr. Evgeniy Gabrilovich, for their great feedback and suggestions on my Ph.D. research and thesis work. Prof. Han has been giving me invaluable support in many aspects of my research, from fellowship application to job applications. I have also benefited a lot from the discussions with him and his Data Mining course. Prof. Roth has given me many insightful comments about machine learning research and invaluable support and advice in my job applications. Dr. Gabrilovich has provided me helpful guidance and suggestions in research since I was an intern student in Yahoo! Labs in 2011.

I also sincerely thank my mentor Yang Song, Xiaodong He, and colleagues Ming-Wei Chang, Ryen White, Hao Ma, Kuansan Wang in Microsoft Research, and my mentor Anlei Dong and Yi Chang in Yahoo! Labs for their long-term support and collaborations. The collaborations have given me the unique opportunities to be exposed to practical industrial problems, and the outcome of these fruitful collaborations has made the very important pieces of this thesis.

Many thanks to my other collaborators, including Yue Lu, Chi Wang, Duo Zhang, Yanen Li, Mianwei Zhou and Hongbo Deng from our DAIS group, Lihong Li, Wei Chu and Ahmed Hassan

from Microsoft Research, Prof. Feng Liang and Prof. Yuguo Chen from the Department of Statistics at University of Illinois.

I also want to thank all TIMAN group members and visitors, who have built such a great environment for research. Especially many thanks to the senior group members, Hui Fang, Qiaozhu Mei, Xuanhui Wang and Yuanhua Lv. I have benefited so much from discussions with them. And I would like to thank Gong Chen and Jinyao Xu for their contribution in developing the ReviewMiner system.

Finally and above all, I owe my deepest gratitude to my wife Hui and my parents. I want to thank my wife, for her love, understanding, patience, and support at every moment. And I want to thank my parents for their endless and unreserved love, who have been encouraging and supporting me all the time. This thesis is dedicated to them.

This thesis was supported in part by Google Ph.D. fellowship.

Table of Contents

Chapter 1

Introduction

"Human behavior flows from three main sources: desire, emotion, and knowledge."

-Plato

Modern information service systems, such as search engines and recommender systems, give ordinary people the ability to access vast volume of information easily. In almost every scenario of decision making, e.g., online shopping, vacation planning and health condition self-diagnose, people collect, filter, and synthesize information from multiple sources via the help of different types of information service systems. It is evident that such systems have become an indispensable component in most people's daily life when accessing information.

Various types of research effort have been devoted onto building a more useful and effective information service system, including document analysis [37] and indexing [51], automatic query recommendation [10] and reformulation [63], supervised ranking model estimation [84], and search result diversification [5]. Nevertheless, an essential component in such systems has not received enough attention in previous studies: in-depth understanding and modeling of system users. Because the end users are the final customer of the system and the ultimate judge of utilities of such systems, understanding their information need within the system provide invaluable insight for system design and optimization.

In this thesis, we focus on developing computational methods for user modeling via broad exploration of the user-generated data. Both the users' generated opinionated review text data, i.e., their explicit opinions towards a place or a service, and their interaction behaviors recorded in the system, e.g., clicking on a document or reformulating the previous query, are analyzed for understanding their underlying intentions. By building formal mathematic models for the system users, a positive feedback loop can be formed between the end users and systems: users make

informed decisions based on the synthesized information from the system, and the system refines its service strategy according to users' feedback. Both users and system collaboratively optimize their own objectives within such a feedback loop.

In this chapter, we first present the motivation and overview of this thesis in Section 1.1, and then introduce the definition of the general problem studied in this thesis in Section 1.2, and in the end discuss the organization of this thesis in Section 1.3.

1.1 Motivation and Overview

Accurate user modeling is essential for any information service system (e.g., search engines, recommender systems, and computational advertising) to optimize its service to the end users. An ideal user model should be capable to accurately and comprehensively capture users' distinct information need in various application scenarios so as to assist the systems to deliver the most relevant information to the users. However, the information needs and decision making process of different users vary significantly, and they are largely hidden from system designers. As a result, a generic user model can hardly be optimal, and the level of user understanding directly determines the upper bound of optimality that such information service systems can achieve in assisting their customers.

Consider the following example. When a vacation planner first enters the query "hotels around Orlando Disneyland" into a search engine and focuses on the reservation pages of hotels with low price and far away from Disneyland theme parks, the system should keep in mind not only that the user is performing a search task of preparing for an upcoming vacation to Orlando, but also she cares more about the price aspect than the location aspect. Meanwhile, if another user issues the same query but browses mostly the expensive hotels in a close neighborhood to Disneyland theme parks, the system can then infer that this user concerns more about location aspect than price. Such a detailed understanding of users at the level of decision factors (aspects) and different users' preferences over multiple aspects would enable the system to effectively personalize and optimize the service for each individual user accordingly.

Moreover, the system should also "remember" the inferred user preferences from now on so as to further optimize the service for this user in the future. For example, one week later, the first user comes back and issues another query "flights from Chicago to Orlando." The system

should immediately recall the context of her ongoing task of vacation planning and recognize that she is now at the stage of booking flight tickets for this coming vacation. More importantly, the system should also realize that in the last stage of the task (hotel booking) she preferred a lower price. With such knowledge about this user, the system could then actively retrieve the cheap flights on the top while tolerating the flights with more stops or inconvenient time. Besides, as the next step of vacation planning task, the system should also recommend vacation package deals in Orlando area, shopping center and popular restaurant locations to her, according to the search history and preferences of other similar vacation planners, where similarity between users would be computed based on the inferred user task flow and their preferences on various decision factors. Users' productivity can be greatly improved with such personalized service. In addition, a deeper understanding of user intent will also enable more detailed assessment of user similarities and analysis of user behaviors in a broader context, which has many practical applications in business intelligence, market research, policy impact analysis and etc.

Meanwhile, the system should also be aware of users' satisfaction about the returned results. If the user is satisfied with the results (i.e., positive feedback), it indicates the currently inferred user intent aligns with the user's true information need, and therefore the user model can be retained and enhanced in future; and if the user is not satisfied (i.e., negative feedback), it indicates the user model needs to be updated to better capture the user's underlying intention. We should note that it is difficult to directly acquire users' explicit feedback in most of the practical information service systems. Therefore, modeling users' implicit feedback to understand their satisfaction is essential.

The examples above illustrate the picture of *computational user intent modeling* studied in this dissertation. In general, an intelligent information service system needs to: 1). ***figure out the latent decision factors*** in a user's different information seeking processes, e.g., price and location for hotel booking; 2). ***optimize system output according to the identified user's preferences*** during the user's whole decision making process; and 3). ***recognize user's satisfaction with regarding to the personalized output***, and adaptively update the system's service strategy. Such in-depth analysis will help the information service systems to accurately recognize various users' information need, capture the association among their non-consecutive information seeking behaviors, and optimize the quality of delivered information.

Nevertheless, in most of the existing information service systems, e.g., search engine (such as Google.com and Bing.com) and online shopping websites (such as Amazon.com and Newegg.com), users are limited to express their complex information needs via simple keyword queries or some predefined categories due to the restrictive functionalities in current user interaction interfaces. It results in limited signals for analyzing and capturing the users' underlying complex information need. Previous work of user modeling is largely restricted to the study of constructing keyword- [106] or semantic-category- [82] based user profiles from users' input history. However, such shallow user representation is insufficient to capture the higher-level latent factors influencing users' decisions and their preferences over such factors, nor to distinguish the variation and evolution of their preferences over time. Both aspects are critical for a deep understanding of user intentions and the optimality of services provided to the end users. Therefore, it is crucial for us to go beyond the simple user inputs that the systems receive, broadly explore and properly model various types of user-generated data, e.g., the opinionated review text content they have written on the web, news reports they have shared via online social networks, and their search and browsing history in a search engine, in order to acquire a more comprehensive and accurate understanding of users' search intents.

To break the aforementioned limitations, in this thesis, a more comprehensive and deeper computational user modeling principle and specific methods are proposed for capturing the users' latent decision factors, detailed users' preferences over those factors, and associations among their long-term information seeking behaviors, so as to optimize the systems' output according to such identified user intent. Admittedly, deriving such deep understanding of user intent is not a straightforward practice of applying existing data mining or machine learning techniques, since it has unique properties and challenges. First, because a user's decision process is largely hidden from the information service systems, we can only collect scattered pieces of information, e.g., users' queries and clicks, reflecting their underlying requirements, which are known to be sparse and noisy. Second, users' intents are distinct and highly dynamic, different users, or even the same user under different contexts, would hold varied intentions. Third, the users' information seeking behaviors, e.g., searching and browsing, are neither isolated nor independent; instead, they are serving for the same intrinsic short-term or long-term intent, but the interaction mechanism is largely unknown

to the information service systems. To solve these challenges, effective computational models are required to analyze and understand user intents by taking a comprehensive view of their generated data - exploring all information available about each individual user, e.g., the opinionated review comments he or she has written, the queries he or she has issued, the corresponding search result pages he or she has clicked, and also the behaviors from other users, who share similar intentions as the target user.

1.2 Problem Formalization

In this section, we will formally discuss the research of *computational user intent modeling* studied in this thesis, illustrate the formal definition of the input, output and computational problems in it.

The input of the *computational user intent modeling* problem is a set of ***observable*** user behaviors from a particular user u when interacting with a particular type of information service system, e.g., search engine, or a review portal,

Definition (User Behavior) A particular observation of user behavior x_t is a N-dimensional vector, which characterizes user u's action at time t when interacting with the system.

Real examples of user behaviors include: in opinion analysis, writing an opinionated review document is considered as a specific type of user behavior, where x_t can be modeled as a feature vector describing the text content of review document (e.g., a N-dimensional bag-of-word vector); in search engine log analysis, issuing a keyword query is considered as one type of user behavior, where x_t can be modeled as a feature vector describing the issued query, e.g, length of query, timestamp, and number of returned results.

In many application scenarios, a user might take a series of actions to interact with the system, e.g., in retrieval problems, a user might issue an initial query, click on a returned document and then reformulate the query to another in order to fulfill her information need; in opinion mining problems, a user might write several reviews respect to her previous visit to multiple locations. Therefore, we define the input of our problem as a set of user behaviors from user u, i.e., $X^u = \{x_0^u, x_1^u, \ldots, x_t^u\}$, in which the actions are ordered by the time when it was taken.

The output is the prediction of associated labels with respect to each input user behavior. For example, in opinion analysis, the overall rating of a review document from user u is the behavior label for prediction; and in search engine log analysis, satisfaction label of a search task from user u is the behavior label for prediction. Formally, such output can be defined as,

Definition (Behavior Label) A label y_t^u for user behavior x_t^u is an observable numerical variable, discrete or continuous, which is semantically associated with x_t^u or describing x_t^u with respect to a specific application.

We should note that such behavior labels may not be directly available from the user-generated data in certain application scenarios, and therefore third-party annotation is necessary to acquire such labels. For example, in opinion analysis, the overall rating is generally available in most of opinionated review documents; while in search engine log analysis, task labels are not available for the queries in the search log data (i.e., which queries belong to the same search task), and third-party annotators are needed to create such labels. However, in both cases, we require such behavior labels are given in advance for model training purpose.

However, the major focus of the research studied in this thesis is not to model the direct mapping from observed user behaviors to the corresponding labels. What is more important in this thesis research is to properly model the ***latent*** user intents, which are *unobservable* but directly *associated* with user behaviors X^u and the corresponding behavior label y^u for prediction. Take the problem of user search task satisfaction prediction studied in this thesis as an example. In such a problem, a user's detailed search actions, e.g., issuing a query or clicking on a returned document, result in corresponding action-level satisfaction, which in turn directly determine her overall search task satisfaction. But those intermediate action-level labels are not observable and difficult to be manually annotated in the search log. Therefore, to precisely predict task-level user search satisfaction, we need to explicitly model the latent action-level satisfaction labels in the solution.

Formally, the latent user intent is defined as,

Definition (User Intent) User intent h_t^u under a particular information need of user u is a k-dimensional vector, which is not observable but directly related to both user behavior x_t^u and corresponding behavior label y_t^u at time t.

Accordingly, we denote the latent user intents for a set of user behaviors as $H^u = \{h_0^u, h_1^u, \ldots, h_t^u\}$, where h_t^u corresponds to the latent user intent for action x_t^u at time t.

Based on the above definition, the major focus of research in this thesis is to properly model the latent user intents with respect to the observed user behaviors and corresponding behavior labels defined in specific applications. Due to the explicit modeling of latent user intents, the problem we studied in this thesis is not a simple mapping from the observed user behaviors X^u to the corresponding labels y^u as studied in classic supervised machine learning problems (classification or regression) [12], i.e., $f : X^u \rightarrow y^u$, where f is a proper functional form. Instead, we are estimating a joint mapping from observed user behaviors X^u to both latent user intents H^u and corresponding behavior label y^u, i.e., $f : X^u \rightarrow H^u \times y^u$. Because H^u is unobservable from the input, we need to infer the configuration of h_t^u for each corresponding action x_t^u in such a mapping. As a result, the problem becomes a structured learning problem [25, 113].

Moreover, in addition to prediction results of the behavior label y^u in a given problem, the inferred structure H^u also becomes an output for the problem. And such output actually conveys more informative insight about a user's behaviors and is more important than the y^u label in many application scenarios. For example, in opinion analysis, a reviewer's preferences over the aspects of a particular type of entities is very important to understand her purchase decision, since it unveils a user's decision making process in finer granularity comparing to the overall rating the user has given to this item. As a result, in this thesis, we will also consider the inferred latent user intents as the output of the problem.

Based on the definition of input and output of the *computational user intent modeling* problem, we can formalize the computational problem studied in this thesis in a principled way.

Definition (Computational User Intent Modeling) Given a user u's observed behaviors X^u in a specific application scenairo and corresponding behavior labels y^u, build functional mapping $f(\cdot)$ from X^u to y^u with explicit consideration of latent user intent H^u, i.e., $f : X^u \rightarrow H^u \times y^u$, such that the predicted behavior label is close to the ground-truth.

Specifically, from the perspective of probabilistic modeling, the problem can be formalized into

the following optimization framework,

$$\begin{aligned} \min_{f,\epsilon} \quad & E_{p(H^u,y^u|X^u)}\Big[\mathcal{L}(H^u, y^u|X^u, f)\Big] + C||\epsilon||_p \\ s.t. \quad & E_{p(H^u,y^u|X^u)}\Big[g(X^u, H^u, y^u)\Big] \leq \epsilon \end{aligned} \tag{1.1}$$

where $\mathcal{L}$ is an application-specific loss function characterizing the difference between the predicted behavior label $\hat{y}^u$ and the ground-truth label y^u with respect to the latent structure H^u; the expectation is taken under the conditional probability of $p(H^u, y^u|X^u)$; function $g(\cdot)$ establish possible constraints related to specific application scenarios; and C is a trade-off parameter to relax the violation of constraints.

In some application scenario, it might be infeasible to directly model the conditional distribution of $p(H^u, y^u|X^u)$; instead, we can follow the max-margin principle [115] to formalize the problem in a different format,

$$\begin{aligned} \min_{f,\epsilon} \quad & ||f||_q + C||\epsilon||_p \\ s.t. \quad & \max_{H^u \in \mathcal{H}} f(X^u, H^u, y^u) \geq \max_{(\hat{H}^u, \hat{y}^u) \in \mathcal{H} \times \mathcal{Y}} \Big[f(X^u, \hat{H}^u, \hat{y}^u) + \Delta(\hat{H}^u, \hat{y}^u, X^u)\Big] - \epsilon \\ where \quad & \Delta(\hat{H}^u, \hat{y}^u, X^u) = \mathcal{L}(\hat{H}^u, \hat{y}^u|X^u, f) + g(X^u, \hat{H}^u, \hat{y}^u) \end{aligned} \tag{1.2}$$

where $||\cdot||_p$ and $||\cdot||_q$ could be different norms; $\mathcal{Y}$ and $\mathcal{H}$ are the whole space of possible behavior labels and user intents in the specific problem.

In the probabilistic formalization (i.e., Eq (1.1)), we are trying to minimize the expectation of the prediction error about the known behavior labels; while in the max-margin formalization (i.e., Eq (1.2)), we are trying to maximize the distance between the true configuration of the behavior label and latent intents to any other wrong configurations. These two forms of problem formalization essentially capture the same insight of the user behavior modeling problem, and we will explore both of them in this thesis.

There are two important components introduced in Eq (1.1) and Eq (1.2): the functional form of $f(\cdot)$, which determines the joint mapping from X^u to $H^u \times y^u$; and the constraint function $g(\cdot)$, which specifies the domain knowledge about relationship among X^u, H^u and y^u.

Typical form of $f(\cdot)$ is a linear function, e.g., $f(X^u, H^u, y^u) = w^\mathsf{T}\phi(X^u, H^u, y^u)$, where $\phi(\cdot)$ is a feature vector and w is the corresponding feature weight vector. Accordingly, the mapping is defined as: $(\hat{y}^u, \hat{H}^u) = \arg\max_{(y^u, H^u)\in\mathcal{Y}\times\mathcal{H}} w^\mathsf{T}\phi(X^u, H^u, y^u)$. We should note that the design of $f(\cdot)$ determines the dependency relation among the observed user behavior X^u, latent user intent H^u and behavior label y^u, and therefore it is the key to get reasonable prediction performance of H^u and y^u from the input user behaviors X^u. And it needs to be carefully designed for different application scenarios.

The form of $g(\cdot)$ could be indicator functions or distance functions, which regularize the relation between observed user behaviors X^u, behavior labels y^u and latent user intents H^u. Take the problem of user search task satisfaction prediction as the example again: when all of a user's search actions are unsatisfying, it is impossible that the user will be satisfied by the whole search task in the end.

It is worth discussing the purpose of introducing the constraint function $g(\cdot)$ in general. $g(\cdot)$ encodes our knowledge about the specific problem we are trying to solve. In later detailed discussions of the proposed solutions for each specific problem, we will find that modeling user intent as latent variables will significantly increase the parameter space and complexity of inference (since we have more free parameters to tune), which cause serious overfitting problem. With fixed amount of annotation data, explicitly modeling latent user intent might eventually hurt the prediction performance. Fortunately, by introducing domain knowledge into the modeling framework as constraints, we can alleviate the dependency on human annotation in model estimation, and therefore avoid overfitting due to the lack of observations.

Another consideration of introducing $g(\cdot)$ to encode domain knowledge as constraints instead of treating such knowledge as features in $f(\cdot)$ in the proposed framework is that with $g(\cdot)$ as constraints we can explicitly control the confidence of such knowledge and use it to better estimate the parameters for other features during model training. For example, if we know some properties are strict in the given problem, we can set them as hard constraints and do not allow the model to violate them; while some properties are more tolerable, and we can give the model more freedom via relaxation. And more importantly, in some form of $f(\cdot)$, e.g., linear function, features are independent from each other, such that the knowledge about one feature has little influence on the

other. But if we treat such domain knowledge as constraints, it will effectively affect the estimation of other features during model training.

In the following, we will discuss the organization of this thesis, where concrete examples of instantiation of the proposed general framework will be given.

1.3 Thesis Organization

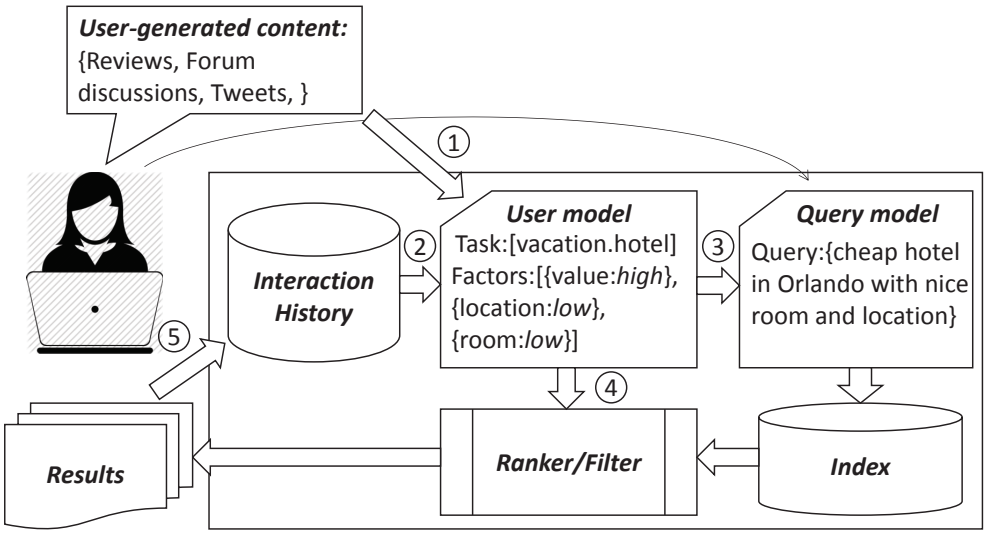


Figure 1.1: Positive feedback loop between users and information service systems enabled by the computational user models. Procedures related to user modeling include: ① - latent user intent identification via exploration of user-generated text data; ② - user search behavior modeling; ③ - user-specific query interpretation; ④ - system output personalization; ⑤ - user satisfaction prediction.

Figure 1.1 illustrates the positive feedback loop between end users and information service systems enabled by the computational user models studied in this thesis. The labels on the control flow indicate the procedures directly related to user modeling in a typical information service system, and this flow graph highlights the organization of this thesis.

According to the type of user-generated data for user intent modeling, this thesis is organized into two parts: 1) modeling opinionated text data for latent user intent identification; and 2) modeling interactive behavior data for system optimization. These two parts of the thesis are not isolated; instead, they are closely connected from three different perspectives. First, from the perspective of positive feedback loop, identifying latent user intents is the basis for constructing

an effective computational user model. Once we have successfully identified the latent factors, which influence a user's decisions over an information service system's output, we can optimize the system to maximize each individual user's satisfaction accordingly. On the other hand, from the perspective of the data being studied, a user's behavior and the underlying intent are not insulated, but the observed behaviors are a means to satisfy her latent intent. It is necessary to associate the scattered user-generated data over time, e.g., their generated text content, issued queries, clicked documents and filled forms, into a semantically coherent unit with respect to their information need. Such unit is helpful for us to refine our understanding of the users' latent intents. The last but not the least, all the computational problems studied in these two parts are based on the same modeling principle discussed in Section 1.2; they are different instantiations of the modeling framework to real problems.

In particular, the overview of the subsequent chapters in this thesis is as follows. In Part I of ***modeling opinionated text data for latent user intent identification***, a new opinionated text mining problem called Latent Aspect Rating Analysis (LARA) is proposed and studied, and a prototype system called ReviewMiner has been developed based on the techniques proposed in the LARA work.

• **Chapter 2: Latent User Intent Identification.** To exploit and analyze user-generated opinionated text content and support a deeper and more detailed understanding of user opinions, A novel text mining problem called Latent Aspect Rating Analysis (LARA) is proposed and studied [118, 119]. The work of LARA aims at analyzing opinions expressed in text document at the level of topical aspects to discover each individual user's latent opinion on each aspect as well as the relative preference the users have placed onto those different aspects when forming the overall judgment. More specifically, in the work of LARA, users' generated opinionated review document is modeled as user behaviors X^u, the corresponding overall rating is treated as the observed behavior label y^u, and the latent aspect ratings and weights are considered as latent intents H^u (i.e., two types of intents). A two-stage approach based on bootstrapping aspect segmentation and latent rating regression model is first proposed to solve the problem of LARA in [118], where it assumes a user's latent intents can be specified by a set of predefined keywords. Later on, a unified framework (rooted in Eq (1.1)) is introduced by incorporating the topic modeling technique to jointly identify

the latent topical aspects, and infer the latent aspect weights/ratings from each user's review article [119]. Clearly distinct from all previous work in opinion analysis that mostly focuses on integrated entity-level opinions, LARA *for the first time* reveals individual users' latent sentiment preferences at the level of topical aspects in an unsupervised manner. Discovering such detailed user preferences (which are often hard to obtain by a human from simply reading many reviews) enables many important applications. First, such analysis facilitates in-depth understanding of user intents. For example, by mining the product reviews, LARA recognizes which aspect influences a particular user's purchase decision the most. Second, by identifying each user's latent aspect preference in a particular domain (e.g., hotel booking), personalized result ranking and recommendation can be achieved. Third, discovering the general population's sentiment preferences over different aspects of a particular product or service provides a more effective way for businesses to manage their customer relationship and conduct market research.

• **Chapter 3: ReviewMiner: A Multi-Modal Opinion Analysis System for Decision Support.** Based on the latent aspect rating analysis technique proposed in Chapter 2, a prototype system called ReviewMiner [1] is developed to support multi-modal opinion analysis and decision making. 17,629 hotels with 1,598,961 reviews from TripAdvisor (www.tripadvisor.com), and 15,356 products with 472,631 reviews from Amazon (www.amazon.com), and 129 medications with 14,725 reviews from WebMD (www.webmd.com) are indexed and analyzed in the system. Besides providing the end users with basic search functions to explore the analyzed entities and reviews in the system, ReviewMiner also personalizes the retrieved results according to the users' input preferences over the aspects of different types of entities, recommends similar entities based the detailed aspect-level opinions, summarizes the aspect-level opinions in textual, temporal and spatial dimensions. This multi-modal opinion summarization and visualization mechanism provides the system users different perspectives to digest information from the review content, and it also provides effective support for the end users to compare across different entities in order to make more informed decisions. In addition, the developed ReviewMiner system actively collects real users' interactive behaviors in an information service system, such that it works as an experimental platform for the algorithms developed in second part of this thesis.

[1] http://timan100.cs.uiuc.edu:8080/ReviewMiner/

In Part II of ***modeling interactive behavior data for system optimization***, users' interaction patterns recorded in search engine logs are analyzed (e.g., their issued queries and clicked documents) for understanding their longitudinal information seeking behaviors. And the internal logic behind this part of work can be understood by the control flow of ②→④→⑤ in Figure 1.1: first, we need to organize the users' longitudinal search behaviors into a semantically coherent unit, which we define as "search task," to reflect their underlying information need; by looking into a particular user's historical search behaviors in the same task, we can effectively personalize the system's output according to her identified preference within the task; in the end, it is necessary for the system to assess the users' satisfaction with regard to its output so as to actively adjust its service strategy in the future. The detailed overview of those chapters is as follows.

• **Chapter 4: Long-Term User Search Tasks Extraction.** To fulfill the goal of understanding the users' longitudinal search behaviors and developing information service systems to support users' long-running tasks, a novel structured learning method is proposed for accurately extracting long-term search tasks from users' historic search activities. In this line of work, a particular user's search queries in a given period of time is modeled as user behaviors X^u, the corresponding search-task assignment is considered as the behavior label y^u, and the internal strcuture of search task assignment is modeled as latent user intents H^u (i.e., query reformulation chain). A semi-supervised clustering model is proposed based on the latent structural SVM framework (rooted in Eq (1.2)), which is capable of learning inter-query dependencies from users' searching behaviors [120]. A set of effective automatic annotation rules were introduced as weak supervision to release the burden of manual annotation (i.e., in the form of constraint function $g(\cdot)$ in Eq (1.2)). Importantly, the proposed method is able to obtain performance gains while reducing the reliance on costly human annotations via such automatically generated weak supervision. Besides partitioning the queries into semantically coherent groups, the proposed method also unveils the in-depth structure among the queries, e.g., the reformulation and evolution chain of queries within the same task over time, which enables detailed analysis of a user's information need. This work paves the way for a wide range of future work in this area: such as automatic task completion helper, which expedites the process of user's ongoing task by automatically recommending and collecting information relevant to the current or even next stage of the task; and task-based user modeling, which recognizes user's

unique in-task behavior patterns and preferences, and adjust system's output for each particular task.

• **Chapter 5: Personalized Ranking Model Adaptation.** Searchers' information needs are diverse and cover a broad range of topics. A single user-independent ranking model is thus insufficient to satisfy different users' result preferences. Inspired by the linear-regression-based model adaptation methods widely studied in automatic speech recognition (e.g., maximum likelihood linear regression [79], minimum classification error linear regression [62]), a general framework of personalized ranking model adaptation is developed [117]. In particular, by assuming that in a parametric ranking model different users' ranking preferences can be fully characterized by different settings of model parameters, personalization can be achieved via adjustment of the generic ranking model's parameters with respect to each individual user's ranking preference, e.g., click feedback. In the proposed framework, such adjustment is achieved via linear transformations. One important merit of the proposed ranking adaptation framework is its generality: in this thesis, we demonstrate the instantiation of the proposed framework with three frequently used learning-to-rank algorithms, i.e., RankNet [19], LambdaRank [20] and RankSVM [69], which achieved significant improvement in, not only adaptation efficiency, but also adaptation accuracy, against several state-of-the-art ranking model adaptation methods in extensive experimentation.

• **Chapter 6: Modeling Action-level Satisfaction for Search Task Satisfaction Prediction.** User satisfaction is a property of a user's information seeking process. Understanding it is critical for information service providers to evaluate the performance and improve the effectiveness of their service strategies. In this work, we studied a specific problem of search satisfaction in the search engine systems. Existing methods model search satisfaction holistically at the search-task level, ignoring important dependencies between action-level satisfaction and overall task satisfaction. We hypothesize that searchers' latent action-level satisfaction label H^u (i.e., whether they believe they were satisfied with the results of a particular search action, e.g., query or click) influences their observed search behaviors X^u and contributes to overall search satisfaction y^u. We conjecture that by modeling search satisfaction at the action level, we can build more complete and accurate predictors of search-task satisfaction. To achieve this, we develop a latent structural learning method (rooted in Eq (1.2)), whereby rich structured features and dependency relations

unique to search satisfaction prediction are explored. Using in-situ search satisfaction judgments provided by searchers, we show that there is significant value in modeling action-level satisfaction in search-task satisfaction prediction.

In Chapter 7, conclusion about the research work conducted in this thesis and the research frontiers related to research of *computational user intent modeling* will be discussed.

Part I

Modeling Opinionated Text Data for Latent User Intent Identification

Chapter 2

Latent Aspect Rating Analysis

In this chapter, we define and study a new opinionated text data analysis problem called Latent Aspect Rating Analysis (LARA), which aims at analyzing opinions expressed about an entity in a review document at the level of topical aspects to discover each individual reviewer's latent opinion on each aspect as well as the relative emphasis on different aspects when forming the overall judgment of the entity.

To solve this new text mining problem in a general way, a two-stage approach based on bootstrapping aspect segmentation and latent rating regression model is first proposed; and later on, a unified model, Latent Aspect Rating Analysis Model (LARAM) is introduced by incorporating the topic modeling technique to jointly identify the aspect segments and infer the latent aspect weights/ratings. The detailed analysis of opinions at the level of topical aspects enabled by the proposed model can support a wide range of application tasks, such as aspect opinion summarization, entity ranking based on detailed aspect ratings, and analysis of reviewers rating behavior, which open up a new direction for user intent modeling.

2.1 Introduction

With the emergence and advancement of Web 2.0, more and more people can freely express opinions on all kinds of entities such as products and services on the Internet. These reviews are useful to other users for making informed decisions and to merchants for improving their service. Nevertheless, with the current techniques [129, 39, 66], it is still hard for users to easily digest and exploit the large number of reviews due to inadequate support for understanding each individual reviewer's opinions at the fine-grained level of topical aspects.

Consider a typical hotel review shown in Figure 2.1. This review discusses multiple aspects of

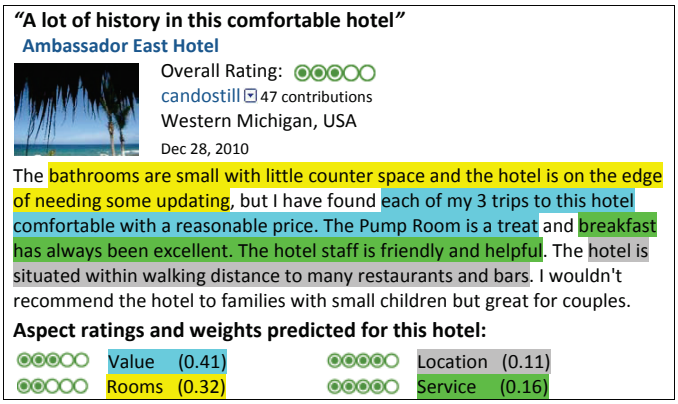


Figure 2.1: Expected output of LARA in hotel review.

the "Ambassador East Hotel," such as price, room condition, and service, but the user only gives an overall rating for the hotel; without an explicit rating on each aspect. Other users would not be able to easily know the reviewer's opinion on each aspect. Going beyond the overall rating to know a reviewer's detailed opinions on different aspects is important because different users may give a hotel the same overall rating for very different reasons. For example, one user may have liked the location, but another may have enjoyed the room. In order to help users tell this difference, it is necessary to understand a reviewer's rating on each of the major decision aspects (i.e., rating factors) of a hotel. Furthermore, even if we can reveal the rating on an aspect such as "price," it may still be insufficient because "cheap" may mean different price ranges for different users. Even the same user may use a different standard to define "cheap" depending on how critical other factors (e.g. location) are; intuitively, when a user cares more about the location, the user would tend to be more willing to tolerate a higher price. To understand such subtle differences, it is necessary to further reveal the relative importance that a user placed on each aspect when assigning the overall rating.

To achieve such a deeper and more detailed understanding of a user from her opinionated review data, I propose to study a novel text mining problem called Latent Aspect Rating Analysis (LARA). Given a set of reviews with overall ratings, LARA aims at *analyzing opinions expressed in each review at the level of topical aspects to discover each individual user's latent rating on each aspect as well as the relative importance weight on different aspects when forming the overall judgment.*

Revealing the latent aspect ratings and aspect weights in each individual review will enable a wide range of important applications. For example, the identified latent ratings on different aspects can immediately support aspect-base opinion summarization; aspect weights are directly useful for analyzing users' rating behaviors; and the combination of latent ratings and aspect weights can support personalized ranking of entities by using only those reviews from the reviewers who share similar aspect weights to those preferred by an individual user.

To solve this new opinion mining problem, a two-stage approach based on bootstrapping aspect segmentation and latent rating regression model was first proposed in [118]. In the first stage, we employed a bootstrapping-based algorithm to identify the major aspects (guided by a few manually drafted seed words describing the aspects) and segment the review content. In the second stage, we proposed a generative Latent Rating Regression (LRR) model which aims at inferring aspect ratings and weights for each individual review based only on the review content and the associated overall rating. More specifically, the basic idea of solving LARA is to assume that the overall rating is "generated" based on a weighted combination of the latent ratings over all the aspects, where the weights are used to model the relative emphasis that the user has placed on each aspect when giving the overall rating. We further assume that latent rating of each aspect depends on the text content in the segment of a review discussing the corresponding aspect through a regression model. In other words, we may also view the latent rating on each aspect as being "generated" by another weighted sum of word features where the weights indicate the corresponding sentimental polarities. Since we do not observe the detailed ratings on different aspects, the response variable of this regression model (i.e., aspect rating) is latent.

Later on a unified model rooted in the general user behavior modeling framework as defined in Eq (1.1), Latent Aspect Rating Analysis Model (LARAM), was introduced by incorporating the topic modeling technique to jointly identify the aspect segments and infer the latent aspect weights/ratings [119]. In LARAM, it is assumed that the text content describing a particular aspect is generated by sampling words from a topic model (i.e., a multinomial word distribution) corresponding to the latent aspect; and based on those automatically identified aspect segments, LRR module is utilized to decompose the overall rating into aspect ratings and unveil the corresponding importance weights of the aspects. As a result, LARAM simultaneously identifies: 1) latent topical

aspects, 2) ratings on each identified aspect, and 3) weights placed on different aspects by a user.

2.2 Related Work

To the best of our knowledge, no previous work has studied the proposed LARA problem, but there are several lines of related work.

Analysis of the overall sentiment of review text data has been extensively studied. Related research started from a definition of binary classification of a given piece of text into the positive or negative class [35, 39, 96, 32, 74]. Later, the definition is generalized to a multi-point rating scale [96, 55]. Many approaches have been proposed to solve the problem, including supervised, unsupervised, and semi-supervised approaches, but they all attempt to predict an *overall* sentiment class or rating of a review, which is not so informative as revealing aspect ratings as we attempt to do in this thesis.

Since an online review usually contains multiple opinions on multiple aspects, some recent work has started to predict the aspect-level ratings instead of one overall rating. For example, Snyder et al. [102] show that modeling the dependencies among aspects using good grief algorithm can improve the prediction of aspect ratings. In [110], Titov et al. propose to extract aspects and predict the corresponding ratings simultaneously: they use topics to describe aspects and incorporate a regression model fed by the ground-truth ratings. However, they have assumed that the aspect ratings are *explicitly* provided in the training data. In contrast, we assume the aspect ratings are *latent*, which is a more general and more realistic scenario. Recent work by Lu et al. [87] is the closest to ours, but their goal is to generate an *aggregated* summary with aspect ratings inferred from overall ratings. Most importantly, none of the previous work considers the reviewer's emphasis on different aspects, i.e., aspect weight. Our work aims at inferring both the aspect ratings and aspect weights at the level of individual reviews; the result can be useful for multiple tasks, including opinion-based entity ranking, analysis of user rating behavior, and rated aspect summarization.

Regarding to the proposed LARAM, which is a hybrid generative model containing both aspect modeling and rating prediction, our work is also related to the work of using topic modeling techniques to extract aspects and associated opinions. Mei et al. incorporated two additional

sentiment language models into standard topic models to extract the facets and positive/negative opinions in weblogs [91]. Later, researchers further introduced aspect-specific sentiment models in different ways, e.g., using supervision from sentiment priors [81, 68] or supervision from labeled sentences [128]. In [111], Titov and McDonald extended their multi-grain topic model [110] to discover topics that are representative of ratable aspects. Their regression module requires "ground truth" user ratings on the pre-defined aspects, which are not always available. In contrast, the proposed LARAM does not require aspect ratings from users and can decompose overall ratings into the ratings on the discovered aspects. More importantly, none of the work in this line is able to identify a reviewer's relative emphasis on different aspects, which is required for accurate interpretation of aspect ratings as we have discussed in Section 2.1.

Moreover, the work of LARA is also related to the semantic analysis of text documents in information extraction (IE) studies, e.g., entity and relation extraction [45, 90, 92, 127, 33]. Although, in our current solution, the aspects are defined by a set of keywords and the users' preferences are identified only based on the relation between latent aspect ratings and overall ratings, the work from IE can provide our solution of LARA with informative signals about users' latent intentions. For example, by identifying the named entities in review content, one can easily group the related entities by a domain-specific taxonomy and treat such clusters as latent aspects with high confidence (e.g., associating "restaurants" and "convenient stores" together as location aspect when analyzing hotel reviews). And by exploring the relations between different items (or aspects), e.g., comparative relation, we can precisely understand the users' preferences from the text content. All those signals can be incorporated into our solution of LARA: the identified named entities can be treated as a special form of features and included for aspect identification; and the recognized comparative relations among the aspects and items can be utilized as constraints (as in the $g(\cdot)$ function) to regularize the inferred aspect ratings and weights. And on the other hand, our LARA work can also provide additional signals for entity and relation extraction: e.g., by assuming consistency of a user's preference over certain entities, the inferred aspect ratings and weights can help the IE modules to perform better entity disambiguation. However, since our primary focus in this work is to provide solutions to the newly proposed LARA problem, we will leave the study of combining LARA with semantic analysis of text documents as our future work.

2.3 Problem Definition

In this section, we formally define the problem of Latent Aspect Rating Analysis (LARA).

As a computational problem, LARA assumes that the input is a set of reviews of some interesting entity (e.g., hotel), where each review has an overall rating. Such a format of reviews is quite common in most of the merchants' web site, e.g. Amazon (www.amazon.com) and Epinions (www.epinions.com).

Formally, let $D = \{d_1, d_2, \ldots, d_{|D|}\}$ be a set of review text documents for an entity or topic of interest, and each review document $d \in D$ is associated with an overall rating r_d. We also assume that there are n unique words in the vocabulary $V = \{w_1, w_2, \ldots, w_n\}$.

Definition (Overall Rating) An overall rating r_d of a review document d is a numerical rating indicating the level of overall opinion of d, i.e. $r_d \in [r_{min}, r_{max}]$, where r_{min} and r_{max} are the minimum and maximum ratings respectively.

In this work, the overall rating of a review is deemed as the observable behavior label.

We further assume that we are given k aspects, which are rating factors that potentially affect the overall rating of the given topic. For example, for hotel reviews, possible aspects may include "price" and "location." An aspect is specified through a few keywords, and provides a basis for latent aspect rating analysis.

Definition (Aspect) An aspect A_i is a (typically very small) set of words that characterize a rating factor in the reviews. For example, words such as "price," "value," and "worth" can characterize the price aspect of a hotel. We denote an aspect by $A_i = \{w | w \in V, A(w) = i\}$, where $A(.)$ is a mapping function from a word to an aspect label.

Definition (Aspect Ratings) Aspect rating $\mathbf{s}_d$ is a k dimensional vector, where the i-th dimension is a numerical measure, indicating the degree of satisfaction demonstrated in the review d toward the aspect A_i, and $\mathbf{s}_{di} \in [r_{min}, r_{max}]$. A higher rating means a more positive sentiment towards the corresponding aspect.

Definition (Aspect Weights) Aspect weight α_d is a k dimensional vector, where the i-th dimension is a numerical measure, indicating the degree of emphasis placed by the user of review

d on aspect A_i, where we require $\alpha_{di} \in [0,1]$ and $\sum_{i=1}^{k} \alpha_{di} = 1$ to make the weights easier to interpret and comparable across different reviews. A higher weight means more emphasis is put on the corresponding aspect.

The aspects, aspect ratings and aspect weights are treated as latent user intents, which connect the observed user review content (i.e., user behavior) and behavior labels (i.e., overall rating). As a result, the problem of "Latent Aspect Rating Analysis" is defined as,

Definition (Latent Aspect Rating Analysis (LARA)) Given a review collection D about a topic T where each review document d is associated with an overall rating r_d, and k aspects $\{A_1, A_2, \ldots, A_k\}$ to be analyzed, the problem of Latent Aspect Rating Analysis (LARA) is to discover each individual review's rating $\mathbf{s}_{di}$ on each of the k aspects as well as the relative emphasis α_{di} the reviewer has placed on each aspect.

2.4 Methods

Major challenges in solving the problem of LARA are: 1) we do not have detailed supervision about the latent rating on each aspect; 2) it is unclear how we can discover the relative weight placed by a user on each aspect. To solve these challenges, we propose a novel Latent Rating Regression (LRR) model to tie both latent ratings and latent weights with the review contents on the one hand and the overall rating of the review on the other. Specifically, we assume that the reviewer generates the overall rating of a review based on a weighted combination of his or her ratings on all aspects, and the rating on each aspect is generated based on another weighted combination of the words in the review that discusses the corresponding aspect. After fitting such a regression model on the given review data, we would be able to obtain the latent aspect ratings and weights, and thus solving the problem of LARA.

Since the LRR model assumes that we know the aspect segments in a review, we first perform aspect segmentation in a review document based on the given keywords describing aspects to obtain text segment(s) for each aspect. In the first approach, we assume each aspect can be described by a set of predefined keywords. Later on, we release this restriction by introducing a statistic topic modeling based approach to discover the possibly unknown aspects jointly with discovering the

latent ratings/weights on the identified aspects.

2.4.1 Keyword-based Aspect Segmentation

The goal of aspect segmentation is to map the sentences in a review into subsets corresponding to each aspect. In this approach we assume that a few keywords are given to describe each aspect, and thus we design a boot-strapping algorithm to obtain more related words for each aspect.

Specifically, the basic workflow of the proposed *Aspect Segmentation Algorithm* is as follows: given the seed words for each aspect and all the review text data as input, we assign each sentence to the aspect that shares the maximum term overlapping with this sentence; based on this initial aspect annotation, we calculate the dependencies between aspects and words by Chi-Square (χ^2) statistic [125], and include the words with high dependencies into the corresponding aspect keyword list. These steps are repeated until the aspect keyword list is unchanged or the number of iterations exceeds the limit. The full description of the algorithm is illustrated in Figure 2.2. The χ^2 statistic used to compute the dependencies between a term w and aspect A_i is defined as follows:

$$\chi^2(w, A_i) = \frac{C \times (C_1 C_4 - C_2 C_3)^2}{(C_1 + C_3) \times (C_2 + C_4) \times (C_1 + C_2) \times (C_3 + C_4)}$$

where C_1 is the number of times w occurs in sentences belonging to aspect A_i, C_2 is the number of times w occurs in sentences not belonging to A_i, C_3 is the number of sentences of aspect A_i that do not contain w, C_4 is the number of sentences that neither belong to aspect A_i, nor contain word w, and C is the total number of word occurrences.

After aspect segmentation, we would get k partitions of each review d, and represent them as a $k \times n$ feature matrix $\mathbf{W}_d$, where $\mathbf{W}_{dij}$ is the frequency of word w_j in the text assigned to aspect A_i of d normalized by the total counts of words in the text of that aspect.

2.4.2 Latent Rating Regression Model (LRR)

The LRR model treats $\mathbf{W}_d$ as independent variables (i.e., features of review d) and the overall rating r of the review as the response variable (i.e., variable to predict). In order to model the latent ratings on different aspects and the latent weights in the aspects, the LRR model further assumes that the overall rating is not directly determined by the review content, but rather, based

Algorithm: Aspect Segmentation Algorithm

Input: A collection of reviews $\{d_1, d_2, \ldots, d_{|D|})\}$, set of aspect keywords $\{T_1, T_2, \ldots, T_k\}$, vocabulary V, selection threshold p and iteration step limit I.
Output: Reviews split into sentences with aspect assignments.
Step 0: Split all reviews into sentences, $X = \{x_1, x_2, \ldots, x_M\}$;
Step 1: Match the aspect keywords in each sentence of X and record the matching hits for each aspect i in $Count(i)$;
Step 2: Assign the sentence an aspect label by $a_i = argmax_i\ Count(i)$. If there is a tie, assign the sentence with multiple aspects.
Step 3: Calculate χ^2 measure of each word (in V);
Step 4: Rank the words under each aspect with respect to their χ^2 value and join the top p words for each aspect into their corresponding aspect keyword list T_i;
Step 5: If the aspect keyword list is unchanged or iteration exceeds I, go to ***Step 6***, else go to ***Step 1***;
Step 6: Output the annotated sentences with aspect assignments.

Figure 2.2: Boot-strapping method for aspect segmentation.

on a set of latent ratings on different aspects which are more directly determined by the words associated with each aspect.

Formally, as we have defined in Section 2.3, $\mathbf{s}_d$ and α_d are review-level k-dimensional aspect rating vector and aspect weight vector. The reviewer in d is assumed to first generate an aspect rating for each A_i as a linear combination of $\mathbf{W}_{di}$ and β_i, i.e.

$$\mathbf{s}_i \leftarrow \sum_{j=1}^{n} \beta_{ij} \mathbf{W}_{dij} \tag{2.1}$$

where $\beta_i \in \Re$ indicates the word sentiment polarities on aspect A_i.

Then, the reviewer generates the overall rating based on the weighted sum of α_d and $\mathbf{s}_d$, i.e. $\alpha_d^T \mathbf{s}_d = \sum_{i=1}^{k} \alpha_{di} \mathbf{s}_{di}$. Specifically, the overall rating is assumed to be a sample drawn from a Gaussian distribution with mean $\alpha_d^T \mathbf{s}_d$ and variance δ^2, which indicates the uncertainty of the overall rating predictions. Thus putting all together, we have,

$$r_d \sim N(\sum_{i=1}^{k} \alpha_{di} \sum_{j=1}^{n} \beta_{ij} \mathbf{W}_{dij}, \delta^2) \tag{2.2}$$

Intuitively, the key idea here is to bridge the gap between the observed overall rating and the

detailed text descriptions by the latent aspect weight α_d and term sentiment weight β, which enable us to model the overall rating based on ratings of specific aspects.

Looking further into the user's rating behaviors, we find that reviewers' emphasis on different aspects can be complicated: 1) different reviewers might have different preferences for the aspects, e.g. business travelers may emphasize on internet service while honeymoon couples may pay more attention to rooms; 2) aspects are not independent, especially when the aspects have overlaps, e.g. an emphasis on cleanliness would indicate a preference to room quality too. In order to take the diversity of reviewer's preferences into consideration, we further treat the aspect weight α_d in each review d as a set of random variables drawn from an underline prior distribution for the whole corpus. Furthermore, to capture the dependencies among different aspects, we employ a multivariate Gaussian distribution as the prior for aspect weights, i.e.

$$\alpha_d \sim N(\mu, \mathbf{\Sigma}) \tag{2.3}$$

where μ and $\mathbf{\Sigma}$ are the mean and variance parameters.

Combining Eq (2.2) and (2.3), we get a Bayesian regression problem. The probability of observed overall rating in a given review in our LRR model is given by:

$$\begin{aligned} P(r|d) &= P(r_d|\mu, \mathbf{\Sigma}, \delta^2, \beta, \mathbf{W}_d) \\ &= \int p(\alpha_d|\mu, \mathbf{\Sigma}) p(r_d| \sum_{i=1}^{k} \alpha_{di} \sum_{j=1}^{n} \beta_{dij} \mathbf{W}_{dij}, \delta^2) \mathrm{d}\alpha_d \end{aligned} \tag{2.4}$$

where r_d and $\mathbf{W}_d$ are the observed data in review d, $\mathbf{\Theta} = (\mu, \mathbf{\Sigma}, \delta^2, \beta)$ are the set of *corpus-level* model parameters, and α_d is the latent aspect weight for review d. Note that we assume that δ^2 and β do not depend on individual reviewers, and are thus also *corpus-level* model parameters. A graphical model illustration of LRR model is given in Figure 2.3.

Posterior Inference

Suppose we are given the LRR model parameters $\mathbf{\Theta} = (\mu, \mathbf{\Sigma}, \delta^2, \beta)$, we can apply the model to get the aspect ratings and weights in each review as follows: (1) the latent aspect rating $\mathbf{s}_d$ in a particular review d could be calculated by Eq (2.1); (2) we appeal to the maximum a posteriori

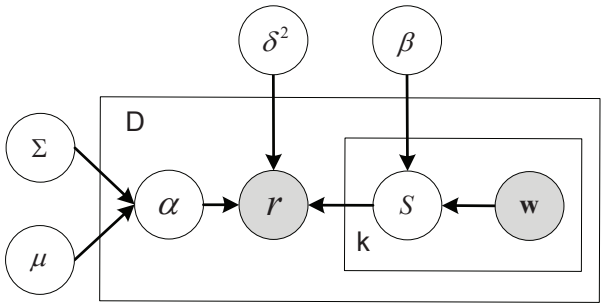


Figure 2.3: Graphical model representation of LRR. The outer box represents reviews, while the inner box represents the composition of latent aspect ratings and word descriptions within a review.

(MAP) estimation method to retrieve the most probable value of α_d in the given review. The object function of MAP estimation for review d is defined as:

$$\mathcal{L}(d) = \log p(\alpha_d|\mu, \mathbf{\Sigma})p(r_d|\sum_{i=1}^{k}\alpha_{di}\sum_{j=1}^{n}\beta_{dij}\mathbf{W}_{dij}, \delta^2) \tag{2.5}$$

We expand this object function and associate all the terms with respect to α_d in each review (denote as $\mathcal{L}(\alpha_d)$) as follows:

$$\begin{aligned}\hat{\alpha}_d &= \arg\max \mathcal{L}(\alpha_d) \\ &= \arg\max \left[-\frac{(r-\alpha_d^T\mathbf{s_d})^2}{2\delta^2} - \frac{1}{2}(\alpha_d-\mu)^T\mathbf{\Sigma}^{-1}(\alpha_d-\mu)\right] \\ \text{s.t.} \sum_{i=1}^{k}\alpha_{di} &= 1, \text{ and } 0 \le \alpha_{di} \le 1 \text{ for } i = 1, 2, \ldots, k\end{aligned} \tag{2.6}$$

To address above constraint optimization problem, we apply the conjugate-gradient-interior-point method with the following formula for the derivatives with respect to α_d:

$$\frac{\partial\mathcal{L}(\alpha_d)}{\partial\alpha_d} = -\frac{(\alpha_d^T\mathbf{s}_d - r_d)\mathbf{s}_d}{\delta^2} - \mathbf{\Sigma}^{-1}(\alpha_d-\mu)$$

Model Estimation

In the previous section, we discuss how to apply our LRR model to infer aspect weight α_d in each review d when given the model $\mathbf{\Theta} = (\mu, \mathbf{\Sigma}, \delta^2, \beta)$. In this section, we discuss how to estimate these model parameters using the Maximum Likelihood (ML) estimator, i.e., how to find the optimal

$\hat{\mathbf{\Theta}} = (\hat{\mu}, \hat{\mathbf{\Sigma}}, \hat{\delta^2}, \hat{\beta})$ that can maximize the probability of observing all the overall ratings.

The log-likelihood function on the whole set of reviews is:

$$\mathcal{L}(D) = \sum_{d \in D} \log p(r_d | \mu, \mathbf{\Sigma}, \delta^2, \beta, \mathbf{W}_d) \tag{2.7}$$

Thus the ML estimate is

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} \sum_{d \in D} \log p(r_d | \mu, \mathbf{\Sigma}, \delta^2, \beta, \mathbf{W}_d).$$

To compute this ML estimation, we would first randomly initialize all the parameters to obtain $\mathbf{\Theta}_{(0)}$ and then use the following EM-style algorithm to iteratively update and improve the parameters by executing the E-step and then M-step in each iteration:

E-Step: For each review d in the corpus, infer aspect rating $\mathbf{s}_d$ and aspect weight α_d based on the current parameter $\mathbf{\Theta}_{(t)}$ by using Eq (2.1) and (2.6).

M-Step: Given the inferred aspect rating $\mathbf{s}_d$ and aspect weight α_d based on the current parameters $\mathbf{\Theta}_{(t)}$, update the model parameters and obtain $\mathbf{\Theta}_{(t+1)}$ by maximizing the "complete likelihood," i.e., the probability of observing all the variables including the overall ratings r_d and the inferred aspect ratings $\mathbf{s}_d$ and aspect weights α_d for all the reviews.

First, we look at the case of updating the two parameters of the Gaussian prior distribution of the aspect weight α_d. Here our goal is to maximize the probability of observing all the α_d computed in the ***E-Step:*** for all the reviews, thus we have the following updating formulae based on the ML estimation for a Gaussian distribution.

$$\begin{aligned} \mu_{(t+1)} &= \arg\max_{\mu} - \sum_{d \in D} (\alpha_d - \mu)^T \mathbf{\Sigma}^{-1} (\alpha_d - \mu) \\ &= \frac{1}{|D|} \sum_{d \in D} \alpha_d \end{aligned} \tag{2.8}$$

$$\begin{aligned} \mathbf{\Sigma}_{(t+1)} &= \arg\max_{\mathbf{\Sigma}} \left[-|D| \log \mathbf{\Sigma} - \sum_{d \in D} (\alpha_d - \mu_{(t+1)})^T \mathbf{\Sigma}^{-1} (\alpha_d - \mu_{(t+1)}) \right] \\ &= \frac{1}{|D|} \sum_{d \in D} (\alpha_d - \mu_{(t+1)})(\alpha_d - \mu_{(t+1)})^T \end{aligned} \tag{2.9}$$

Second, we look at how to update β and δ^2. Since α_d is assumed to be known, we can update δ^2 and β to maximize $P(r_d|\alpha_d, \delta^2, \beta, \mathbf{W}_d)$ (defined in Equation 2.2). Solving this optimization problem, we have the following updating formulae:

$$\begin{aligned}\delta^2_{(t+1)} &= \arg\max_{\delta^2} \left[-|D| \log \delta^2 - \frac{\sum_{d\in D}(r_d - \alpha_d^T \mathbf{s}_d)^2}{\delta^2} \right] \\ &= \frac{1}{|D|} \sum_{d\in D} (r_d - \alpha_d^T \mathbf{s}_d)^2 \end{aligned} \tag{2.10}$$

$$\beta_{(t+1)} = \arg\max_{\beta} \sum_{d\in D} -\frac{(r_d - \sum_{i=1}^{k} \alpha_{di} \beta_i^T \mathbf{W}_{di})^2}{2\delta^2_{(t+1)}} \tag{2.11}$$

The closed-from solution for β requires an inversion on a $|V| \times |V|$ matrix, which is expensive to directly compute. To avoid this, we apply the gradient-based method to find the optimal solution of β with the following gradients:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta_i} = \sum_{d\in D} (\sum_{i=1}^{k} \alpha_{di} \beta_i^T \mathbf{W}_{di} - r_d) \alpha_{di} \mathbf{W}_{di}$$

The E-step and M-step are repeated until the likelihood value of Eq (2.7) converges.

2.4.3 A Fully Generative Model for LARA

Previously we assumed aspects can be specified by several predefined keywords. A main challenge in solving LARA without the aspect keywords supervision is to properly associate the review content with those meaningful aspects corresponding to the major opinions. To address this challenge, our basic idea is to use topic modeling techniques which provide a convenient way of segmenting the review contents by exploiting the word co-occurrence patterns in the data.

The basic assumption in LARAM is that the latent aspects of a particular entity are characterized by a set of coherent topics, which are shared across different reviews discussing the same entity (e.g., "*service*" and "*location*" for a hotel). The topics can be used to identify aspect text segments which contribute to the observed overall ratings in each review via the latent aspect ratings and weights. Along this line, we propose a fully generative framework to capture such dependency between the review contents and overall sentiment ratings.

We define and combine two components in LARAM: 1) an aspect modeling module based on

statistical topic modeling technique is introduced to discover the topical aspects from the review contents, and 2) the LRR model described in previous section is employed to infer the latent aspect ratings and weights based on the aspect segmented review contents.

In the aspect modeling part, we assume an aspect A_i can be characterized by a multinomial word distribution $Mul(\epsilon_i)$ over the vocabulary V. The proportion of aspects θ being discussed in each review d is drawn from a Dirichlet distribution $Dir(\gamma)$, where γ postulates the prior distribution of aspects in the whole corpus. Then, a review is treated as a mixture over the latent aspects, and the joint probability of observed word contents $\mathbf{W}$, latent aspect assignments $\{z_n\}_{n=1}^{|d|}$ and aspect proportion θ is defined as follows:

$$p(\mathbf{W}, \mathbf{z}, \theta|\gamma, \epsilon) = p(\theta|\gamma) \prod_{n=1}^{|d|} p(w_n|z_n, \epsilon)p(z_n|\theta) \tag{2.12}$$

where z_n is an indicator variable representing the latent aspect from which the word w_n is drawn.

In the rating analysis part, aspect rating $\mathbf{s}_i$ is still assumed to be determined by the aggregated sentiment over the text segments discussing aspect A_i as before:

$$\mathbf{s}_i = \sum_{n=1}^{|d|} \beta_{ij} \mathbf{\Delta}[w_n = v_j, z_n = i] \tag{2.13}$$

where we employ an indicator function $\mathbf{\Delta}[w_n = v_j, z_n = i]$ to map w_n to aspect A_i. In previous approach, such mapping is achieved via the keyword-based aspect segmentation.

As a result, the probability of observing the overall rating r and aspect weight α given the aspect segments in review d defined in LARAM is specified as:

$$p(r, \alpha|\mathbf{W}, \mu, \mathbf{\Sigma}, \delta^2) = p(\alpha|\mu, \mathbf{\Sigma})p(r|\sum_{i=1}^{k} \alpha_i \sum_{n=1}^{|d|} \beta_{ij} \mathbf{\Delta}[w_n = v_j, z_n = i], \delta^2)$$

Combining the two components, the joint probability of the observed text content $\mathbf{W}$ and

overall rating r in a given review d can thus be defined as:

$$
\begin{aligned}
& P(r, \mathbf{W}|\epsilon, \gamma, \beta, \mu, \mathbf{\Sigma}, \delta^2) \qquad (2.14)\\
=& P(\mathbf{W}|\epsilon, \gamma) \times P(r|\mathbf{W}, \beta, \mu, \mathbf{\Sigma}, \delta^2)\\
=& \iint \prod_{n=1}^{|d|} \sum_{z_n=1}^{k} p(w_n|z_n)p(z_n|\theta)p(\theta|\gamma)d\theta \times\\
& p(r|\sum_{i=1}^{k} \alpha_i \sum_{n=1}^{|d|} \beta_{ij} \mathbf{\Delta}[w_n = v_j, z_n = i], \delta^2)p(\alpha|\mu, \Sigma)d\alpha
\end{aligned}
$$

where we denote $\mathbf{\Theta}$ as the set of *corpus-level* model parameters. A graphical model illustration of the unified LARAM is shown in Figure 2.4.

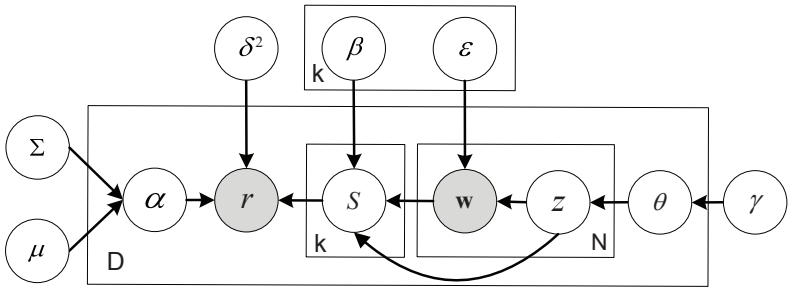


Figure 2.4: Graphical model representation of LARAM. The outer box represents reviews, while the inner box represents the composition of latent aspect rating regression and word generation within a review.

As illustrated in Figure 2.4, the word usage patterns embedded in the review text collection are captured by the aspect modeling part; the latent aspect assignments $\{z_n\}_{n=1}^{|d|}$ associate such word clusters with the corresponding aspect, and based on such probabilistic aspect segmentations, the latent aspect ratings and weights are discovered accordingly.

$$
r \sim N(\sum_{i=1}^{k} \alpha_i \sum_{n=1}^{|d|} \beta_{ij} \mathbf{\Delta}[w_n = v_j, z_n = i], \delta^2) \qquad (2.15)
$$

We should also note the proposed LARAM is not a trivial replacement of previous keyword-based bootstrapping method with topic models. Instead, it naturally bridges the gap between the aspect segmentation and sentiment analysis procedure in the previously used two-stage approach. There are two important factors distinguishing the proposed LARAM from the previous two-step

LRR model. First, in the current model, the latent aspect rating is a random variable, or more specifically, sum of a set of random variables (because each word's aspect assignment is a probability distribution over all the aspects), while in LRR model, it is treated as a fixed response variable of a rating regression module based on the given aspect segments. This difference enables us to model the uncertainty from the aspect segmentation part as well. Second, the latent rating analysis part provides informative feedback for the aspect segmentation part in the new model. Intuitively, in LARAM, we should assign the word w_j to a proper aspect A_i, in which the word's sentiment orientation β_{ij} is the most consistent with the identified orientation $\mathbf{s}_i$, or in other words, to be consistent with other words in its segment. In the previous LRR model, such segmentation is fixed once the set of aspect keywords is determined.

Posterior Inference

Given the model parameters $\mathbf{\Theta}$, the key step of applying the proposed LARAM is to infer the posterior distribution of the latent aspect assignment $\{z_n\}_{n=1}^{|d|}$ and aspect weight α in a given review d. Once this is done, the aspect ratings can be easily derived from Eq (2.13). In this work, we use variational inference method [73] due to its computational efficiency. The key idea behind variational approximation is to optimize the free parameters of a distribution over the latent variables so that the approximated distribution is close to the true posterior by Kullback-Leibler divergence.

More specifically, we introduce a family of factorized variational distribution for the hidden variables (z, θ, α) in each review d,

$$q(\mathbf{z}, \theta, \alpha|\phi, \eta, \lambda, \sigma^2) = q(\theta|\eta) \prod_n q(z_n|\phi_n) q(\alpha|\lambda, \sigma^2) \tag{2.16}$$

where the aspect assignment z for each word in d is specified by a k-dimensional multinomial distribution $Mul(\phi)$, aspect proportion θ is governed by a k-dimensional Dirichlet distribution $Dir(\eta)$, and aspect weight α is determined by a k-dimensional multivariate Normal distribution $N(\lambda, \sigma^2)$ with a diagonal covariance matrix to simplify the computation.

Based on the introduced variational distribution, the lower bound of the log-likelihood in review

d can be approximated as follows:

$$
\begin{aligned}
&\log p(r, \mathbf{W}|\epsilon, \gamma, \beta, \mu, \mathbf{\Sigma}, \delta^2) \\
=&\log p(\mathbf{W}|\epsilon, \gamma) + \log p(r|\mathbf{W}, \beta, \mu, \mathbf{\Sigma}, \delta^2) \\
\geq& E_q[\log p(\mathbf{z}, \theta, \mathbf{W}|\epsilon, \gamma)] - E_q[\log q(\mathbf{z}, \theta|\phi, \eta)] \\
+&E_q\big[\log p(r, \alpha, \mathbf{z}|\mathbf{W}, \beta, \mu, \mathbf{\Sigma}, \delta^2)\big] - E_q[\log q(\alpha, \mathbf{z}|\epsilon, \gamma, \phi, \eta)]
\end{aligned} \tag{2.17}
$$

where the bound is derived from the Jensen's inequality for convex function. In the following discussions, we will use $\mathcal{L}_d(\phi, \eta, \lambda, \sigma^2)$ to represent the lower bound defined by a set of variational parameters $(\phi, \eta, \lambda, \sigma^2)$ in review d.

The explanation for this lower bound is quite intuitive: the first part of $\mathcal{L}_d(\phi, \eta, \lambda, \sigma^2)$ represents the objective of aspect modeling module, which aims at finding the optimal aspect assignments $\{z_n\}_{n=1}^{|d|}$ and the corresponding aspect proportion θ for the observed review contents; the second part explains the objective of rating analysis module, which tunes both $\{z_n\}_{n=1}^{|d|}$ and α to fit the overall ratings. These two parts are not independently separated but connected by the common aspect assignments $\{z_n\}_{n=1}^{|d|}$. Both parts attempt to allocate proper aspect assignments to better accommodate the observed text contents and overall rating, respectively.

Details for calculating the expectation of the first part can be found in [14]. The expectation of the complete-data log-likelihood function for the rating analysis module under the variational distribution is derived as:

$$
\begin{aligned}
&E_q\big[\log p(r, \alpha, \mathbf{z}|\mathbf{W}, \beta, \mu, \mathbf{\Sigma}, \delta^2)\big] \\
=&-\frac{(\lambda^T \bar{\mathbf{s}} - r)^2}{2\delta^2} - \frac{1}{2\delta^2}\sum_{i=1}^{k}\Big\{(\lambda_i^2 + \sigma_i^2)Var[\mathbf{s}_i] + \sigma_i^2 \bar{\mathbf{s}_i}^2\Big\} \\
&-\frac{1}{2}(\lambda - \mu)^T\Sigma^{-1}(\lambda - \mu) - \frac{1}{2}Tr\big(diag(\sigma^2)\Sigma^{-1}\big) - \frac{1}{2}\log\delta^2 - \frac{1}{2}\log|\Sigma|
\end{aligned}
$$

where $\bar{\mathbf{s}_i} = \sum_{n=1}^{|d|}\beta_{ij}w_n^j\phi_{ni}$, $Var[\mathbf{s}_i] = \sum_{n=1}^{|d|}(\beta_{ij}w_n^j)^2\phi_{ni}(1-\phi_{ni})$ (w_n^j is a short representation for the indicator function $\mathbf{\Delta}[w_n = j]$).

Once the expectations in $\mathcal{L}_d(\phi, \eta, \lambda, \sigma^2)$ are analytically determined, an iterative fixed-point method is employed to find the set of variational parameters $(\phi, \eta, \lambda, \sigma^2)$ in order to maximize the lower bound of the original log-likelihood, which in turn would minimize the KL divergency

between the variational posterior and true posterior in each review. In particular, we compute the derivative of $\mathcal{L}_d(\phi, \eta, \lambda, \sigma^2)$ with respect to the variational parameters accordingly, and use them to find the optimal setting for each variational parameter.

In the aspect modeling part, the variational parameter η can be easily estimated as

$$\hat{\eta}_i = \gamma_i + \sum_{n=1}^{|d|} \phi_{ni} \tag{2.18}$$

Due to the involvement of rating analysis part, it is hard for us to get a closed-form solution for ϕ as in [14]. Therefore, we appeal to the gradient-based optimization procedure to obtain the optimal solution:

$$\begin{aligned}\hat{\phi}_n = \arg\max_{\phi_n} & \sum_{i=1}^{k} w_n^j \phi_{ni} \Big[\psi(\eta_i) - \psi(\sum_{j=1}^{k} \eta_j) + w_n^j \log \epsilon_{ij} - \log \phi_{ni} \Big] \\ & - \frac{1}{2\delta^2}(\lambda^T \bar{\mathbf{s}} - r)^2 - \frac{1}{2\delta^2} \sum_{i=1}^{k} \Big[(\lambda_i^2 + \sigma_i^2) Var[\mathbf{s}_i] + \sigma_i^2 \bar{\mathbf{s}_i}^2 \Big]\end{aligned} \tag{2.19}$$

s.t. $\forall$ i, $0 \leq \phi_{ni} \leq 1$ and $\sum_{i=1}^{k} \phi_{ni} = 1$

Similar procedure of gradient-based searching algorithm is applied to find the optimal solution of the variational parameter λ in the rating analysis part:

$$\hat{\lambda} = \arg\min_{\lambda} \Big\{ \frac{1}{2\delta^2} \Big[\sum_{i=1}^{k} \lambda_i^2 Var[\mathbf{s}_i] + (\lambda^T \bar{\mathbf{s}} - r)^2 \Big] + \frac{1}{2}(\lambda - \mu)^T \Sigma^{-1} (\lambda - \mu) \Big\} \tag{2.20}$$

s.t. $\forall$ i, $0 \leq \lambda_i \leq 1$ and $\sum_i^k \lambda_i = 1$.

Finally, σ^2 could be easily calculated as,

$$\sigma_i^2 = \frac{\delta^2}{Var[\mathbf{s}_i] + \bar{\mathbf{s}}_i^2 + \delta^2 \Sigma_{ii}^{-1}} \tag{2.21}$$

Model Estimation

In the previous section, we have discussed how to infer the latent aspect assignments $\{z_n\}_{n=1}^{|d|}$ and aspect weight α in each review d when given the model $\mathbf{\Theta}$. In this section, we discuss how to estimate these *corpus-level* parameters using the Expectation Maximization (EM) algorithm by maximizing the expectation of observing review contents and the overall ratings in a given review

document collection.

As defined in Eq (2.17), we can approximate the log-likelihood in each individual review d by the introduced variational distribution $q(z, \theta, \alpha|\phi, \eta, \lambda, \sigma^2)$. Since the variational inference is carried out independently for each review in the collection, the log-likelihood over the whole collection D is simply a summation over the lower bound of each review:

$$\mathcal{L}(D) = \sum_{d \in D} \mathcal{L}_d(\phi, \eta, \lambda, \sigma^2) \tag{2.22}$$

Following similar procedures used in posterior inference in LARAM, we maximize Eq (2.22) by finding the optimal *corpus-level* model parameters $\mathbf{\Theta}$.

The detailed procedures for updating the parameters in the aspect modeling part is the same as derived in [14], so we only list them here:

$$\epsilon_{ij} \propto \sum_{d}^{D} \sum_{n}^{N_d} \phi_{dni} w_{dn}^{j} \tag{2.23}$$

$$\frac{\partial \mathcal{L}(\gamma)}{\partial \gamma_i} = D\Big[\psi(\sum_j \gamma_j) - \psi(\gamma_i)\Big] + \sum_{d}^{D} \Big[\psi(\eta_{di}) - \psi(\sum_j \eta_{dj})\Big] \tag{2.24}$$

$$\frac{\partial^2 \mathcal{L}(\gamma)}{\partial \gamma_i \partial \gamma_j} = D\big[\psi'(\sum_j \gamma_j) - \sigma(i,j)\psi'(\gamma_i)\big] \tag{2.25}$$

The updating equations for the aspect weight prior $(\mu, \mathbf{\Sigma})$, and overall rating prediction variance δ^2 in the rating analysis part can be easily obtained from their sufficient statistics accordingly, as we did in keyword-based two-step LRR:

$$\hat{\mu} = \frac{1}{D} \sum_{d=1}^{D} \lambda_d \tag{2.26}$$

$$\hat{\Sigma} = \frac{1}{D} \sum_{d=1}^{D} \big[(\lambda_d - \hat{\mu})(\lambda_d - \hat{\mu})^T + diag(\sigma_d^2)\big]\hat{\delta}^2 \tag{2.27}$$

$$= \frac{1}{D} \sum_{d=1}^{D} \Big\{(r_d - \lambda_d^T \bar{\mathbf{s}}_d)^2 + \sum_{i=1}^{k} \big[(\lambda_{di}^2 + \sigma_{di}^2) Var[\mathbf{s}_{di}] + \sigma_{di}^2 \bar{\mathbf{s}}_{di}^2\big]\Big\} \tag{2.28}$$

The same as before, a closed-form updating formula for the term weight matrix β is hard to

write out:

$$\hat{\beta} = \arg\min_{\beta} \sum_{d}^{D} \left\{ (\lambda_d^T \bar{\mathbf{s}}_d - r_d)^2 + \sum_{i=1}^{k} \left[(\lambda_{di}^2 + \sigma_i^2) Var[\mathbf{s}_{di}] + \sigma_{di}^2 \bar{\mathbf{s}}_{di}^2 \right] \right\} \tag{2.29}$$

We apply the gradient-based optimization technique to find the optimal solution of β with the following gradients:

$$\frac{\partial \mathcal{L}(\beta_i)}{\partial \beta_{ij}} = \sum_{d}^{D} \Big[(\lambda_d^T \bar{\mathbf{s}}_d - r_d)\lambda_{di} + \sigma_{di}^2 \bar{\mathbf{s}}_{di} + (\lambda_{di}^2 + \sigma_{di}^2)\beta_{ij} w_{dn}^j (1 - \phi_{dni}) \Big] \phi_{dni} w_{dn}^j \tag{2.30}$$

From the above equation, we can find another evidence of the interaction between aspect modeling module and rating analysis module: an optimal word sentiment polarity setting should not only ensure the sentiment orientation expressed in the review content consist with the observed overall sentiment judgment, but also reduce the uncertainty on each latent aspect's opinion prediction, which would help the model allocate the aspect assignment for each word more accurately.

A similar EM algorithm as used for two-step LRR model is applied to iteratively update and improve the parameters by alternatively executing the *E-step* with posterior inference and *M-step* for model parameter estimation in each iteration until the log-likelihood defined in Eq (2.22) converges.

2.5 Experimental Results

We included two review data sets for evaluation purpose in our experiments: a hotel review data set crawled from www.tripadvisor.com, and an MP3 player review data set crawled from www.amazon.com. In the hotel review data set, in addition to the overall ratings, reviewers are also asked to provide ratings on 7 pre-defined aspects in each review: i.e., *value, room, location, cleanliness, check in/front desk, service*, and *business service*, ranging from 1 star to 5 stars. This can serve as the ground-truth for quantitative evaluation of both aspect identification and latent aspect rating prediction. In the MP3 player review data set, there is only one overall rating in each review, ranging from 1 star to 5 stars.

We first performed simple pre-processing on these two data sets: 1) remove the reviews with any missing aspect rating or document length less than 50 words (to keep the content coverage of all possible aspects); 2) convert all the words into lower cases; and 3) remove punctuations, stop

words defined in [1], and the terms occurring in less than 10 reviews in each collection. After the pre-processing, we have 37,181 hotel reviews and 16,680 MP3 player reviews; the detailed statistics are listed in Table 2.1.

Table 2.1: Statistics of review data sets.

	#Item	#Review	#Reviewer	Avg Len	Rating
Hotel	2,232	37,181	34,187	96.5	3.92±1.23
MP3	686	16,680	15,004	87.3	3.76±1.41

To apply the keyword-based two-step LRR model, we need to manually specify the aspects by carefully selecting a small set of seed words. Table 2.2 shows the initial aspect seed words we used as input to the bootstrapping aspect segmentation algorithm described in Section 2.4.1, where we set the selection threshold p=5 and iteration step limit I=10 in our experiments. .

Table 2.2: Aspect seed words.

Aspects	**Seed words**
Value	value, price, quality, worth
Room	room, suite, view, bed
Location	location, traffic, minute, restaurant
Cleanliness	clean, dirty, maintain, smell
Check In/Front Desk	stuff, check, help, reservation
Service	service, food, breakfast, buffet
Business service	business, center, computer, internet

2.5.1 Aspect Identification

In Amazon reviews, the reviewers are only asked to give an overall rating, so they would have more freedom, or less guidance to write the comments. In this case, it is *very difficult* to pre-specify aspects in keywords without necessary domain-knowledge. Here, we will first qualitatively demonstrate that our unified model, LARAM, can automatically identify meaningful aspects based on the data characteristics. We separate the reviews into two subsets, one with low overall ratings (at most 3 stars) and the other with high overall ratings (at least 4 stars), and apply LARAM to extract 20 aspects on each subset.

It is expected that users usually comment on different aspects in positive and negative reviews. In Table 2.3, we show the top 10 words with the highest generation probability under the three

Table 2.3: Topical aspects learned on MP3 player reviews.

Low Overall Ratings			High Overall Ratings		
unit	jack	service	files	player	vision
usb	headphone	charge	format	music	video
battery	warranty	problem	included	download	player
charger	replacement	support	easy	headphones	quality
reset	problem	hours	convert	button	great
time	player	months	mp3	set	product
hours	back	weeks	videos	hours	sound
work	months	back	file	buds	radio
thing	buy	customer	wall	volume	accessory
wall	amazon	time	hours	ear	fm

aspects with the largest aspect weight prior μ, which can be considered as the aspects contributing the most to overall ratings. We can observe that LARAM automatically adapts to such data characteristics: in the negative reviews, the most complained aspects are about warranty and service; while the positive reviews emphasize the admirable product features such as flexible file format and great video quality.

Next, we quantitatively compare our model with existing methods on the quality of identified topical aspects.

Algorithms for Comparison: Since we employed the statistical topic models to discover aspects, we compare LARAM with two different topic models: unsupervised **LDA** [14] model and supervised **sLDA** [13] model. LDA behaves similarly as our aspect modeling module, but it can only fit the word co-occurrence patterns in the review content. sLDA extends LDA by adding a regression module to model the observed overall response, so that it uses the same input as our LARAM. However, since sLDA and LARAM employ different generation assumptions for the overall response, it would be interesting to compare these two models.

Measure: In the hotel data set, because TripAdvisor asks its reviewers to rate the predefined 7 aspects, it is reasonable to assume those are the major aspects most reviewers comment about. Thus, we use the full set of keywords (in total 309 words) generated by the bootstrapping method as prior to train a LDA model on this data set, and treat the learned topics as the "ground-truth" aspect descriptions. Then we train all the three models *without* any supervision of aspect keywords, calculate the optimal alignment between the learned topics with "ground-truth" aspects by Kuhn-

Table 2.4: KL divergence between the align aspects.

	LDA	sLDA	LARAM
7 topics	**5.675**	14.878	5.827
14 topics	8.819	19.074	**8.356**
21 topics	12.745	22.411	**11.167**

Munkres algorithm [86] and quantitatively measure the quality of the identified aspects using KL divergence:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

where p(x) is the "ground-truth" word distribution, and q(x) is the word distribution learned by a given model.

Result Analysis: We train these three models with 7, 14 and 21 topics (since we already know there are 7 aspects) on the hotel data set separately and list the results in Table 2.4. From the results, we can find that compared with the unsupervised LDA model, the aspects identified by LARAM are closer to the ground-truth aspects when we have more topics (i.e., smaller KL divergence); sLDA's performance is the worst, even though it has additional information from the overall ratings for estimating the topics. From the comparison we can see that the inferred aspect ratings from LARAM help the aspect model to better allocate the words across different aspects, which would not be easily distinguished solely from the co-occurrence patterns (LARAM versus LDA). The sLDA model assumes that the topics (aspects) directly characterize the overall ratings, rather than the words' sentiment polarities in the specific aspects as we assumed in LARAM, so it prefers rating-sensitive words than those aspect-specific words. As a result, the word distribution under each topic learned by sLDA is quite different from those aspects specified by the keywords. Besides, we can observe that when we use more topics, KL divergence gets larger. The reason is that with more topics, all the models will then have more freedom to distinguish the aspects in a finer granularity, which is not covered by the predefined aspect keywords.

2.5.2 Aspect Rating Prediction

Although the problem of LARA is designed to infer ratings on any discovered topical aspects, we can only quantitatively evaluate the predicted ratings on the collection where we have the ground-truth aspect ratings from reviewers. In this experiment, we use the hotel review data set as the testing corpus.

Since we have proposed two different approaches to solve the problem of LARA, i.e., keyword-based two-step LRR model and topic-model-based unified LARAM model, we will employ different set of baseline methods to make comprehensive comparisons for the two methods we have proposed.

Measures: We quantitatively evaluate the algorithms using six different measures, including: (1) Mean Square Error (Δ^2_{aspect}) of the predicted aspect ratings compared with the ground-truth aspect ratings; (2) Pearson correlation inside reviews (ρ_{aspect}) measures how well the predicted aspect ratings can preserve the relative order of aspects within a review given their ground-truth ratings; (3) percentage of mis-ordered aspects inside reviews (Mis_{aspect}) measures the cases when the predicted aspect ratings confuse the best and worst aspects within reviews (if they are different as in ground-truth); (4) nDCG of aspect ranking inside reviews ($nDCG_{aspect}$) evaluates the model's ranking accuracy of aspects inside a review, where the ground truth aspect ratings are used as the graded relevance in the measure; (5) Pearson correlation across hotels (ρ_{hotel}) measures how well the predicted aspect ratings (averaging over the predicted aspect ratings of all the reviews commenting on this hotel) can preserve the relative order of hotels by their ground-truth ratings; and (6) Mean Average Precision (MAP_{hotel}@10) evaluates the model's ranking accuracy of hotels. In MAP_{hotel}@10, we treat each aspect as a query, the top 10% of hotels ranked by the ground-truth aspect ratings as the relevant answers, and test whether we are able to rank these relevant hotels on the top, if we use the predicted aspect ratings to rank them. Those measurements can be categorized into two different groups: aspect-level evaluation, including the first four metrics, which are averaged over all the reviews, and hotel-level evaluation, including the last two metrics, which are averaged over all the aspects.

Algorithms for comparison with two-step LRR: The closest work to our two-step LRR model is [87], in which the authors proposed two methods, i.e. **Local prediction** and **Global prediction**, to solve a similar problem. Therefore we take these two methods as our baseline

methods for comparison. We also include another baseline, in which we take the overall rating of a review as the rating for each aspect and train a supervised regression model based on such data. We implement this method using the Support Vector Regression (SVR) model [21] and name it as **SVR-O**. Besides, as an upper-bound algorithm, we also test a fully supervised algorithm **SVR-A**, i.e., SVR model fed with the aspect ratings in the ground-truth for training, and compare it with what LRR can achieve without such supervision. Both SVR-O and SVR-A use the same aspect segmentation results from our keyword-based segmentation method, and we use RBF kernel with default parameters implemented in the libsvm package [24] for SVR-O and SVR-A.

All the models are evaluated on the same data set: for Local prediction method, Global prediction method and LRR, we use the whole data set for both training and testing; for SVR-based models, we perform 4-fold cross validation and report the mean value of performance.

Result Analysis (1): First, we report the performance comparison between the two-step LRR with five baseline algorithms in Table 2.5. Since SVR-A is fully supervised while others are not, we list it separately on the last line as an upper bound. We also highlight the best performance in each measure for all the non SVR-A models in bold.

Table 2.5: Performance comparison between two-step LRR with other models.

Method	Δ^2_{aspect}	ρ_{aspect}	ρ_{review}	MAP_{hotel}@10
Local prediction	**0.588**	0.136	**0.783**	0.131
Global prediction	0.997	0.279	0.584	0.000
SVR-O	0.591	0.294	0.581	0.358
LRR	0.896	**0.464**	0.618	**0.379**
SVR-A	0.306	0.557	0.673	0.473

A general observation is that LRR performs much better than all other non SVR-A models on ρ_{aspect} and MAP@10, but it does not perform the best on Δ^2_{aspect} and ρ_{review}. High ρ_{aspect} means that LRR can better distinguish the ratings of different aspects within a review. Note that such information about a reviewer's relative preferences on different aspects cannot be obtained only with an overall rating. In addition, the high MAP_{hotel}@10 performance show that LRR also can better recognize the top 10 hotels under each aspect based on inferred aspect rating than other methods, leading to more useful ranking results from a user's perspective since it is the top ranked results that would affect user's satisfaction most.

Note that SVR fed with overall ratings did not achieve desirable performance, which to some extent confirms our assumption that aspect ratings are different from overall ratings. As a result, looking at only the overall ratings is insufficient to understand user's detailed opinions. Not surprisingly, LRR does not perform as well as SVR-A, which was trained with ground-truth aspect ratings. However, LRR does not require any annotated aspect ratings for training, and can thus be applied to more application scenarios than SVR-A.

Algorithms for Comparison with LARAM: As an alternative method to the proposed unified **LARAM** model, we can take a two-stage approach: applying topic models (e.g., LDA or sLDA) first to identify the aspect segments and then apply LRR to predict aspect ratings. Therefore we include two methods for comparison, i.e., **LDA+LRR** and **sLDA+LRR**. Since we are only interested in predicting the latent aspect ratings, we used all the data for both training and testing of LDA+LRR, sLDA+LRR and LARAM, as we did in evaluating the two-step LRR model.

Result Analysis (2): In general, LARAM outperformed other methods in all measures except that its MAP_{hotel}@10 performance is basically the same as sLDA+LRR. The top part of the table shows MSE, which directly measures the difference between the predicted aspect ratings and ground truth ratings. The middle part shows the performance of ranking different aspects within a review. All the three measures of ρ_{aspect}, $nDCG_{aspect}$ and Mis_{aspect} show superior performance of the proposed unified model in aspect ranking inside reviews which can answer questions like "Do the reviewer like the *location* better than *cleanliness* of hotel XXX?" Finally, the bottom part of the table demonstrates the model's ranking capability of hotels based on the predicted aspect ratings. Since we average the aspect ratings from individual reviews to get aspect ratings for the same hotel, which are in real value, there is no bias in the ρ_{hotel} measure.

2.5.3 Applications

The detailed understanding of opinions obtained by solving the problem of LARA can be potentially useful for many applications. Here we present four novel applications based on the keyword-based two-step solution, and similar results can also be achieved by the unified LARAM model.

Table 2.6: Aspect rating prediction performance on hotel reviews.

	LDA+LRR	sLDA+LRR	LARAM
Δ^2_{aspect}	2.130	2.360	**1.234**
ρ_{aspect}	0.080	0.079	**0.228**
Mis_{aspect}	0.439	0.439	**0.387**
$nDCG_{aspect}$	0.860	0.886	**0.901**
ρ_{hotel}	0.558	0.450	**0.622**
MAP_{hotel}@10	0.427	**0.437**	0.436

Corpus Specific Word Sentimental Orientation

In addition to predicting the latent aspect ratings and weights for the whole text, LRR can also identify the word's sentimental orientations. Being different from traditional unsupervised sentiment classification methods, which rely on a predefined lexicon, LRR can uncover such sentimental information directly from the given data. In Table 2.7, we show some interesting results of LRR on the hotel review set by listing the top 5 words with positive weights and top 5 words with negative weights for each aspect, and we compare them with the opinion annotations in *SentiWordNet* [44].

We can find interesting results in Table 2.7: the word "ok" is positive as defined by *SentiWordNet*, but in our corpus reviewers use this word to comment on something barely acceptable; words "linen", "walk" and "beach" do not have opinion annotations in *SentiWordNet* since they are nouns, while LRR assigns them positive sentiment likely because "linen" may suggest the "cleanliness" condition is good and "walk" and "beach" might imply the location of a hotel is convenient. Thus, LRR can provide us with word orientation information that is specific to the given domain, which is useful for augmenting an existing sentiment lexicon for specific domains.

Aspect-Based Summarization

Since LRR can infer the aspect ratings $\mathbf{s}_d$ for each review d, we can easily aggregate the aspect ratings of all reviews about the same hotel to generate one numerical rating for each aspect of a given hotel (e.g. $\frac{1}{|D|}\sum_{d\in D}\mathbf{s}_d$). Such aspect ratings for a hotel can be treated as a concise aspect-based opinion summary for the hotel. On top of that, we can also select the sentences in each review about the given hotel by calculating the aspect scores according to Eq (2.1), and selecting the highest and lowest scored sentences for each aspect to help users better understand the opinions

Table 2.7: Estimated word sentiment polarities under aspects.

Value	Rooms	Location	Cleanliness
resort 22.80	view 28.05	restaurant 24.47	clean 55.35
value 19.64	comfortable 23.15	walk 18.89	smell 14.38
excellent 19.54	modern 15.82	bus 14.32	linen 14.25
worth 19.20	quiet 15.37	beach 14.11	maintain 13.51
quality 18.60	spacious 14.25	perfect 13.63	spotlessly 8.95
bad -24.09	carpet -9.88	wall -11.70	smelly -0.53
money -11.02	smell -8.83	bad -5.40	urine -0.43
terrible -10.01	dirty -7.85	mrt -4.83	filthy -0.42
overprice -9.06	stain -5.85	road -2.90	dingy -0.38
cheap -7.31	ok -5.46	website -1.67	damp -0.30

on different aspects.

We show a sample aspect-based summary generated in this way in Table 2.8. We can find that reviewers agree that Hotel Max's price is excellent when considering its great location in Seattle. However, there is also room for improvement: poor heating system and the charge for Internet access. This kind of detailed information would be very useful to the users for digesting the essential opinions in a large number of reviews.

Table 2.8: Aspect-based comparative summarization (Hotel Max in Seattle).

Aspect	Summary	Rating
Value	Truly unique character and a great location at a reasonable price Hotel Max was an excellent choice for our recent three night stay in Seattle.	3.1
	Overall not a negative experience, however considering that the hotel industry is very much in the impressing business there was a lot of room for improvement.	1.7
Room	We chose this hotel because there was a Travelzoo deal where the Queen of Art room was $139.00/night.	3.7
	Heating system is a window AC unit that has to be shut off at night or guests will roast.	1.2
Location	The location ,a short walk to downtown and Pike Place market , made the hotel a good choice.	3.5
	when you visit a big metropolitan city, be prepared to hear a little traffic outside!	2.1
Business Service	You can pay for wireless by the day or use the complimentary Internet in the business center behind the lobby though.	2.7
	My only complaint is the daily charge for internet access when you can pretty much connect to wireless on the streets anymore.	0.9

User Rating Behavior Analysis

By inferring hidden aspect weights α_d from each individual review, we can recognize the relative emphasis placed by a reviewer on different aspects, which can be regarded as knowledge about a user's rating behavior. One potential application of analyzing the reviewers' rating behavior is to discover what factors have the most influence on reviewers' judgment when they make such evaluations. To look into this, we selected two groups of hotels with different price ranges: one group have prices over $800, which would be named as "expensive hotels," while the other group have prices below $100 and would be named as "cheap hotels." For each group, we then selected top 10 and bottom 10 hotels based on their average overall ratings, resulting in four subgroups of hotels. We show the average aspect weights α_d of these four different subgroups of hotels in Table 2.9. (In the group of "expensive hotels," the lowest overall rating is 3 stars.)

Table 2.9: User rating behavior analysis.

	Expensive Hotel		Cheap Hotel	
Aspect	5 Stars	3 Stars	5 Stars	1 Star
Value	0.134	0.148	**0.171**	0.093
Room	0.098	0.162	0.126	0.121
Location	0.171	0.074	0.161	0.082
Cleanliness	0.081	**0.163**	0.116	**0.294**
Service	**0.251**	0.101	0.101	0.049

It is interesting to note that reviewers give the "expensive hotels" high ratings mainly due to their nice services and locations, while they give low ratings to such hotels because of undesirable room condition and overprice. In contrast, reviewers give the "cheap" hotels high ratings mostly because of the good price/value and good location, while giving low ratings for its poor cleanliness condition.

Additionally, those numerical ratings may contain different meanings across various reviewers: users with a low budget might give a cheaper hotel 5 star rating of "*value*", while some others seeking for better service might also give a more expensive hotel 5 star rating for its "*value*". Only predicting the ratings for the aspects is not enough to exploit such subtle differences between the users. We appeal to some external knowledge to verify the correctness of the inferred aspect weights, which can uncover the differences among the users who give the same aspect rating for

different reasons or by different standards. We select the hotels with the same 5 star *"value"* ratings from 4 different cities: *Amsterdam*, *Barcelona*, *Florence* and *San Francisco*, which possess the largest volume of hotels in our corpus. In Table 2.10, we rank those hotels according to ratios of value/location weight, value/room weight and value/service weight, and calculate the average price of top 10 (in the upper part of corresponding column) and bottom 10 (in the lower part of corresponding column) ranked hotels.

We find that hotels with relatively higher *"value"* weights tend to have a lower price while the hotels with higher *"location"*, *"room"* and *"service"* weights tend to have a higher price. That is a reasonable result: for hotels with the same overall ratings, people who care value over other aspects would prefer a cheaper hotel and others who prefer a better location or service would accept a higher price. This interesting result indicates the soundness of the inferred aspect weights α_d and indicate potential applications of this inferred reviewers rating behavior.

Table 2.10: Hotel group identification by weight ratios

City	Avg. Price	Value/Location	Value/Room	Value/Service
Amsterdam	241.6	190.7	214.9	221.1
		270.8	333.9	236.2
Barcelona	280.8	270.2	196.9	263.4
		330.7	266.0	203.0
San Francisco	261.3	214.5	249.0	225.3
		321.1	311.1	311.4
Florence	272.1	269.4	248.9	220.3
		298.9	293.4	292.6

Personalized Ranking

The learned weights on different aspects at the level of each individual review would enable us to personalize hotel ranking by selectively using only the reviews written by reviewers whose rating behavior is similar to a current user. Specifically, given a specific user's weighting preference as a query, we can select the reviewers whose weighting preferences are similar, and rank the hotels only based on the reviews written by those "similar" reviewers.

To show the effectiveness of LARA in supporting such personalized ranking, consider a sample query: Query = {value weight:0.9, others:0.016}, which indicates that the user cares most about

the "value" aspect and does not care about other aspects. We use two different ranking approaches: 1) approach 1 would simply rank the hotels by the predicted aspect rating without considering the input query; 2) approach 2 would select the top 10% reviewers who have the closet aspect weights (i.e. α_d) to the query and predict the associated hotels aspect ratings only based on those selected reviews. Using both approaches, we rank hotels based on the weighted sum of the predicted aspect ratings for all the aspects with the weight defined in the query (thus the rating on "value" would contribute most to final ranking), and show the top-5 returned hotels using each approach in Table 2.11.

Table 2.11: Personalized Hotel Ranking

	Hotel	Overall Rating	Price	Location
Approach 1	Majestic Colonial	5.0	339	Punta Cana
	Agua Resort	5.0	753	Punta Cana
	Majestic Elegance	5.0	537	Punta Cana
	Grand Palladium	5.0	277	Punta Cana
	Iberostar	5.0	157	Punta Cana
Approach 2	Elan Hotel Modern	5.0	216	Los Angeles
	Marriott San Juan Resort	4.0	354	San Juan
	Punta Cana Club	5.0	409	Punta Cana
	Comfort Inn	5.0	155	Boston
	Hotel Commonwealth	4.5	313	Boston

It is interesting to find that although the top-5 results from approach 1 all have 5-star overall ratings (presumably they also have high ratings on "value" since the ranking is based on weights specified in the query), their prices tend to be much higher than the top-5 results returned from approach 2; indeed, the average price of the top-5 hotels from approach 1 is $412.6, while that of the top-5 hotels from approach 2 is only $289.4, which is much lower. (The average price of all the hotels in the data set is $334.3). Intuitively, for this sample query, the results of approach 2 are more useful to the user. This means that due to the selective use of reviews from reviewers who have a similar weight preference to the query, approach 2 is able to personalize the ranking and correctly place more weight on the "value" aspect to ensure that the top-ranked hotels really have relatively low prices.

2.6 Conclusion

In this chapter, we defined a novel text mining problem named Latent Aspect Rating Analysis (LARA) to analyze opinions expressed in online reviews at the level of topical aspects. LARA takes a set of review texts with overall ratings as input, and discovers each individual reviewer's latent ratings on the given aspects and the relative emphasis a reviewer has placed on different aspects. To solve this problem, we proposed a novel two-step Latent Rating Regression (LRR) model and a unified Latent Aspect Rating Analysis Model (LARAM). Our empirical experiments on a hotel review data set and a MP3 player review data set showed that the proposed models can effectively solve the problem of LARA, revealing interesting differences between aspect ratings and overall ratings, as well as differences in users' rating behaviors. The results also show that the detailed analysis of opinions at the level of topical aspects enabled by the proposed model can support multiple application tasks, including aspect opinion summarization, ranking of entities based on aspect ratings, and analysis of reviewers rating behavior.

Our work opens up an interesting new direction in text mining where the focus is on analyzing latent ratings in opinionated text. There are many interesting future research directions to further explore. For example, although we defined LARA based on reviews, LARA is clearly also applicable to any set of opinionated text documents with overall opinion judgments to achieve detailed understandings of opinions. It would be interesting to explore other possible application scenarios. Besides, both of our LRR models (i.e., two-step LRR and LARAM) are not strictly limited to word features, other kinds of features could be easily embedded into the proposed models.

Chapter 3

ReviewMiner: A Multi-Modal Opinion Analysis System for Decision Support

The proposed solutions to the latent aspect rating analysis problem studied in Chapter 2 open new directions for opinion mining applications. In this chapter, based on the techniques studied in Chapter 2, a prototype system called ReviewMiner [1] is developed to support multi-modal opinion analysis and decision making based on opinionated review text documents.

3.1 System Overview

Despite the abundant studies in opinion mining research, to the best of our knowledge, there is few practical system providing easy access of opinions to ordinary users for making informed decisions nor to merchants for improving their services. Therefore, in this thesis work, we aim at building such an opinion analysis system, which processes and indexes opinionated review documents and provides multi-modal analysis of opinions to the end users for making informed decisions.

As we have demonstrated in Section 2.5.3, the detailed understanding of opinions at the level of topical aspects obtained by solving the problem of LARA enables many interesting new applications in both text analytics and business intelligence. More importantly, such applications give ordinary users the ability to access large amount of opinionated text data easily. In order to grant such power to the end users in practice, ReviewMiner not only provides basic search functions for the end users to explore the analyzed entities and reviews in the system, but also personalizes the retrieved results according to the users' input preferences over the identified aspects of different entities, recommends similar entities based on the detailed aspect-level opinions, and summarizes the aspect-level opinions in textual, temporal and spatial dimensions. The function of personalization and recommendation assists the users to identify the results of interest and explore alternative choices more efficiently.

[1] http://timan100.cs.uiuc.edu:8080/ReviewMiner/

And the unique multi-modal opinion summarization and visualization mechanism provides the system users various perspectives to digest information from review content when making informed decisions.

In addition, ReviewMiner also actively records users' interactive search behaviors in the system, including input queries, clicked results, update of aspect preference and helpful votes of the retrieved review documents. The logged information is then utilized to analyze users' search intent and build accurate user models for assisting her in the future. What's more, the logged users' information seeking behaviors provide a precious mine for the research of "modeling interactive behavior data for system optimization" conducted in the second part of this thesis.

3.2 System Description

Figure 3.1 highlights the overview of the developed ReviewMiner system. There are four major components in this system: 1) review document crawler, LARA analyzer and analyzed review repository; 2) query parser; 3) multi-model user interface; and 4) interaction behavior logging system and user modeling. In the following, we will discuss the implementation details of each component accordingly.

3.2.1 Crawler, Analyzer and Review Repository

This component forms a pipeline to collect, analyze and store the online opinionated review documents into a structured database.

• **Crawler:** ReviewMiner keeps crawling opinionated review documents from three websites: 1) TripAdvisor (www.tripadvisor.com) for hotel reviews; 2) Amazon (www.amazon.com) for product reviews; 3) WebMD (www.webmd.com) for medication reviews. In particular, in the Amazon product reviews, we focus on six subcategories of products, i.e., digital cameras, TVs, video surveillance systems, mobile phones, tablets, and laptops. Basic information of an item, e.g., name, image, overall ratings and short descriptions (i.e., address for hotels, feature specification for Amazon products, and usage for medications), is collected while crawling. And in each review, the information about author, review date, title and content is collected. Simple filtering is performed at the crawling stage: 1) items with less than five reviews are discarded; 2) reviews with less than three

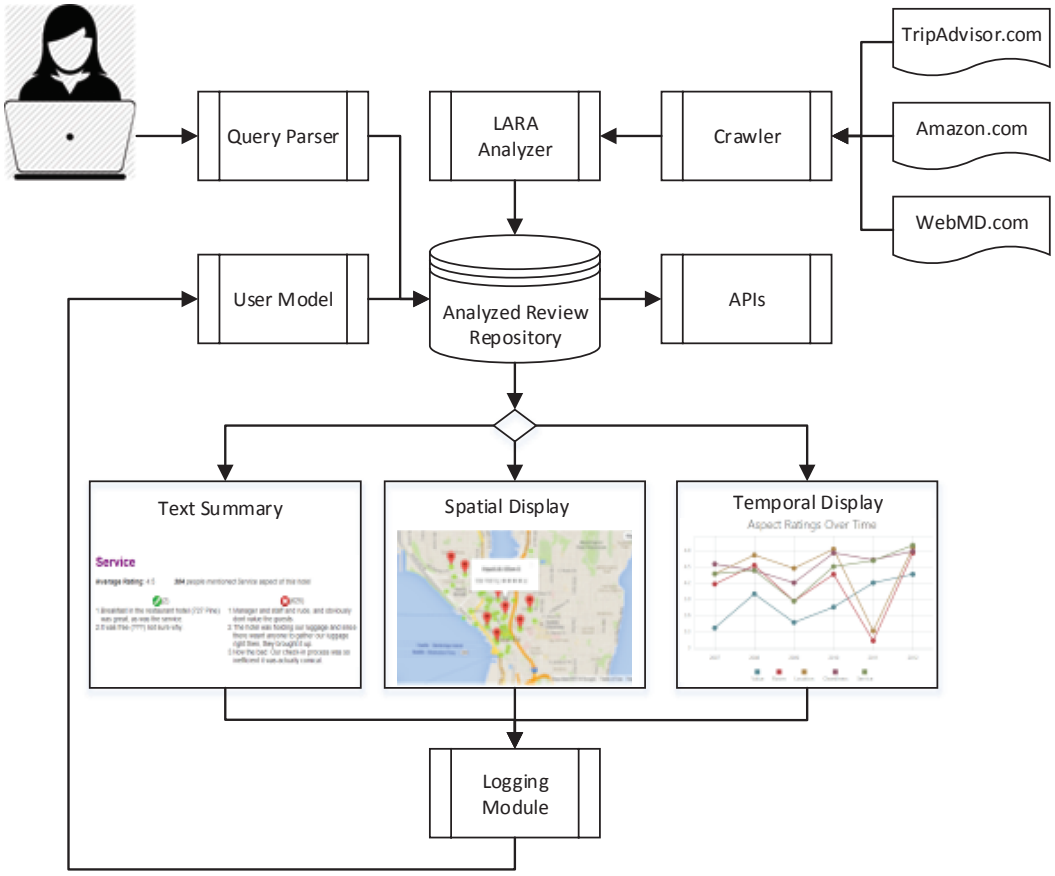


Figure 3.1: Overview of ReviewMiner system. There are four major components in this system: 1) review document crawler, LARA analyzer and analyzed review repository; 2) query parser; 3) multi-model user interface; 4) interaction behavior logging system and user modeling.

sentences in their content are discarded. Crawler is invoked periodically to capture the update of user reviews on these three different review sites.

• **Analyzer:** The LARA analyzer is implemented based on the two-step solution [118], i.e., keyword-based bootstrapping for aspect segmentation and latent rating regression for latent aspect rating and weight prediction. The choice of this two-step solution is mainly due to its computational efficiency and the fixed types of entities analyzed in the system. The unified solution of LARAM [119] is more appropriate for analyzing open domain opinionated documents, given it requires more complex computation to perform the joint modeling.

In particular, we manually determine the number of aspects in each category of entities, and select the most representative words as seed words for the bootstrapping-based aspect segmentation method. Due to the practical aspect coverage in individual reviews (not all the reviewers would

talk about every aspect of the entity in their reviews), it is infeasible for us to infer the latent aspect ratings and weights in every review document. As our solution, we aggregate reviews under each item and estimate the item-level aspect ratings and weights in the ReviewMiner system.

Although we only analyzed aspect opinions at item-level, we can still study the detailed aspect-level opinions within each review by applying the learned LRR model on the identified aspect segments. Such analysis helps us to visualize the detailed review content and extract opinionated sentences for summarizing the items of interest.

• **Analyzed Review Repository:** The analyzed reviews and items are stored in a back-end structured database. In order to ensure runtime efficiency of frontend execution, the aspect segments, ratings and weights for each item and review are pre-computed and stored in the database. In addition, in order to provide flexible search functions over all the analyzed items, keyword-based Lucene indices [49] are built over the item name and description fields for every category of entities. The details of these keyword indices will be discussed in the description of "Query Parser" component.

3.2.2 Query Parser

As described in the "Analyzed Review Repository" module, keyword-based Lucene indices are built based on the fields of item name and description in each category. Standard keyword search is supported by such indices: for example, in hotel reviews, users can type specific hotel name or location (or part of its name and location) as the query, such as "hotels in Champaign Illinois," and get the corresponding results of hotels and analyzed reviews. Specifically, we give the items matching the query term in its name field higher relevance score than those matching in description field to emphasize the importance of name field in search.

However, such simple keyword-based query scheme cannot support users to explicitly express their complex specifications over the search results in an ad-hoc manner. For example, if a user wants to find hotels in downtown Seattle with good service and price lower than $200/night, she has to first find all the hotels in Seattle area (e.g., by query "hotels in Seattle"), and manually filter out the irrelevant results. To support such complex search operations, in the query parser module of ReviewMiner system, similar techniques developed in LARA problem is leveraged to analyze the

semantic interpretation of the queries.

The basic workflow of the query parser is as follows:

1. Segment input free text query into phases using Stanford NLP parser [36].

2. Recognize if the user has specified the name or description of the item in the segmented phases using some predefined query templates. For example, in the query of "42" LCD Samsung TVs," the user is looking for *42 inches LCD* televisions manufactured by *Samsung*.

3. Classify the rest phases into different aspects by the learned aspect seed words: e.g., the phase "around downtown area" will be assigned to location aspect in hotel search.

4. Predict the sentiment polarity of each phase with respect to the identified aspect by the learned LRR model. Then normalize such polarities into three categories, i.e., "low," "medium" and "high." For example, the query phase "with very good cleanliness condition" would be interpret as expressing "high" requirement over "cleanliness" aspect, and it will favor hotels with high aspect rating of cleanliness aspect.

After these query parsing steps, a user's input free-text query is compiled into a semantically structured format, i.e., in the form of $\{aspect \rightarrow specification\}$, which facilitates the ReviewMiner system to retrieve more relevant items. In particular, the identified items' name and description from the query are used to retrieve the candidate items initially; and then each candidate is evaluated against the recognized aspect specification to estimate its relevance quality to the query. For example, if one query specifies "imperative" requirement on the value aspect but "optional" requirement on the service aspect, then the items with good value aspect ratings will be promoted over the items with low value aspect ratings but high service aspect ratings. The detailed ranking procedure will be discussed in the component of "interaction behavior logging system and user modeling."

Besides the natural language text query input, users can also explicitly specify their preferences over the identified aspects via a dropdown menu on the system interface (see in Figure 3.2). The users' explicitly input aspect preferences will be used to determine the final ranking of the retrieved items with other ranking factors in the component of "interaction behavior logging system and user modeling," which will be discussed later.

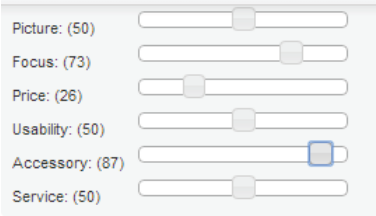


Figure 3.2: Dropdown menu for users to specify their preferences over the identified aspects. In this case, seven aspects are identified for the category of "digital cameras."

3.2.3 Multi-modal User Interface

As the output of the system, multi-modal result display is enabled by the detailed aspect-level analysis of review documents. In particular, ReviewMiner supports three types of user interaction interface, i.e., text-based opinion summary and comparison, spatial display of the retrieved results, and temporal display and comparison of the user's specified items. Users can easily access any of these three interfaces when interacting with the system.

• **Text-based Opinion Display:** The review text content under each retrieved item is segmented into aspects and highlighted with different colors for the users to quickly digest the opinions expressed in it (see in Figure 3.3). By specifying the preference weight over the aspects via the dropdown menu, ReviewMiner dynamically reorders the associated reviews by putting the reviews that mostly describe the aspects the user cares to the top.

Figure 3.3: Aspect-segmented review text content display. Sentences describing different aspects are highlighted with different colors.

In the text-based opinion summary (see in Figure 3.4), opinionated sentences are selected from the associated review text contents and ordered according to their sentiment polarities. As a result, comparative summarization is enabled by listing the top ranked opinionated sentences aspect by

aspect across different items. With such function support, the users can easily navigate through the selected item candidates and make informed comparisons.

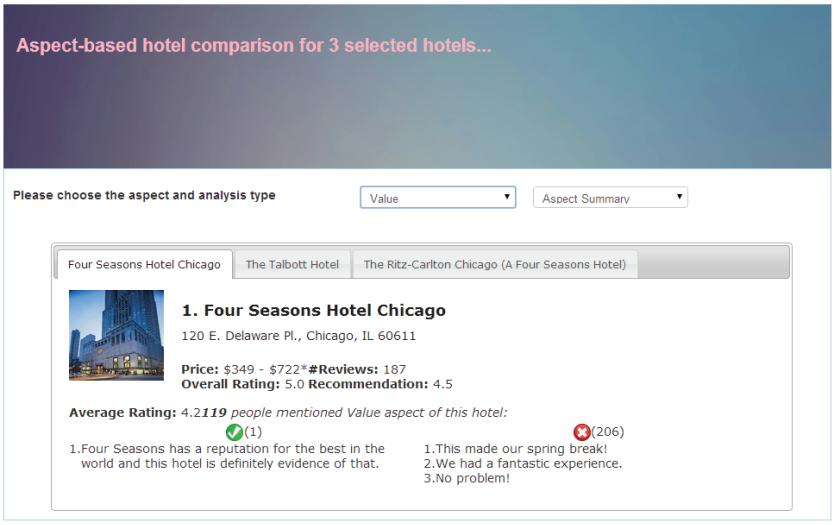


Figure 3.4: Comparative text-based summarization in ReviewMiner system. In this case, opinionated sentences are ranked according to their sentiment polarity under value aspect across different selected hotels.

• **Spatial Opinion Display:** For the hotel reviews, ReviewMiner visualizes the opinions in the spatial dimension, which helps the users to quickly find out where those "good" choices locate and explore comparative alternatives nearby (see in Figure 3.5). We want to emphasize that although spatial visualization of the retrieved items has been adopted in many practical systems, e.g., TripAdvisor (www.tripadvisor.com) and kayak (www.kayak.com), all of those systems simply list the location of items on a map. From a user's perspective, in order to assess the quality of the candidate items, she still has to go to the detailed review page of every item. Such interface design has made the spatial-based hotels comparison close to impossible.

In order to solve this deficiency, an extra layer of opinion-based heatmap [38] is introduced to represent the overall rating distribution of the retrieved hotels at the target location in ReviewMiner. On the heatmap, the areas with red color indicate the region with more high overall rating hotels, comparing to those with light green color. The markers indicate the top ranked hotels in the area with respect to the users' aspect preferences. With the support of spatial opinion display, the users no longer need to dig deep into the detailed reviews for comparison; instead, they can visually browse the area and thus it greatly simplifies their decision making process.

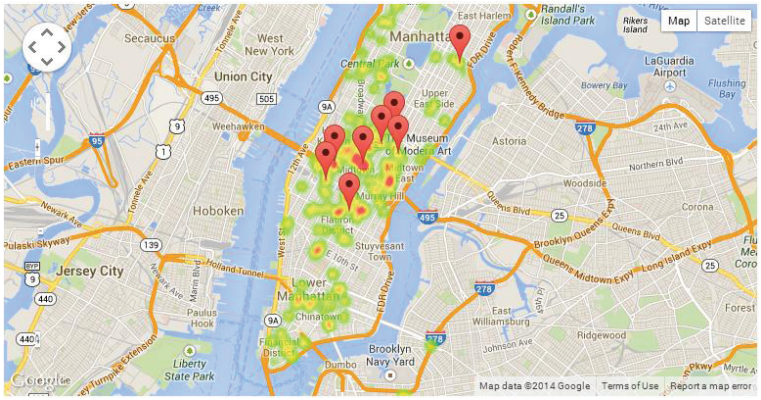


Figure 3.5: Spatial visualization of opinion distribution of hotels at Manhattan area in New York City. The markers indicates the top rated hotels in the region.

• **Temporal Opinion Display:**

Besides displaying the identified sentiment polarities of the selected items in spatial dimension, ReviewMiner also provides the visualization of opinions in the temporal dimension, i.e., displaying the inferred aspect ratings and weights over time for each selected item (see in Figure 3.6).

In the temporal opinion display, the reviews associated with each selected item are first grouped according to the date when they were published. We chose the unit of temporal dimension to be year, in order to balance the number of reviews in each unit and the total number of units for display. The inferred aspect ratings are aggregated from the reviews in each time unit. By such visualization of the aspect ratings and aspect mentions overtime, users can easily understand the dynamics of the reviewers' opinions on this particular item, and the trends of the sentiment polarities towards this item over time. In addition, this temporal opinion display also enables side-by-side comparison across multiple items in temporal dimension. This helps users to acquire more comprehensive and detailed opinion assessment over the selected items.

3.2.4 Interaction Behavior Logging and User Modeling

ReviewMiner supports user registration and login, in order to accurately keep track of individual users' information seeking behaviors in the system. All of a user's actions in the system, including typing a query, clicking on a returned result, browsing the analyzed review page, updating her aspect preference, clicking on a vote button under each review document, will be logged and analyzed for understanding the user's underlying information need. If a user logs into the system, all those

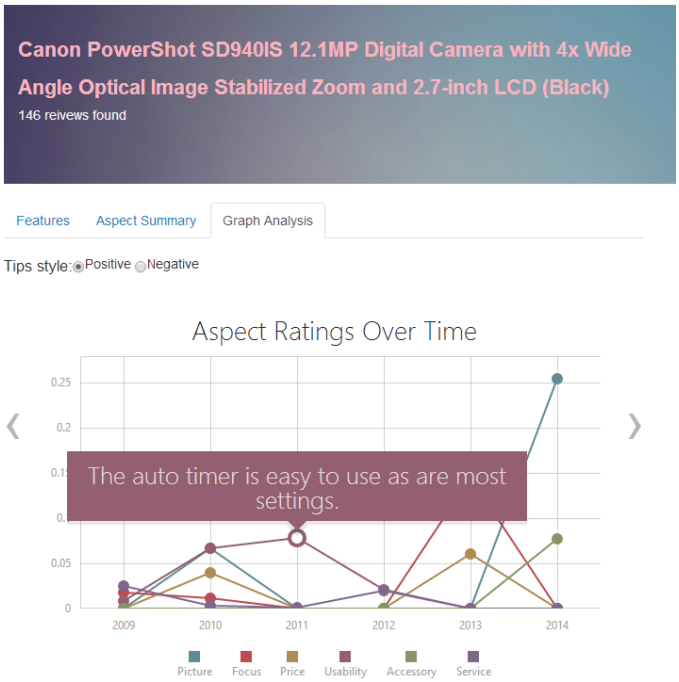


Figure 3.6: Temporal visualization of opinions over time. In this case, the aspect ratings of the selected Canon digital camera over five years are displayed.

actions will be logged under her unique and anonymized system ID; otherwise, those actions will be logged under the IP address of the user's browser.

The system maintains a unique profile of aspect preferences for each registered user under each category, and updates such profile when the user explicitly inputs her aspect preferences or clicks on a result in the search result page. The employed profile updating strategy follows the ranking model adaptation method developed in Section 5. In particular, ReviewMiner keeps track of the dwell time of users' browsing behaviors, and treats the clicked result page with dwell time longer than 30 seconds as positive feedback [50], and those skipped [70] or quick back clicks (dwell time less than 5 seconds) as negative feedback. In addition, each feedback document is represented by a vector of the inferred latent aspect ratings for each corresponding item. Such input is feeded into the personalized ranking model adaptation module to estimate the user's preference over the identified aspects in real time.

So far, in ReviewMiner, we have multiple criteria to rank the retrieved items: 1) a user's explicitly input aspect preference (w_u); 2) inferred aspect specification from the input query (w_p); 3) estimated personalized aspect specification from the user's interaction history (w_q). Those

different aspect preferences are linearly combined to get the final aspect weights for candidate ranking,

$$s_i \leftarrow (\lambda_u w_u + \lambda_p w_p + \lambda_u w_u)^{\mathsf{T}} r_i \tag{3.1}$$

where s_i is the final ranking score for item i, $\{\lambda_u, \lambda_p, \lambda_q\}$ represent the relative importance of those three types of aspect preferences, and r_i is inferred aspect rating vector for item i.

In the detailed system implementation, we intentionally bias towards the user's explicitly input aspect preference w_u by setting $\lambda_u = 0.7$, and gives less importance to inferred aspect preference from the user's input query and interaction history, i.e., setting $\lambda_p = 0.2$ and $\lambda_q = 0.1$.

3.3 Conclusions

In this chapter, based on the techniques developed in solving the problem of LARA, a practical system named ReviewMiner is built for multi-modal opinion analysis and decision support. ReviewMiner provides aspect-based opinion analysis, and it outputs results in textual, spatial and temporal dimensions in order to provide users with different perspectives to digest the opinions conveyed in the review text content. In addition, ReviewMiner automatically adapts to different users' aspect preferences from their interaction history in the system to perform personalized result ranking and recommendation. Currently, no commercial review websites, e.g., Amazon (www.amazon.com) or Newegg (www.newegg.com), can provide such in-depth analysis of user opinions and preferences when assisting users to make informed purchase decisions.

In addition, we want to emphasize that ReviewMiner not only provides easy access of massive opinion data to ordinary users, but also supports business analytic researchers to keep track of customer feedback and understand customer opinions of products and services. For example, ReviewMiner can recognize the inquired item's mostly commented aspects in the customer reviews, identify the corresponding relative emphasis the users have expressed over those aspects and track the temporal dynamics of user opinions and emphasis over those aspects. Such analysis can hardly be achieved in any other existing opinion mining or business analytic systems.

Part II

Modeling Interactive Behavior Data for System Optimization

Chapter 4

Long-term Search Task Extraction

In the second part of this thesis, we will shift our focus from analyzing user-generated opinionated text data for latent intent identification to modeling users' interactive behavior data logged in an information service system (e.g., their issued queries and clicked documents) for understanding their longitudinal information seeking behaviors. Three specific problems are addressed in this part: first, users' long-term search behaviors are automatically organized into "search tasks," i.e., ***long-term search task identification***; then the system's output is optimized according to the user's identified preference within the task, i.e., ***search personalization***; in the end, users' satisfaction with regard to their performed search task is analyzed for understanding the system's utility in assisting the users, i.e., ***search satisfaction prediction***.

In this chapter, we start from the problem of long-term search task extraction.

Search tasks, comprising a series of search queries serving for the same information need, have recently been recognized as an accurate atomic unit for modeling user search intent. Most prior research in this area has focused on short-term search tasks within a single time-based search session, and heavily depend on human annotations for supervised model learning. In this chapter, we target the identification of long-term, or *cross-session*, search tasks (transcending session boundaries) by investigating inter-query dependencies learned from users' searching behaviors. A semi-supervised clustering model is proposed based on the latent structural SVM framework, and a set of effective automatic annotation rules are proposed as weak supervision to release the burden of manual annotation. Our proposed long-term search task identification method enables a more comprehensive understanding of users' search behaviors via search logs and facilitates the development of dedicated search-engine support for long-term tasks.

4.1 Introduction

Search engine users' information needs span a broad spectrum [71, 88]: simple needs, such as homepage finding, can mostly be satisfied via a single query; but users may also issue a series of queries, collect, filter, and synthesize information from multiple sources to solve a complex task, e.g., planning a vacation. To comprehensively and accurately understand these needs from recorded actions in the user query logs, we must segment and associate chronologically-ordered queries into a semantically-coherent structure.

The primary mechanisms for segmenting the logged query streams in modern search engines are *session*-based, where short inactivity timeouts between user search actions are applied as a means of demarcating session boundaries [97, 100]. Recently, there has been significant research on identifying *tasks* within these sessions, e.g., Lucchese et al. [88] proposed the concept of a "*task-based session*": where a cluster of queries within the same session serves a particular common search intent. However, those methods rely on the accurate identification of the original session boundaries and the empirically-set timeout threshold may not be a valid criterion for identifying the semantic structure among queries: many tasks have been shown to span multiple search sessions [4, 71].

Motivating Example: Consider a real example of search tasks from a single user shown in Table 4.1, which is extracted from the logs of Bing.com. We manually annotated the in-session tasks in the last column of the table and segmented the sessions using 30-min inactivity threshold. We can observe that the user performed two tasks in the first search session on May 29, 2012, one for personal banking and another for shopping (for shoe-brand San Antonio Shoes). And on the second day, the user performed two individual search sessions, and each session consists of one single task, i.e., banking and shopping (at the online discount merchant 6pm.com) accordingly. However, humans can easily recognize that those four tasks annotated in three different sessions happen to be only two unique tasks: a shopping task including queries of "sas", "sas shoes", "6pm.com" and "coupon for 6pm," and a personal banking task including queries of "bank of america" and "credit union."

Prior work on identifying cross-session tasks has targeted *pairs* of queries, and made predictions about whether they share the same goal or represent the same task [71, 78]. Unfortunately, pairwise

Table 4.1: An example of cross-session search tasks.

Time	Query	SessionID	TaskID
05/29/2012 14:06:04	bank of america	1	1
05/29/2012 14:11:49	sas	1	2
05/29/2012 14:12:01	sas shoes	1	2
05/30/2012 10:19:34	credit union	2	3
05/30/2012 12:25:19	6pm.com	3	4
05/30/2012 12:49:21	coupon for 6pm	3	4

predictions alone cannot generate the partition of tasks, and post-processing is needed to obtain the final task partitions [80]. Besides, such pairwise predictions might not be consistent: e.g., predicting query i and j, query i and k to be in the same task, but query j and k are not. As a result, definite decisions have to be made in post-processing; but such decisions are isolated from the classifier training, and are therefore not guaranteed to be optimal.

To understand this limitation, taking the search tasks shown in Table 4.1 as an example. A lexicon-similarity-based classifier can easily recognize the query "6pm.com" and "coupon for 6pm," and "sas" and "sas shoes" belong the same search tasks, because of query overlap; but it can hardly associate "sas" with "6pm.com." Furthermore, the query "sas" is ambiguous: it has other interpretations such as the business analytic software SAS or special air service in British Army. Hence, even the features leveraging external knowledge bases [88] may be unable to assist. But when we consider the temporal juxtaposition of "sas shoes" and "sas," we can confidently infer that the "sas" here refers to "San Antonio Shoes"; and since we know that the queries "6pm.com" and "sas shoes" are both associated with shoe shopping, we can safely conclude that those four different queries are part of the same shopping task. From this example, we can conclude that the queries belonging to the same search task convey rich dependency relationships, which provide us with valuable information to analyze and exploit the search task structure. In contrast, traditional binary classification methods are only optimized for independent predictions and thus cannot explore such in-depth relationships among queries.

Moreover, existing methods for cross-session search task extraction heavily depend on the manual annotation of tasks [71, 78, 80], which is expensive to acquire at scale. Fortunately, we have the opportunity to leverage problem-specific knowledge to assist with model learning, where various

informative signals are available for us to identify such knowledge. For example, identical and reformulated queries, e.g., "sas" and "sas shoes" in Table 4.1, and queries with identical returned URLs should belong to the same search task with high confidence. Such knowledge can be summarized by a set of annotation rules, i.e., must-link and cannot-link [116], and applied at scale to reduce the burden of manual annotation. We refer to such knowledge as *weak supervision*, because it only provides pairwise supervision over a subset of queries; and the quality of such supervision might vary.

4.2 Related Work

Various methods have been proposed to segment and organize query logs into semantically coherent structures. The most commonly used unit, the search *session*, was often defined based on a timeout criterion, where different thresholds, ranging from 5 to 120 minutes, have been proposed [23, 61, 100]. In addition, Radlinski and Joachims [97] used a 30-minute timeout together with query similarity measures to define sequences of similar queries that combine to form so-called query chains.

Search tasks within the temporally-demarcated session boundaries have also been studied. Spink et al. [105] demonstrated that multi-tasking behavior, whereby multiple tasks are intertwined within the same time period, occurs frequently. Lucchese et al. [88] referred to such sessions as *task-based* sessions (or in-session tasks). Various methods, based on time splitting [6, 61], lexicon similarity [71, 88], and query reformulation patterns [61, 71], have been proposed to identify in-session tasks.

Recently, researchers have realized the necessity of going beyond the session timeout, and several methods have been proposed to tackle the problem by classifying whether two queries share the same search goal, i.e., same-task prediction. Jones et al. [71] claimed that no particular time-out threshold is necessary a valid constraint for identifying task boundaries. They found over 15% of search tasks are performed across time-out based session boundaries in their search log data set. To extract the cross-session tasks (which were defined as mission and goal), they built classifiers to identify task and sub-task boundaries, as well as pairs of queries belonging to the same task. Kotov et al. [78] and Agichtein et al. [4] studied the problem of cross-session task extraction via

binary same-task classification, and found different types of tasks demonstrate different life spans.

The major difference between our work and existing cross-session task extraction work is that instead of making a series of binary same-task predictions, we cast this problem as a structured learning problem, which explicitly models the dependency among queries in a search task. As we have discussed in Section 4.1, independent binary classification cannot capitalize on dependencies between pairs of predictions. In addition, existing classification-based methods heavily depend on manual annotations for model training. This will greatly limit their generalization capability when there is few or no task annotation available. In this work, we explored a variety of informative signals as weak supervision to release the burden of manual annotation and guide model learning.

4.3 Problem Definition

In this section, we formally define the problem of cross-session search task extraction.

Query log records the interaction behaviors from a set of different users, $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, in a search engine. It stores a sequence of queries $\mathcal{Q}_n = \{q_{n1}, q_{n2}, \ldots, q_{nM}\}$ from user u_n, together with the timestamp t_{ni} when the query q_{ni} is submitted and the corresponding list of returned URLs, $\mathcal{URL}_{ni} = \{url_{ni1}, url_{ni2}, \ldots, url_{niL}\}$. Each query q_{ni} is represented as the original string that users submitted to the search engine, and $\mathcal{Q}_n$ is ordered according to query timestamp t_{ni}. Each URL url_{nil} has two attributes: URL string and click timestamp c_{nil} (c_{nil}=0 if it was not clicked).

Definition (Session) Given user u_n's search history $\mathcal{Q}_n$ and a fixed time-out threshold τ_{cut}, a session $\mathcal{S}_{nt}$ is a set of consecutive queries from $\mathcal{Q}_n$, such that $\forall q_{ni} \in \mathcal{S}_{nt}, q_{nj} \in \mathcal{S}_{nt}, q_{nl} \notin \mathcal{S}_{nt}$, $|t_{ni} - t_{nj}| \leq \tau_{cut}$ and $|t_{ni} - t_{nl}| > \tau_{cut}$.

The definition of session implies that $\{\mathcal{S}_{nt}\}_{t=1}^{T}$ is a set of disjoint partitions of query sequence $\mathcal{Q}_n$, such that $\forall i \neq j$, $\mathcal{S}_{ni} \cap \mathcal{S}_{nj} = \emptyset$ and $\mathcal{Q}_n = \bigcup_i \mathcal{S}_{ni}$. A typical time-out threshold is set to be 30 minutes [78, 88, 97].

Definition (Search Task) Given user u_n's search history $\mathcal{Q}_n$, a search task $\mathcal{T}_{nk}$ is a maximum subset of queries in $\mathcal{Q}_n$, such that all the queries in $\mathcal{T}_{nk}$ correspond to a particular information need.

This definition of search task indicates $\{\mathcal{T}_{nk}\}_{k=1}^{K}$ is also a set of disjoint partitions of query sequence $\mathcal{Q}_n$: $\forall j \neq k$, $\mathcal{T}_{nj} \cap \mathcal{T}_{nk} = \emptyset$ and $\mathcal{Q}_n = \bigcup_k \mathcal{T}_{nk}$. Therefore, each $\mathcal{T}_{nk}$ is not confined to a particular session $\mathcal{S}_{nt}$; instead they can overlap, or one search task can contain multiple sessions. To emphasize such a difference, we will refer to our definition of search task as *Cross-session Search Task* as opposed to the previous definition of *In-session Search Task* [88, 105].

Based on the above notations and definitions, we define the problem of cross-session search task extraction as,

Definition (Cross-Session Search Task Extraction) Given user u_n's search query log $\mathcal{Q}_n$, partition the sequence into disjoint subsets $\{\mathcal{T}_{n1}, \mathcal{T}_{n2}, \ldots, \mathcal{T}_{nk}\}$, such that the partition is consistent with the user's underlying information need; when explicit task annotation is available, the extracted tasks should be consistent with the annotation.

In particular, such task partition can be uniquely determined by a mapping function $y(q_{ni}) \rightarrow \mathcal{T}_{nk}$ from query q_{ni} to its corresponding task partition $\mathcal{T}_{nk}$ for the query sequence $\mathcal{Q}_n$. In addition, we should note that the number of tasks, e.g., K, user u_n can take is not specified in our definition, and therefore the learning method should find the appropriate K for each given $\mathcal{Q}_n$ automatically.

4.4 Search Task Extraction with Latent Structured SVM

We model the cross-session search task extraction as a *supervised clustering problem (SCP)* [29, 48, 116], where given the clustering membership, we need to build up a model which captures the connection between queries.

4.4.1 Motivation: Best Link vs. All Links

A commonly used assumption in SCP is the *all-link* clustering structure [48, 65], where one needs to associate the queries belonging to the same task together, such that the in-cluster similarity defined by the summation of similarities over all pairs of instances within a cluster is maximized. However, this objective may not be the most appropriate for our problem: in a task consisting of m queries, many of the $O(m^2)$ pairs are not necessarily similar, or even quite different. Recall the example search tasks shown in Table 4.1, the query "sas" and "coupon for 6pm" are not directly

related under most of similarity metrics, e.g., edit distance or term overlap; putting them into the same task can only hurt the in-cluster similarity. As a result, any algorithm aims at maximizing the *all-link*-based in-cluster similarity can hardly discover this type of task.

A more reasonable way for clustering queries into tasks is to find the *strongest* link between a candidate query and queries in the target cluster, i.e., *bestlink* [65]. For example, after scanning through all the queries listed in Table 4.1, we can easily infer the relation between "sas" and "coupon for 6pm" based on the decision over the other two queries, "sas shoes" and "6pm.com," which have been recognized as being in the same shoe shopping task.

This motivates us to revise the objective of clustering queries: a query belonging to one particular search task *does not* need to be similar to all the other queries in this task (*all-link*), but there has to be *at least* one query, which is *strongly* associated with this query in that task (*bestlink*). Intuitively, this modeling assumption simulates how a human editor annotates search tasks in the query log: one might determine if two queries belong to the same task by reasoning transitively over strong connections between queries in the same task. And it explains the possible query reformulation chain among a user's issued queries.

4.4.2 Best Link as Latent Structure

Unfortunately, the bestlink structure is *hidden* in the query log, and it is even impossible for the human editors to explicitly annotate, since such structure might not be unique. Therefore, we adopt the structured learning method with latent variables, i.e., latent structural SVMs [25, 126], to realize the bestlink modeling assumption (rooted in Eq (1.2)), and utilize the hidden structure to explore the dependency among queries within the same task. We name our method as *bestlink SVM*.

To formalize the idea of bestlink SVM, we denote the hidden best-link structure as h. Before stating clearly the detailed definition of h, it helps to consider h as a graph whose edges connect the "most similar" queries. Given a query sequence $\mathcal{Q} = \{q_1, q_2, \ldots, q_M\}$[1], we define a feature vector for the task partition y specified by the hidden best-link structure h as $\Phi(\mathcal{Q}, y, h)$. And based on $\Phi(\mathcal{Q}, y, h)$, our bestlink SVM is a linear model parameterized by w, and predicts the task partition

[1]In the following discussion, when no ambiguity is invoked, we drop the index n for user u_n to simplify the notations.

at testing time by,

$$(\hat{y}, \hat{h}) = \underset{(y,h)\in\mathcal{Y}\times\mathcal{H}}{\arg\max}\ w^{\mathsf{T}}\Phi(\mathcal{Q}, y, h), \tag{4.1}$$

where $\mathcal{Y}$ and $\mathcal{H}$ represent the sets of possible structures of y and h respectively. $\hat{y}$ becomes the output for cross-session tasks and $\hat{h}$ is the inferred latent structure. As stated in Eq (4.1), the functional form $f(\cdot)$ in Eq (1.2) is realized by a linear function. In this paper, we refer to solving Eq (4.1) as the decoding problem.

The decoding problem of Eq (4.1) clearly distinguishes the proposed bestlink SVM model from the previous binary-classification-based methods. In bestlink SVM, we model the entire query sequence $\mathcal{Q}$ as a whole, and predict the task membership for all the queries simultaneously; while the previous two-step approaches cannot explore the interactions among queries in the same task, and isolated predictions are made on each pair of queries in those methods.

The definition of h needs to be carefully designed, otherwise the decoding problem (hence the training algorithm as well) can be intractable. We define $h(q_i, q_j) = 1$ if query q_i and q_j are directly connected in h; and otherwise, $h(q_i, q_j) = 0$. To model the first query of a new search task, i.e., the query that does not have a strong connection with any previous queries, we add a dummy query q_0 at the beginning of each user's query log. All the queries connecting to q_0 would be treated as the initial query of a new search task. Besides, we enforce that a query can only link to another query *in the past*, or formally,

$$\sum_{i=0}^{j-1} h(q_i, q_j) = 1, \forall j \geq 1$$

Taking the search tasks shown in Table 4.1 as an example, we illustrate the idea of bestlink structure in Figure 4.1. From the figure, we can clearly notice that the bestlink defines a hierarchical tree structure of "strong" connections among the queries: rooted in the dummy query q_0, each subtree of q_0 corresponds to one specific search task in a user's search history. For a new query, it can only belong to a previous search task or be the first query of a new task. Therefore, the temporal order provides us a helpful signal to explore the dependency between queries.

We require h to be consistent with y – that is, $h(q_i, q_j) = 1$ implies $y(q_i) = y(q_j)$; in other words, the task partition y is determined by the connected components in h. As a result, the dependency among the queries belonging to the same task is explicitly encoded by the latent bestlink structure

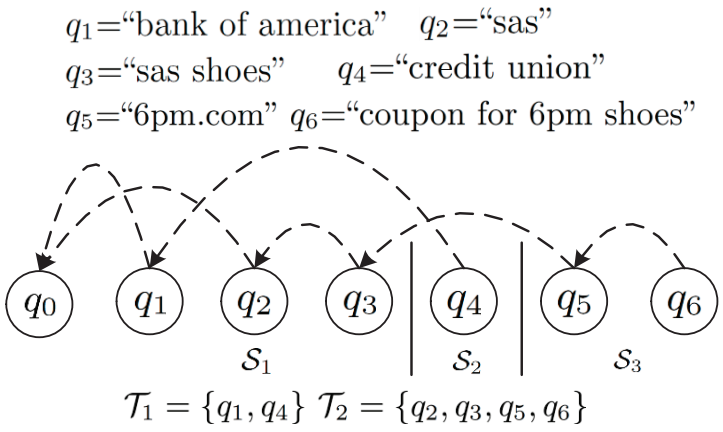


Figure 4.1: Illustration of hidden search task structure specified in bestlink SVM. $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ are the sessions segmented by the 30-minutes inactivity threshold, $\{\mathcal{T}_1, \mathcal{T}_2\}$ are the search tasks annotated by human editor. The dotted arrows indicate one possible hidden structure identified by bestlink SVM.

h: as shown in Figure 4.1, predicting "sas" and "sas shoes", "sas shoes" and "6pm.com" belonging to the same task would immediately lead to the conclusion that all these three queries belong to the same task, even though "sas" and "coupon for 6pm.com" are not directly connected to each other in h.

Accordingly, our feature vector for a particular task partition y is defined over the links in h as,

$$\Phi(\mathcal{Q}, y, h) = \sum_{i,j} h(q_i, q_j) \sum_{s=1}^{S} \phi_s(q_i, q_j), \tag{4.2}$$

where a set of symmetric pairwise features $\{\phi_s(\cdot,\cdot)\}_{s=0}^{S}$ is given to characterize the similarity between query q_i and q_j. In particular, to accommodate the dummy query q_0, we set $\phi_0(q_0, \cdot) = 1$ and $\forall s > 0, \phi_s(q_0, \cdot) = 0$.

Based on our feature vector design and the directed linkage structure of h, exact inference can be efficiently calculated for the decoding problem in Eq (4.1). Algorithm 1 described an incremental implementation to solve the exact inference problem, where we only need the queries appearing before the given query to determine its task membership. This makes bestlink SVM feasible to be deployed in the search engine query log system in an online fashion, since the newly arrived queries will not affect the method's prediction on previous queries.

Algorithm 1: Task Partition Prediction

Input: Query sequence $\mathcal{Q} = \{q_1, q_2, \ldots, q_M\}$, pairwise features $\{\phi_k(\cdot,\cdot)\}_{s=0}^{S}$ and linear weight w.
Output: Task partition $\hat{y}$.

```
//Step 1: Initialize the latent structure ĥ
ĥ(·,·) = 0;
//Step 2: Search for the best latent structure ĥ
for i = 1...M do
    j' = arg max_{0≤j<i} Σ_{s=1}^{S} w_s φ_s(q_i, q_j);
    ĥ(i, j') = 1;
end
//Step 3: Construct the best task partition ŷ:
t = 0;
for i = 1...M do
    j' = arg max_{0≤j<i} h(i, j);
    if j' = 0 then
        ŷ(i) = t;
        t = t + 1;
    end
    else
        ŷ(i) = ŷ(j');
    end
end
return ŷ
```

4.4.3 Pairwise Similarity Features

Our bestlink SVM requires a set of pairwise similarity features as input to characterize the connection between a pair of queries. In this work, we explored a variety of signals, from lexicon similarity to query semantic category similarity, to measure the similarity between a pair of queries.

Our proposed pairwise similarity features are list in Table 4.2, and categorized into three types: query-based, URL-based and session-based similarities. To analyze the semantic relationships between queries, we assign each URL to a topic distribution over 385 categories from the second level of "Open Directory Project" (ODP, dmoz.org) with a content-based classifier [98]. The inner product of the predicted topic distribution is used to measure the semantic similarity between queries. Besides, to make the features comparable across each other, we normalize them into the range of (0,1] accordingly, e.g., taking reciprocal of the absolute time difference (in seconds) between two queries.

Table 4.2: Pairwise Similarity Features.

Type	Feature	Description
Query -based	Q-COSINE	cosine similarity between the term sets of q_i and q_j
	Q-EDIT	norm edit dist between query strings of q_i and q_j
	Q-JAC	Jaccard coeff between the term sets of q_i and q_j
	Q-TIME	1.0/(absolute time difference in seconds between q_i and q_j)
	Q-DIST	(# of queries in between of q_i and q_j)/$\|Q_n\|$
	Q-URL-MATCH-SUM	$\sum_{url \in \mathcal{URL}_i} \big(c(q_j, url)\big) + \sum_{url \in \mathcal{URL}_j} \big(c(q_i, url)\big)$
	Q-URL-MATCH-MAX	$\max_{url \in \mathcal{URL}_i} \big(c(q_j, url)\big) + \max_{url \in \mathcal{URL}_j} \big(c(q_i, url)\big)$
	Q-CLICK-URL-MATCH-AVG	$\sum_{url \in \text{clicked } \mathcal{URL}_i} \big(c(q_j, url)\big) + \sum_{url \in \text{clicked } \mathcal{URL}_j} \big(c(q_i, url)\big)$
	Q-CLICK-URL-MATCH-MAX	$\max_{url \in \text{clicked } \mathcal{URL}_i} \big(c(q_j, url)\big) + \max_{url \in \text{clicked } \mathcal{URL}_j} \big(c(q_i, url)\big)$
URL -based	U-EDIT-DOMAIN-MIN	min norm edit dist between domain of $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-EDIT-ALL-MIN	min norm edit dist between $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-EDIT-ALL-CLICK-MIN	min norm edit dist between clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-EDIT-DOMAIN-AVG	avg norm edit dist between domain of $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-EDIT-ALL-AVG	avg norm edit dist between $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-EDIT-ALL-CLICK-AVG	avg norm edit dist between clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-JAC-ALL-CLICK	Jaccard coeff between clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-JAC-ALL	Jaccard coeff between $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-JAC-DOMAIN-CLICK	Jaccard coeff between domain of clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-JAC-DOMAIN	Jaccard coeff between domain of $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-SIM-CLICK-MAX	max ODP category similarity of clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-SIM-CLICK-AVG	avg ODP category similarity of clicked $\mathcal{URL}_i$ and clicked $\mathcal{URL}_j$
	U-SIM-MAX	max ODP category similarity of $\mathcal{URL}_i$ and $\mathcal{URL}_j$
	U-SIM-AVG	avg ODP category similarity of $\mathcal{URL}_i$ and $\mathcal{URL}_j$
Session -based	S-SAME	if q_i and q_j are in the same session
	S-FIRST	if both q_i and q_j are the first query of session
	S-DIST	# queries in between of q_i and q_j

Note: 1) ***norm edit dist*** is the edit distance between string s and t divided by the maximum length of s and t;
2) $c(q, url)$ is a function counting the number of query terms in q contained in url;
3) ***clicked*** $\mathcal{URL}$ is a subset of URLs, whose click timestamp $c_{il} > 0$.

4.4.4 Solving the bestlink SVM

For a given set of query logs with annotated tasks, $\{(\mathcal{Q}_n, y_n)\}_{n=1}^N$, we need to retrieve the optimal weight setting w for the proposed bestlink SVM. Empirically, the optimal weight w should minimize the error between the predicted task partition $\hat{y}_n$ and ground-truth y_n. In addition, w should also be optimized for good generalization capability, e.g., maximize the margin between ground-truth partition and wrong partitions [114]. This naturally gives rise to the following optimization problem within the latent structural SVMs framework [25, 126]:

$$\min_{w,\xi} \ \frac{1}{2}||w||^2 + C\sum_{n=1}^{N} \xi_n^2 \tag{4.3}$$

$$s.t. \forall n, \max_{h \in \mathcal{H}} w^{\mathsf{T}} \Phi(\mathcal{Q}_n, y_n, h) \geq \max_{(\hat{y},\hat{h}) \in \mathcal{Y} \times \mathcal{H}} [w^{\mathsf{T}} \Phi(\mathcal{Q}_n, \hat{y}, \hat{h}) + \Delta(y_n, \hat{y}, \hat{h})] - \xi_n$$

where $\Delta(y_n, \hat{y}, \hat{h})$ characterizes the distance between the ground-truth partition y_n and predicted partition $\hat{y}$ specified by the latent structure $\hat{h}$, $\{\xi_n\}_{n=1}^N$ is a set of slack variables to allow errors in the training set, and C controls the trade-off between empirical loss and model complexity.

Because the ground-truth bestlink structure h_n^* for $\mathcal{Q}_n$ is unobservable in the training data, we cannot measure the distance between (y_n, h_n^*) and $(\hat{y}, \hat{h})$. As a result, we define the margin between the ground-truth task partition y_n and predicted task partition $\hat{y}$ based on the inferred latent structure $\hat{h}$ as,

$$\Delta(y_n, \hat{y}, \hat{h}) = |\mathcal{Q}_n| - |\mathcal{T}_n| - \sum_{i,j} h(i,j)\sigma(y_n, (i,j)) \tag{4.4}$$

where $|\mathcal{Q}_n|$ is the number of queries in $\mathcal{Q}_n$, $|\mathcal{T}_n|$ is the number of annotated tasks in $\mathcal{Q}_n$, and $\sigma(y, (i,j)) = 1$ if $y(i) = y(j)$, otherwise $\sigma(y, (i,j)) = -1$. It is easy to verify that $\Delta(y_n, \hat{y}, \hat{h})$ is non-negative, and equals to zero if and only if the task partition $\hat{y}$ is the same as y_n.

Since we are minimizing the square hinge loss over the training set, the optimization problem introduced in Eq (4.3) can be efficiently solved by the iterative algorithm proposed in [25]: the optimization procedure minimizes Eq (4.3) by constructing a sequence of convex problems in each iteration, and each iteration guarantees to decrease the objective function. In the employed optimization algorithm, two types of additional inference are required: loss-augmented inference, i.e., $\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}}[w^\mathsf{T}\Phi(\mathcal{Q}_n, \hat{y}, \hat{h}) + \Delta(y_n, \hat{y}, \hat{h})]$; and latent variable completion inference, i.e., $\max_{h\in\mathcal{H}} w^\mathsf{T}\Phi(\mathcal{Q}_n, y_n, h)$. Since the calculation of $\Delta(y_n, \hat{y}, \hat{h})$ can be decomposed onto the edges in h, loss-augmented inference can be directly solved via Algorithm 1 by adding an additional cost $\sigma(y_n, (i,j))$ into Step 2 when finding the best link for query q_i. And the latent variable completion inference can also be achieved via Algorithm 1 by restricting Step 2 to only search in the queries with the same task label as q_i. Both inference algorithms are exact, which renders us a more precise optimization result for Eq (4.3).

4.5 Improving the Model with Weak Supervision Signals

The bestlink SVM proposed in Section 4.4.2 is a supervised clustering algorithm that requires full annotation of search tasks in the query log. As we have discussed in Section 4.1, various types

of signals, which can be automatically derived from the query logs, are helpful for identifying the search tasks. In this section, we discuss how to make use of large quantities of unlabeled data with weak supervision signals in the proposed bestlink SVM.

We explore weak supervision signals for the cross-session search task extraction problem from different perspectives, and formalize them in terms of "must-link" and "cannot-link" [116]. Query matching, e.g., identical or reformulated queries, is a strong indication that two queries belong to the same task. Besides, the returned URLs for the given query are also an important source for determining the task membership: because modern search engines have sophisticated query pre-processing procedures, e.g., spelling correction [31] and query rewriting [72], when it decides to return identical URLs for two different queries, it is a strong signal that the two queries are related. Table 4.3 lists four types of must-link and one type of cannot-link that we have defined in this work. When there is conflict between the automatically generated must-links and cannot-links, e.g., nontransitive, we will drop the cannot-links to make the annotations consistent.

Table 4.3: Partial Annotation Rules.

Type	Description
Must-link ($\tilde{y}(i) = \tilde{y}(j)$)	$q_i = q_j$ $q_i \subset q_j$ or $q_j \subset q_i$ $\mathcal{URL}_i = \mathcal{URL}_j$ clicked $\mathcal{URL}_i$=clicked $\mathcal{URL}_j$
Cannot-link ($\tilde{y}(i) \neq \tilde{y}(j)$)	$q_i \neq q_j$ AND $\mathcal{URL}_i \cap \mathcal{URL}_j = \emptyset$

Though one may treat such signals as features and manually tune the weights to stress their importance, we want emphasize that this approach is sub-optimal for the following two reasons: 1) features are independent in linear models, the knowledge about one feature cannot help the model learn for other features; instead, if we treat such information as supervision, all the features can be adjusted accordingly; 2) it is difficult to manually set the appropriate weights for all the features; while optimizing the objective function defined on both weak supervision and manual annotations would estimate the weights in a more systematic way.

Note that when we apply the proposed must-link and cannot-link to the unlabeled user query logs, we can only get partial annotations on those queries given that the coverage of the weak

supervision is not perfect. We denote the partial annotation as $\tilde{y}$, and to accommodate such partial annotations in bestlink SVM, we modify the margin defined in Eq (4.4) as follows,

$$\tilde{\Delta}(\tilde{y}_n, \hat{y}, \hat{h}) = |\mathcal{Q}_n| - |\mathcal{C}_n| - \sum_{i,j} h(i,j)\tilde{\sigma}(y,(i,j)) \tag{4.5}$$

where $|\mathcal{C}_n|$ is the number of connected components (including singletons) defined by must-links in $\mathcal{Q}_n$, and $\tilde{\sigma}(y,(i,j)) = \lambda^+$ if $\tilde{y}(i) = \tilde{y}(j)$, $\tilde{\sigma}(y,(i,j)) = -\lambda^-$ if $\tilde{y}(i) \neq \tilde{y}(j)$, otherwise $\tilde{\sigma}(y,(i,j)) = 0$ when there is no annotation between query i and j. This modificatoin makes our bestlink SVM a semi-supervised clustering algorithm.

We can easily verify that $\tilde{\Delta}(\tilde{y}_n, \hat{y}, \hat{h})$ is a more general definition of the distance between the given (or partial) task partition and the predicted task partition, in which we count how many edges in $\hat{h}$ are consistent with given annotation (or must-links) in $\tilde{y}$, and how many of them are conflicting with the annotation (or must-/cannot-links). In addition, to distinguish the creditability of the rule-based must-link and cannot-link, we assign them different cost factors, i.e., $\lambda^+ > 0$ and $\lambda^- > 0$, which can be set according to model's performance on a manually annotated held-out set.

4.6 Experimental Results

In order to evaluate the proposed method, we performed a series of experiments on a large scale search dataset sampled from the query logs from Bing.com. First, we compared the performance of the proposed bestlink SVM to several state-of-the-art methods for the cross-session search task extraction problem. Then, a set of experiments were conducted to study the effectiveness of using weakly supervised data, which is automatically derived from user query logs, for identifying cross-session search tasks.

4.6.1 Query Log Dataset

We extracted five days' search logs from Bing.com, from May 27 2012 to May 31 2012, for our experiments. During this period, a subset of users are randomly selected and all their search activities are collected, including the anonymized user ID, query string, timestamp, returned URL sets and the corresponding user clicks. The 30-minutes inactivity threshold is used to segment

queries into sessions as pre-processing [80, 122]. Since the focus is identifying cross-session search tasks, we further filtered out the users who submitted less than two queries or had less than two sessions during this period. As a result, we collected 7,628 users with 114,723 queries. The basic statistics of this data set are shown in Table 4.4.

Table 4.4: Statistics of evaluation query log data set.

# User	# Session	# Query
7628	37547	114723
Query/User	Session/User	Query/Session
15.1±17.2	4.9±3.5	3.1±1.2

In order to evaluate the performance of the proposed method in identifying cross-session search tasks, three editors were recruited to annotate the search tasks. Editors were instructed to group the queries into tasks according to their understanding of users' information needs, and they were encouraged to use external resources, e.g., search for the logged queries and browse the clicked URLs, to infer the relation between queries. The same set of 200 users' query logs are distributed in each editor's annotation assignment to measure their annotation agreement. Cohen's kappa on pairwise annotation of queries showed high inter-annotator agreement, 0.68, 0.73 and 0.77, for the three pairs of editors. After aggregating the three editors' annotations, we got a collection of 10,327 tasks annotated out of 1,436 users' search logs, and the basic statistics of this data set are shown in Table 4.5.

Table 4.5: Statistics of annotated search tasks.

Single-query Task	Multi-query Task
8044	2283
Multi-session Task	Interleaving Task
1307	709
Task/User	Query/Task*
7.2±10.1	6.6±8.2
Session/Task*	Task duration (mins)*
2.8±2.6	491.1±933.5

∗count only in multi-query tasks

As shown in Table 4.5, in average a user takes 7.2 different tasks during this period, 22.1% of which contain multiple queries, more than 57.2% multi-query tasks span across session boundaries, and 31.1% of them are interleaving. This shows the need of going beyond session boundaries to

extract the long-term search tasks. In particular, when we look into those multi-query tasks, they span 6.6 queries, 2.8 sessions and more than 8 hours in average. This indicates that cross-session task extraction is not a trivial problem, and one needs to leverage rich information for identifying the structure of a cross-session search task.

4.6.2 Search Task Extraction

Algorithms for Comparison: Several methods have been proposed to identify cross-session search tasks based on the idea of same-task classification [71, 78]. However, those methods only provide predictions over pair of queries, and post-processing is needed to obtain the final task partitions. In our experiment, we adapted two best performing clustering methods from Lucchese et al.'s work [88], i.e., QC_wcc and QC_htc, as the post-processing procedure for the baselines. QC_wcc performs clustering by dropping "weak edges" among queries and extracting the connected components as tasks. QC_htc assumes a cluster of queries can be well represented by only the chronologically first and last query in the cluster, and therefore only the similarity among the first and last queries of two clusters is considered in agglomerative clustering. We trained a linear SVM model to classify if two queries are in the same task, treated the predicted positive query pairs as "strong edges," and applied QC_wcc and QC_htc to obtain the final task partition. In this setting, QC_wcc works exactly the same as Liao et al. proposed in [80].

Since our proposed bestlink-SVM can be viewed as a supervised clustering method [29, 48, 116], we also included two state-of-the-art supervised clustering methods, i.e., "adaptive-clustering" [29] and "cluster-svm" [48] as baselines. Adaptive clustering (AdaptClu) performs single-link agglomerative clustering based on binary classification results. To avoid overfitting, it selects a representative subset of all the candidate pairs based on their similarities when training the binary classifier. In our experiment, we used the summation of all the pairwise similarities as defined in Table 4.2 between two queries (with negative signs for edit-distance-based similarities) for selecting the representative subset of queries. cluster-svm performs correlation clustering by learning a structural SVM model, which simultaneously optimizes the pairwise accuracy and in-cluster similarity defined by *all-link* in one cluster.

To make a fair comparison, all the methods are trained on the same set of pairwise features

defined in Table 4.2.

Performance metrics: A commonly used evaluation metric for search task extraction is pairwise precision/recall [71, 78] defined as,

$$p_{\text{pair}} = \frac{\sum_{i<j} \delta\big(y(q_i), y(q_j)\big)\delta\big(\hat{y}(q_i), \hat{y}(q_j)\big)}{\sum_{i<j} \delta\big(\hat{y}(q_i), \hat{y}(q_j)\big)} \tag{4.6}$$

$$r_{\text{pair}} = \frac{\sum_{i<j} \delta\big(y(q_i), y(q_j)\big)\delta\big(\hat{y}(q_i), \hat{y}(q_j)\big)}{\sum_{i<j} \delta\big(y(q_i), y(q_j)\big)} \tag{4.7}$$

where p_{pair} evaluates how many pairs of queries predicted in the same task, i.e., $\delta\big(\hat{y}(q_i), \hat{y}(q_j)\big) = 1$, are actually annotated as in the same task, i.e., $\delta\big(y(q_i), y(q_j)\big) = 1$; and r_{pair} evaluates how many pairs annotated as in the same task are recovered by the algorithm.

However, it is worth noting that these metrics cannot directly measure the clustering quality, and have some limitations: 1) they ignore singleton tasks, since no pairs can be formed from such tasks; 2) they intrinsically favor methods producing fewer tasks [89]. Inspired by the metrics used in the problem of co-reference resolution in natural language processing, we employed the Constrained Entity-Alignment F-Measure ($f1_{\text{CEAF}}$) as proposed in [89] to evaluate the clustering quality. CEAF defines the clustering precision and recall based on the best alignment between the predicted cluster and ground-truth cluster, where the alignment can be measured by any similarity function defined on two sets:

$$p_{\text{CEAF}} = \frac{\sum_i \pi(\hat{\mathcal{T}}_i, g(\hat{\mathcal{T}}_i))}{\sum_i \pi(\hat{\mathcal{T}}_i, \hat{\mathcal{T}}_i)} \tag{4.8}$$

$$r_{\text{CEAF}} = \frac{\sum_i \pi(\hat{\mathcal{T}}_i, g(\hat{\mathcal{T}}_i))}{\sum_j \pi(\mathcal{T}_j, \mathcal{T}_j)} \tag{4.9}$$

where $\pi(A, B)$ is a similarity measure between set A and B, which is chosen to be Jaccard coefficient in our evaluation; and $g(\cdot)$ is the optimal mapping between the predicted task partition $\mathcal{T}$ and ground-truth task partition $\hat{\mathcal{T}}$. Then, $f1_{\text{CEAF}}$ can be calculated as,

$$f1_{\text{CEAF}} = \frac{2 \times p_{\text{CEAF}} \times r_{\text{CEAF}}}{p_{\text{CEAF}} + r_{\text{CEAF}}} \tag{4.10}$$

Furthermore, we also included Normalized Mutual Information (NMI), a standard metric for

evaluating the clustering quality, as one of our evaluation metrics. The detailed definition of NMI can be found in [22]. Basically, the higher the NMI score the better clustering performance an automatic system achieves: NMI= 1 if the prediction is identical to the ground-truth; and NMI= 0 if the prediction is independent from the ground-truth.

Evaluation of search task extraction methods: We randomly split the annotated user query logs into a training set with 712 annotated users, and a testing set with the rest 725 annotated users. The parameters in each model, e.g., C in SVM-based models, are tuned by 5-fold cross-validation on the training set (splitting the annotated users into different folds).

We trained all the methods on the manually annotated training set, and compared their task extraction performance in Table 4.6, where we averaged the performance under each metric over all the testing cases. A paired two-sample *t-test* is performed to validate the significance of improvement from the best performing method against the runner-up method under each metric.

Table 4.6: Search task extraction performance.

	p_{pair}	r_{pair}	$f1_{\text{CEAF}}$	NMI
Q_wcc	0.8653	**0.9833***	0.4826	0.4058
Q_htc	0.9213	0.8607	0.5461	0.5636
AdaptClu	0.9059	0.9046	0.5583	0.5466
cluster-svm	0.9232	0.7908	0.5363	0.5602
bestlink SVM	**0.9330***	0.9273	**0.5895***	**0.6046***
$\text{AdaptClu}_{\text{all}}$	0.8681	0.4611	0.2880	0.3236
Rule-based	0.8954	0.5570	-	-

* indicates *p-value*<0.01

In Table 4.6 we first observed that cluster-svm, which is also a structural learning method, performed much worse than bestlink SVM, especially on r_{pair}. The reason is that cluster-svm optimizes the in-cluster similarity defined by *all-link* among the queries; while in bestlink SVM, the in-cluster similarity is only defined on the *bestlink* among the queries, or more precisely, the edges exist in h (as shown in Eq (4.2)). To validate this hypothesis, we implemented an additional baseline of *all-link*-based adaptive clustering ($\text{AdaptClu}_{\text{all}}$). In $\text{AdaptClu}_{\text{all}}$, we changed the original single-link agglomerative clustering to all-link agglomerative clustering, where the in-cluster similarity is defined the same as in cluster-svm. As observed in the result, $\text{AdaptClu}_{\text{all}}$ performed significantly worse than AdaptClu, especially on r_{pair}. This result validates our basic modeling assumption

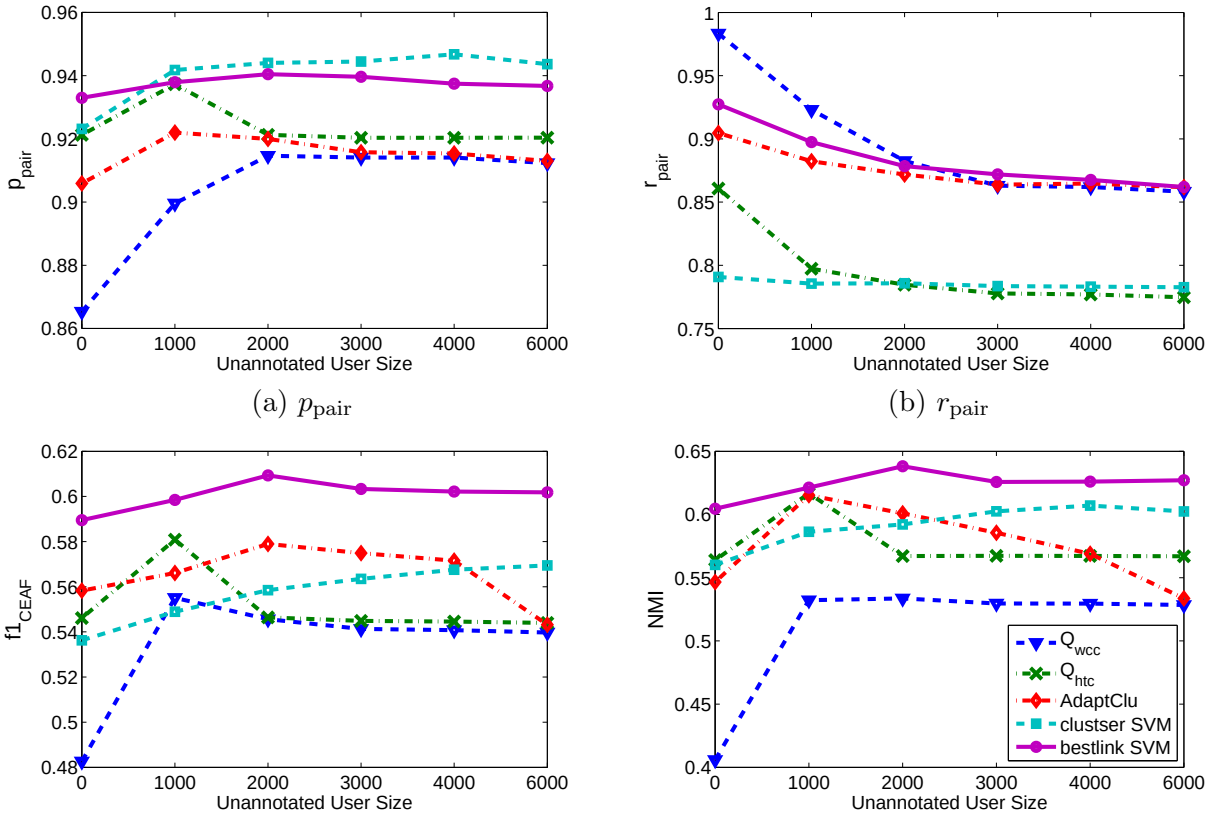
in the proposed bestlink SVM, i.e., a query belonging to a particular task should have a strong connection with at least another one query rather than all the other queries in the same task.

Besides, as discussed in Section 4.1, due to the lack of interaction between the binary classifier training and query clustering in post-processing, the two-step approaches are likely to give sub-optimal task extraction performance. Q_wcc and Q_htc are based on the same binary classifier's output, but their performance differs because of distinct strategies used in post-processing. Q_wcc tends to connect all the queries together, and results in a high r_{pair}, but poor performance on other metrics. On the other hand, because Q_htc only compares the first and last queries between two different clusters, it gives a relatively lower r_{pair}, but better clustering performance due to a better p_{pair}, as compared to Q_wcc.

In Section 4.5, we proposed a method for automatically generating weak supervision from search logs in the form of must-link and cannot-link. In Table 4.6, we also evaluated the quality of such weak supervision. Since the rule-based supervision merely provides pairwise annotations, we only evaluated its p_{pair} and r_{pair}. In general, p_{pair} of these auto-generated annotations is reasonably good, while r_{pair} is relatively poor. This result is expected: the method described in Table 4.3 uses strong signals for annotation; but the coverage of such signals is limited, since some relations between two distinct queries can only be inferred by reasoning over the whole query sequence by human judges.

Effectiveness of weakly supervised data To investigate the effectiveness of the weak supervision in helping to train the supervised model, we gradually added the weakly supervised data into our training set. We first obtained the pairwise annotations, as defined in Table 4.3, for those users who have not been manually annotated; then we gradually added such partially labeled user query logs into the manually-annotated training set. For binary-classification-based baselines, i.e., Q_wcc, Q_htc and AdaptClu, the newly added pairwise annotations are used as regular training supervision; for cluster-svm, the loss function is modified to adopt the partial annotations (similar as Eq (4.5)). The experimental results are summarized in Figure 4.2.

From Figure 4.2 we can study the utility of weakly supervised data on cross-session task extraction. As shown in Figure 4.2 (c) and (d), the supervised learning methods benefit from a medium volume of weakly supervised data; but when the volume reaches certain limit, the performance



(a) p_{pair} (b) r_{pair}

(c) $f1_{\text{CEAF}}$ (d) NMI

Figure 4.2: Task extraction performance with increasing volume of weakly supervised data.

stops improving, and even degrades. Figure 4.2 (a) and (b) help to explain why this happens: all methods' r_{pair} performance drops when adding the weakly supervised data for training, but their p_{pair} performance improves. With the improved p_{pair}, all methods' clustering performance, in terms of $f1_{\text{CEAF}}$ and NMI, gets improved. As shown in Table 4.6, the weakly supervised data has high precision but low recall, adding more such training signals would bias the models toward recognizing the pairs similar to those high-precision must-links. When the volume of weakly supervised data passes a limit, it will overwhelm the signals from human annotations; and therefore hinders further improvement. Figure 4.2 also shows that, compared to the two-step methods, the structural learning based method, i.e., cluster-svm and bestlink SVM, can utilize more weakly supervised data before the performance saturates. The reason is that structured learning method directly optimizes (or approximates) the clustering metrics during training. The two-step methods perform classification and clustering independently, and there is inconsistency between training objective and evaluation in these two-step methods. As a result, errors in the learned binary classifier cannot be recovered easily in the clustering stage in those methods.

Weakly supervised search task extraction We are also interested in investigating how well the models could perform when there is only weakly supervised data. In other words, we want to test if the learning methods' task extraction capability can go beyond the simple annotation rules. In this experiment, we only trained the models on the 6,192 unannotated users with weak supervision, and tested them on the same manually annotated testing set as before. In order to analyze how well the methods generalize from the weakly supervised data, we included a naive baseline Rule-Q_wcc: we adopted Q_wcc by treating the queries connected by the must-links as a task.

Table 4.7: Task extraction performance when trained only on the weakly supervised data.

	p_{pair}	r_{pair}	$f1_{\text{CEAF}}$	NMI
Rule-Q_wcc	0.9084	0.5136	0.5492	0.5602
Q_wcc	0.9123	0.8582	0.5397	0.5285
Q_htc	0.9204	0.7747	0.5440	0.5669
AdaptClu	0.9131	**0.8613***	0.5426	0.5325
cluster-svm	0.9155	0.7565	0.5197	0.4805
bestlinkSVM	**0.9334***	0.8161	**0.5676***	**0.5893***

* indicates $p\text{-}value<0.01$

As shown in Table 4.7, all the methods improved p_{pair} and r_{pair} against Rule-Q_wcc, and especially for r_{pair}. However, not all of them can improve the clustering quality metric: besides bestlink SVM, only Q_htc improves NMI metric. We looked into the detailed output of those methods and found that: Rule-Q_wcc generated many singleton tasks because of the low coverage of must-links; the baseline models merged some of the small clusters into larger ones, but they still created too many smaller clusters than ground-truth. bestlink SVM correctly merged the small clusters, making the number of predicted tasks closest to the ground-truth, and therefore it achieved better clustering performance.

We wanted to further investigate how many "complex tasks," which are not covered by the must-links defined in Table 4.3, can be extracted by learning from the weak supervision. Specifically, we define the complex task as: $\mathcal{T}^*_{\text{strict}}$, in which no must-link can be applied on any pair of queries in it (strict criterion); or $\mathcal{T}^*_{\text{loose}}$, there exists at least one pair of queries cannot be connected via must-links in it (loose criterion). Based on this notation, we define the coverage of complex task as the proportion of complex tasks which can be perfectly recovered by the automatic methods,

$$c_{\text{loose}} = \frac{\sum_{\mathcal{T}_i \in \hat{\mathcal{T}}} \sum_{\mathcal{T}_j \in \mathcal{T}^*_{\text{loose}}} \delta(\mathcal{T}_i, \mathcal{T}_j)}{|\mathcal{T}^*_{\text{loose}}|} \tag{4.11}$$

$$c_{\text{strict}} = \frac{\sum_{\mathcal{T}_i \in \hat{\mathcal{T}}} \sum_{\mathcal{T}_j \in \mathcal{T}^*_{\text{strict}}} \delta(\mathcal{T}_i, \mathcal{T}_j)}{|\mathcal{T}^*_{\text{strict}}|} \tag{4.12}$$

where $\delta(\mathcal{X}, \mathcal{Y}) = 1$ when the set $\mathcal{X}$ and $\mathcal{Y}$ are the same, and otherwise $\delta(\mathcal{X}, \mathcal{Y}) = 0$.

In this experiment, we used all the 1436 annotated users as testing set, where we collected 357 strict complex tasks and 1540 loose complex tasks out of the total 2283 multi-query tasks. All the models are trained on the rest 6192 unannotated users with weak supervision, and the experimental results are list in Table 4.8, where we used sign-test for validating the improvement over the baselines.

We should note that all those complex tasks cannot be identified by the straight-forward Rule-Q_wcc baseline, so that the newly defined task coverage metric measures how well the learning methods can generalize from the weak supervision. From the results we can notice that bestlink SVM, which achieved the best performance against all the other baselines, can successfully recover about 30% of complex tasks by leveraging the knowledge from weak supervision, which validates

the effectiveness of using such signals as supervision.

Table 4.8: Coverage of complex tasks when trained only on the weakly supervised data.

	c_{loose}	c_{strict}
Q_wcc	0.2914	0.2745
Q_htc	0.2617	0.2761
$\text{AdaptClu}_{\text{single}}$	0.2837	0.2717
cluster-svm	0.2883	0.2997
bestlinkSVM	**0.3207***	**0.3501***

* indicates $p\text{-}value<0.01$

4.6.3 Analysis of Identified Tasks

As we have discussed in Section 4.4.2, the latent structure h defined in bestlink SVM is a tree structure formed by strong connections between queries, where each subtree of the dummy query q_0 corresponds to a search task. In Figure 4.3, we illustrated the latent task structure inferred by our bestlink SVM from two different users' query logs.

Comparing to the flat clustering structure given by the traditional search task extraction methods [78, 88], the hierarchical structure inferred by bestlink SVM provides us with more in-depth details to understand users' search behaviors and their information needs. For example, in Figure 4.3 we can clearly notice that the identified task structure for User2 is more complex than that for User1: User1 attempted three consecutive tasks on May 29; while User2's two major search tasks, i.e., checking daily news and looking for solutions of her health issue, spanned from May 29 to May 31, and were performed in an interleaved manner. And the subtrees in an identified search task represent finer grained subtasks. For instance, as shown in Figure 4.3, in User2's second identified task of "plantar fasciitis symptoms," there are two subtasks, one starts with "plantar fasciitis pictures" and another starts with "chagas disease."

At the beginning of Section 4.6, we listed a brief overview of basic properties of search tasks based on a limited number of human annotations. Now we can get a more comprehensive understanding of user's search behaviors based on the automatically extracted search tasks in our whole query log data set. We listed a set of statistics in Table 4.9, where we applied a proprietary multi-class classifier to categorize a query into 80 categories, e.g., navigational, commerce, celebrity and etc., in order to annotate the search intent of queries.

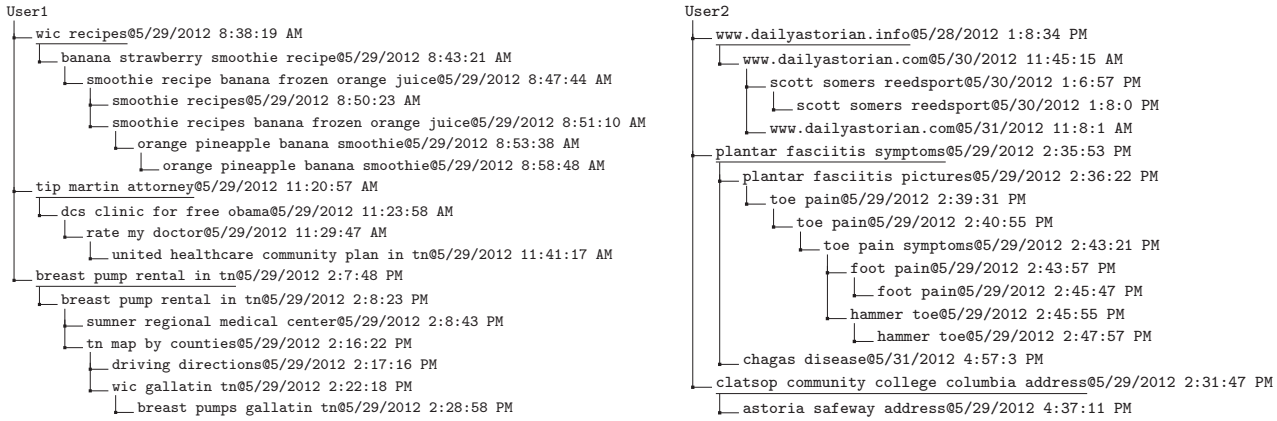


Figure 4.3: Identified latent search task structure.

Table 4.9: Statistics of extracted search tasks.

Query/Task	UniQuery/Task
4.41±7.48	2.80±4.04
Intent/Task	% of NavTask
1.47±1.20	25.37
Query/NavTask	UniQuery/NavTask
2.45±2.67	1.38±0.80
P(non-nav|nav)	P(nav|non-nav)
0.288	0.124

As shown in Table 4.9, user's search intent in each extracted task is quite concentrated: despite the fact that there are in average 4.41 queries in a task, there are only 1.47 different intents. Particularly, when the user's intent is purely navigational, the task will get mostly simplified: only 1.38 unique queries per task. And more than 25% identified tasks only contain navigational queries. Another interesting phenomenon we found is the transition probability between the navigational and non-navigational queries, which is estimated within the identified tasks, is quite different: the chance a user issues a non-navigational query after a navigational query is much lower than the opposite direction. One possible explanation for this is that when user issues a non-navigational query, they usually do not have a clear sense of where to find the information yet, so they are more likely to keep submitting the questions to the search engine; but when they have specific destination in mind, they would start to issue questions to explore more aspects of the information they are interested in.

4.7 Conclusions

Search tasks frequently span multiple sessions, and thus developing methods to extract these tasks from historic data is central to understanding longitudinal search behaviors and in developing search systems to support users' long-running tasks. In this chapter, we have presented a novel method for learning to accurately extract cross-session search tasks from users' historic search activities. We developed a semi-supervised clustering model based on the latent structural SVM framework, which is capable of learning inter-query dependencies from users' searching behaviors. A set of effective automatic annotation rules are proposed as weak supervision to release the burden of manual annotation. Comprehensive experimentation using large-scale search logs from a commercial search engine demonstrated the superior performance of our method in identifying cross-session search tasks versus a number of state-of-the-art algorithms. Importantly, we were able to obtain performance gains while reducing the reliance on costly human annotations via the automatically generated weak supervision. The results are promising and pave the way for a range of future work in this area, including user modeling and long-term task based personalization.

Chapter 5

Personalized Ranking Model Adaptation

Search engine systems train and apply a single ranking model across all users, but searchers' information needs are diverse and cover a broad range of topics. Hence, a single user-independent ranking model is usually insufficient to satisfy different users' result preferences. Conventional personalization methods learn separate models of user interests and use those to re-rank the results from the generic model. Those methods require significant user history information to infer user preferences, have low coverage in the case of memory-based methods that learn direct associations between query-URL pairs, and have limited opportunity to markedly affect the ranking given that they only re-order top-ranked items.

In this chapter, we propose a general ranking model adaptation framework for personalized search. Using a given user-independent ranking model trained off-line and limited number of adaptation queries from individual users, the framework quickly learns to apply a series of linear transformations, e.g., scaling and shifting, over the parameters of the given global ranking model such that the adapted model can better fit each individual user's search preferences.

5.1 Introduction

Prior research has demonstrated that users' aggregated clicks are informative for learning their result preferences and developing global search result ranking models [70, 3]. However, these models only reflect the common preferences across all searchers. Adapting the global ranking model towards each individual's search preferences to maximize search utility for each user is more desirable. Existing personalization methods require rich user history information to learn user preferences [104, 107] (meaning that they are slow to adapt to user interests and interest dynamics), have low coverage in the case of memory-based methods that learn direct associations

between query-URL pairs [109], and have limited opportunity to affect the ranking given that they frequently only re-order top-ranked items [11]. In this work, we proposed a method that effectively learns to adapt a generic ranking algorithm on a per-user basis, and overcomes some or all of the aforementioned challenges faced by existing personalization approaches.

Beside the adapted model's ranking performance, adaptation *efficiency* is also a primary consideration in this work. Existing work of ranking model adaptation in information retrieval (IR) mainly focuses on domain adaptation, e.g., from Web search to image search, where the goal of adaptation is to estimate a new ranking model for a target domain using information from a related source domain [53, 54, 27, 52]. Rooted in the classifier adaptation problem in transfer learning (c.f. [95]), the general assumption of domain adaptation in IR is that in the target domain there are insufficient labeled queries to accurately estimate a ranking model, but there is adequate supervision in the source domain. Therefore, to help model learning in the target domain, the adaptation methods need to effectively exploit supervision from the source domain. However, since most of existing methods estimate the adapted model in an offline manner, adaptation efficiency has received little attention in prior research. Nevertheless, in the scenario of adapting a generic ranking model for personalized search (the focus of this chapter), adaptation efficiency becomes an equally important criterion for two main reasons: 1) such an operation must be executable on the scale of all the search engine users; 2) due to the dynamic nature of users' search intent and the need to offer searchers a great experience quickly, search engines cannot wait weeks or even days to collect adaptation data, since by then user preferences may have already shifted or they may have switched to another search engine. Our specific emphasis on adaptation efficiency prohibits us from directly applying most of existing domain adaptation methods.

In this work, we propose a general framework for ranking model adaptation, which enables rapid personalization, based on the linear-regression-based model adaptation methods widely studied in automatic speech recognition (e.g., maximum likelihood linear regression [79] and minimum classification error linear regression [62]). In particular, we assume in a parametric ranking model different users' ranking preferences can be fully characterized by different settings of the model parameters. For example, some users might prefer high authority websites, i.e., larger weight on the static rank score (page quality independent of query); while other users would emphasize

query-term matching in documents, e.g., larger weight on retrieval-score features such as BM25. As a result, adjustment of the generic ranking model's parameters with respect to individual user's ranking preference, e.g., click feedback, is necessary to satisfy their distinct ranking requirements.

In the per-user basis ranking model adaptation scenario, the lack of adaptation queries is a serious problem leading to sparse observations of ranking features in each user. To alleviate the sparsity problem, transformations are shared across features in a group-wise manner, such that it is possible to adapt the parameters of features that are not observed in the adaptation data. The proposed framework is general, and we demonstrated the detailed instantiation of the framework to three frequently used learning-to-rank algorithms, i.e., RankNet [19], LambdaRank [20] and RankSVM [69], where several important properties of the proposed adaptation framework are unveiled.

To evaluate the effectiveness of the propose adaptation framework, we collected a large set of search logs from a major commercial Web search engine, and compared the proposed method against several state-of-the-art ranking model adaptation methods. Through extensively comparisons, our proposed method achieved significant improvement, in not only adaptation efficiency (measured in terms of the number of queries until reaches a performant state), but also in terms of adaptation accuracy, against the baseline methods.

5.2 Related Work

There are two major types of research closely related to our work, namely, ranking model adaptation and personalized search.

The main body of ranking model adaptation studied in IR focuses on domain adaption, which can be categorized into three classes. One popular class is instance-based adaptation [53, 42, 27], which assumes certain parts of data in source domain can be reused for target domain by re-weighting. Chen et al. [27] weighted the queries in source domain by a heuristically defined utility function. In [53], Gao et al. employed a binary classifier to separate the documents in target domain from those in source domain, and then defined the importance of each source-domain document by the output of this classifier. The second category of work is feature-based [26], where a new feature representation is learned for the target domain and used to transfer knowledge across domains.

Chen et al. proposed CLRank in [26], which constructs a new ranking feature representation so as to reduce the distributional difference between source and target domain. The third category of approaches is model-based [54, 52], which assumes the source and target ranking models share some parameters or priors. Geng et al. [54] regularized target-domain ranking model training by the given model from source domain. Gao et al. [52] updated the source-domain model by the training errors on the adaptation data via stochastic gradient boosting algorithm.

To the best of our knowledge, few work attempts to adapt a generic ranking model for each individual user. Under efficiency constraints, both instance-based and feature-based methods are infeasible for this task, because they have to operate on numerous instances in the source domain, which is prohibitively expensive to perform for each single user. To avoid costly operation for each user, our proposed method falls into the class of model-based adaptation: we update the parameters of global ranking model for each individual user according to the observed click feedback. To alleviate the problem of sparse observation in adaptation data, transformations are shared across features so that unseen features can also be effectively updated.

The task of personalized search aims at leveraging information about individual users to identify the most relevant search results for them. The main stream of existing personalization techniques targets at extracting user-centric profiles or features, e.g., location, gender and click history, and incorporating such information into the original ranking function. Teevan et al. encoded user profiles extracted from relevance feedback to re-rank the retrieved documents [108]. Dou et al. [41] performed a large-scale evaluation of several personalized search strategies, e.g., user profile based re-ranking [28], and revealed that personalization has mixed effects on the ranking performance: some queries will get promising improvement while some even get hurt. These and other personalization models (e.g., [104, 107]) use significant volumes of search history to learn interest profiles for each user, requiring sufficient data available to perform personalization effectively.

Memory-based personalization techniques learn direct associations between query-URL pairs [109] (e.g., given this query, the current user consistently selects a particular URL), which can perform well given high revisitation likelihoods, but has limited query coverage. Shen et al. [99] developed a context-sensitive language model by introducing both click feedback and preceding queries for short-term personalization. However, these short-term models are specific to context

and cannot generalize well to accommodate user's general preferences. Once a model is learned, a common strategy for the *application* of personalization is to re-rank the top-n results (e.g., [11] and [41]). This means that the personalized models do not have the opportunity to promote results of low general interest (i.e., outside of the top n), but of high interest to the current user, into the top-ranked results.

5.3 Ranking Model Adaptation

Inspired by the linear regression based model adaptation methods in speech recognition [79, 62], we propose a general framework to perform ranking model adaptation. We assume that a global ranking model is trained based on a large user-independent training set. For each user, an adapted model is obtained from applying a set of learned linear transformations, e.g., scaling and shifting, to the parameters of the global model based on each individual user's adaptation data, e.g., query with corresponding clicks.

In the following discussions, we first describe our general framework of ranking model adaptation, and then we take three frequently used learning to rank algorithms, i.e., RankNet [19], LambdaRank [20] and RankSVM [69], as examples to demonstrate the detailed procedures of applying the proposed adaptation framework.

5.3.1 General Framework

For a given set of queries $Q^u = \{q_1^u, q_2^u, \ldots, q_m^u\}$ from user u, each query q_i^u is associated with a list of document-label pairs $\{(x_{i1}^u, y_{i1}^u), (x_{i2}^u, y_{i2}^u), \ldots, (x_{in}^u, y_{in}^u)\}$, where x_{ij}^u denotes a retrieved document represented by a V-dimensional vector of ranking features, and y_{ij}^u is the corresponding relevance label indicating if the document x_{ij}^u is relevant to user u (e.g., clicks). Since our focus of this work is on user-level ranking model adaptation, in the following discussions we would ignore the superscript u to make the notations concise when no ambiguity is invoked.

A ranking model f is defined as a mapping from a document x_{ij} to its ranking score s_{ij}, i.e., $f : x_{ij} \rightarrow s_{ij}$, such that when we order the retrieved documents for query q by f, certain ranking metric, e.g., mean average precision (MAP) or precision at k (P@k) [9], is optimized. Such ranking model can be manually set, or estimated by an automatic algorithm based on a collection of

annotated queries [84]. In this work, we focus on *linear* ranking models, which can be characterized by a parametric form of linear combination over ranking features, i.e., $f(x) = w^{\mathsf{T}}x$, where w is the linear coefficients for the corresponding ranking features.

Denoting $f^s(x) = w^{s\mathsf{T}}x$ as the given global ranking model estimated in a user-independent manner, the adaptation of $f^s(x)$ for each individual user is performed via the linear transformations defined by a $V \times (V+1)$ dimensional matrix A^u, by which three linear operations, i.e., scaling, shifting and rotation, can be encoded. More precisely,

$$f^u(x) = (A^u \tilde{w}^s)^{\mathsf{T}}x \tag{5.1}$$

where $\tilde{w}^s$ is an augmented vector of w^s, i.e., $\tilde{w}^s = (w^s, 1)$, to facilitate the shifting operation for parameter adaptation.

There are two major considerations in designing the transformation matrix A^u. First, a full transformation matrix has the number of $O(V^2)$ free parameters, which is redundant and even larger than the number of parameters we need to estimate for learning a new ranking model for each user (i.e., in the scale of $O(V)$). As a result, it is infeasible for us to estimate a full transformation matrix for every user. To reduce the size of free parameters in A^u, we will only focus on the scaling and shifting operations for adapting the parameters in $f^s(x)$. This reduces the size of free parameters in A^u from $O(V^2)$ to $O(V)$. Second, a more important consideration is how to alleviate the problem of sparse observation of ranking features in the limited adaptation data. In order to properly update the parameters for unseen features, we organize the ranking features in groups and share the same shifting and scaling transformations of the parameters within the same group.

Based on the above considerations, we design the transformation matrix A^u to be the following specific form,

$$\mathbf{A}^u = \begin{pmatrix} a^u_{g(1)} & 0 & \cdots & b^u_{g(1)} \\ 0 & a^u_{g(2)} & \cdots & b^u_{g(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a^u_{g(V)} & b^u_{g(V)} \end{pmatrix}_{V \times (V+1)}$$

where $g(\cdot)$ is a feature grouping function, which maps V original ranking features to K different

groups, a_k^u and b_k^u denote the scaling and shifting operations applied to the linear coefficients w^s of the global model $f^s(x)$ in group k. As a result, Eq (5.1) can be realized as,

$$f^u(x) = \sum_{k=1}^{K} \sum_{g(i)=k} (a_k^u w_i^s + b_k^u) x_i \tag{5.2}$$

The grouping function $g(\cdot)$ defines the transformation sharing among original ranking features. It enables the observations from seen features to be propagated to unseen features within the same group during adaptation, which is the key to conquer the problem of sparsity in the limited adaptation data. However, defining the optimal grouping of ranking features is non-trivial; we postpone the discussion of estimating $g(\cdot)$ to Section 5.3.4.

Once the grouping function $g(\cdot)$ is given, another important component in our adaptation framework is the choice of criterion to estimate the optimal transformation matrix A^u. An ideal transformation should be able to adjust the generic ranking model to meet each individual's ranking preference, i.e., maximizing the search utility for each user. In the study of learning-to-rank in IR, various types of objective functions, e.g., pairwise and listwise loss, have been proposed to realize the goal of optimizing ranking metrics [84]. Therefore, to make the proposed framework generally applicable, we do not restrict our adaptation objective to any specific form, but instantiate it with the objective function from the ranking algorithm we choose to adapt.

We want to emphasize that though in our framework we utilized the objective function from the ranking algorithm to be adapted as the criterion for estimating the transformation matrix A^u, it does not restrict the global model to be estimated by the same ranking algorithm. As long as the global model and adapted model share the same model structure, e.g., neural network structure in RankNet and linear model structure in RankSVM, the proposed adaptation framework is applicable.

To summarize, our general framework for ranking model adaptation can be formalized as follows,

$$\begin{aligned} &\min_{A^u} \quad L_{\text{adapt}}(A^u) = L(Q^u; f^u) + \lambda R(A^u) \\ &\text{where} \quad f^u(x) = (A^u \tilde{w}^s)^{\mathsf{T}} x \text{ and } \tilde{w}^s = (w^s, 1) \end{aligned} \tag{5.3}$$

where $L(Q^u; f^u)$ is the objective function defined in the ranking algorithm we chose to adapt, e.g., cross-entropy in RankNet or hinge loss in RankSVM, $R(A^u)$ is a regularization function defined on the transformation matrix A^u, λ is a trade-off parameter, and w^s is the parameters in the global ranking model.

Next, we will discuss the detailed instantiation of the proposed ranking model adaptation framework to three state-of-the-art learning to rank models, i.e., RankNet, LambdaRank and RankSVM.

5.3.2 Adapting RankNet & LambdaRank

RankNet [19] is a probabilistic learning-to-rank algorithm, which models the probability that a document x_{ij} is ranked higher than x_{il} for query q_i, i.e., $P(y_{ij} > y_{il})$. A logistic function is employed to map two outputs of predicted ranking scores, e.g., s_{ij} and s_{il}, to probability of ordering,

$$P(y_{ij} > y_{il}) = \frac{1}{1 + e^{-(s_{ij} - s_{il})}}.$$

The training objective function in RankNet is defined as the cross-entropy between the predicted pairwise ordering probabilities and the observed pairwise preferences in the training data, i.e.,

$$L_{\text{RankNet}} = \sum_{q_i} \sum_{y_{ij}, y_{il}} -\bar{P}(y_{ij} > y_{il}) \log P(y_{ij} > y_{il}) - (1 - \bar{P}(y_{ij} > y_{il})) \log(1 - P(y_{ij} > y_{il})) \quad (5.4)$$

where $\bar{P}(y_{ij} > y_{il})$ is the empirically estimated probability that x_{ij} is ranked higher than x_{il}.

RankNet is usually optimized via a neural network. Because in each layer of a neural network, every neuron's output is linearly combined to feed into the next layer, our adaption framework can be smoothly applied to the linear weights for each neuron (e.g., different transformation matrices for each neuron in the hidden layers). In order to understand the effect of adaptation in RankNet, we will use the RankNet with no hidden layers for discussion, but the same procedure can be applied to general RankNet with an arbitrary number of hidden layers.

To adapt RankNet, we take the same cross-entropy function defined in Eq (5.4) as our adapta-

tion objective, and define the following regularization function on matrix A^u,

$$R(A^u) = \frac{1}{2}\sum_{k=1}^{K}(a_k^u - 1)^2 + \frac{\sigma}{2}\sum_{k=1}^{K} b_k^{u2} \tag{5.5}$$

where we penalize the transformation which increases the discrepancy between the adapted model and the generic model, and σ is a parameter that controls the balance between the penalty on shifting and scaling.

As a result, the gradient with respect to the scaling parameter a_k^u can be calculated as,

$$\begin{aligned}\frac{\partial L_{\text{adaptRankNet}}(A^u)}{\partial a_k^u} &= \sum_{q_i \in Q_u}\sum_{y_{ij}>y_{il}} [P(y_{ij} > y_{il}) - 1]\frac{\partial (A^u \tilde{w}^s)^{\mathsf{T}} \Delta x_{ijl}}{\partial a_k^u} + \lambda \frac{\partial R(A^u)}{\partial a_k^u} \\ &= \sum_{q_i \in Q_u}\sum_{y_{ij}>y_{il}} [P(y_{ij} > y_{il}) - 1] \sum_{g(v)=k} w_v^s \Delta x_{ijlv} + \lambda(a_k^u - 1)\end{aligned} \tag{5.6}$$

where Δx_{ijl} is a V-dimensional vector defined as $\Delta x_{ijl} = x_{ij} - x_{il}$. Accordingly, the gradient with respect to b_k^u is,

$$\frac{\partial L_{\text{adaptRankNet}}(A^u)}{\partial b_k^u} = \sum_{q_i \in Q_u}\sum_{y_{ij}>y_{il}} [P(y_{ij} > y_{il}) - 1] \sum_{g(v)=k} \Delta x_{ijlv} + \lambda\sigma b_k^u \tag{5.7}$$

The above gradients induce a new neural network defined over the linear transformations, where the connection among neurons is specified by the grouping function $g(\cdot)$: the term $P(y_{ij} > y_{il}) - 1$ in Eq (5.6) and Eq (5.7) represents the prediction error of the global ranking model on the adaptation data; and based on this error, the gradients specify the direction in which the transformation should take. We can note that the gradients for a_k^u and b_k^u are estimated based on all the observations in the same group; as a result, by sharing such jointly estimated transformations, the parameters for the unseen features can also get properly updated.

To generalize this procedure to RankNet with multiple hidden layers, we only need to replace the error term defined by $P(y_{ij} > y_{il}) - 1$ with the corresponding back-propagation error in each hidden layer in Eq (5.6) and Eq (5.7), and the original optimization procedure for RankNet can be directly applied to the adapted problem. One thing we should note is that since one can set different number of neurons in each hidden layer, to apply the proposed adaptation in a RankNet

with multiple layers, we need to specify the grouping function $g(\cdot)$ for each neuron in the hidden layers. This can be achieved via the clustering method discussed in Section 5.3.4.

Based on the discussion of adapting RankNet within the proposed framework, it is straightforward to adapt LambdaRank [20] in a similar manner. As a listwise learning-to-rank algorithm, LambdaRank modifies the error term in RankNet by adding an additional correction term and names such modified error as lambda function,

$$\lambda_{ijl} = [P(y_{ij} > y_{il}) - 1]|\Delta_{\text{IR-Metric}}| \tag{5.8}$$

where $|\Delta_{\text{IR-Metric}}|$ is the change of any specific ranking metric, e.g., MAP or NDCG, given by swapping the rank positions of document x_{ij} and x_{il} while leaving the rank positions of all other documents unchanged.

Therefore, to adapt LambdaRank within our framework, we only need to replace the error function of the output layer in RankNet with the lambda function defined in Eq (5.8), and all the other procedures are the same as in RankNet.

5.3.3 Adapting RankSVM

RankSVM [69] is a classic pairwise learning-to-rank algorithm, in which the learning problem is formalized as,

$$\begin{aligned} \min_{w,\xi_{ijl}} \quad & \frac{1}{2}||w||^2 + C\sum_{q_i}\sum_{j,l}\xi_{ijl} \\ \text{s.t.} \quad & w^{\mathsf{T}}\Delta x_{ijl} \geq 1 - \xi_{ijl}, \forall q_i, x_{ij}, x_{il} \\ & \xi_{ijl} \geq 0 \\ \text{where} \quad & y_{ij} > y_{ij} \text{ and } \Delta x_{ijl} = x_{ij} - x_{il} \end{aligned} \tag{5.9}$$

where C is a trade-off parameter to control the balance between model complexity and empirical hinge loss over the identified preference pairs from the training data.

To adapt RankSVM, we keep the hinge loss defined in Eq (5.9) as our adaptation objective, and use the same regularization function for A^u defined in Eq (5.5). By taking the linear transformation

$w^u = A^u \tilde{w}^s$ into Eq (5.9), we get the adapted RankSVM as,

$$\min_{\mathbf{a}^u, \mathbf{b}^u, \xi_{ij}} \frac{1}{2}\sum_{k}^{K}(a_k^u - 1)^2 + \frac{\sigma}{2}\sum_{k}^{K}(b_k^u)^2 + C\sum_{q_i}\sum_{j,l}\xi_{ijl} \tag{5.10}$$

$$\text{s.t. } w^{u\mathsf{T}}\Delta x_{ijl} \geq 1 - \xi_{ijl}, \forall q_i, x_{ij}, x_{il}$$

$$\xi_{ijl} \geq 0$$

$$\text{where } w^u = A^u\tilde{w}^s,\ y_{ij} > y_{il}, \text{ and } \Delta x_{ijl} = x_{ij} - x_{il}$$

Since the input for RankSVM training is only the pairs of documents, in the following discussion, we will briefly denote Δx_{ijl} as $\vec{x}_t$, in which the subscript t ranges over all the preference pairs in the adaptation set, to simplify the notations. Following the conventional derivation of RankSVM, we get the dual problem of Eq (5.9) by introducing a set of Lagrange multipliers α,

$$\max_{\alpha} \sum_{t}\Big[1 - f^s(\vec{x}_t)\Big]\alpha_t - \frac{1}{2}\alpha^{\mathsf{T}}\Big[K_1(\vec{\mathbf{x}}, \vec{\mathbf{x}}) + K_2(\vec{\mathbf{x}}, \vec{\mathbf{x}})\Big]\alpha \tag{5.11}$$

$$s.t.\ 0 \leq \alpha_t \leq C, \forall t$$

$$\text{where } K_1(\vec{x}_t, \vec{x}_r) = \sum_{k}^{K}\Big(\sum_{g(v)=k} w_v^s \vec{x}_{tv}\Big)\Big(\sum_{g(v)=k} w_v^s \vec{x}_{rv}\Big)$$

$$K_2(\vec{x}_t, \vec{x}_r) = \frac{1}{\sigma}\sum_{k}^{K}\Big(\sum_{g(v)=k} \vec{x}_{tv}\Big)\Big(\sum_{g(v)=k} \vec{x}_{rv}\Big)$$

By solving the above dual problem, we can get the optimal transformations as,

$$a_k^u = 1 + \sum_t \alpha_t \sum_{g(v)=k} w_v^s \vec{x}_{tv}$$

$$b_k^u = \frac{1}{\sigma}\sum_t \alpha_t \sum_{g(v)=k} \vec{x}_{tv}$$

The effect of the proposed adaptation on RankSVM is clearly depicted in its dual problem. First, as we know that the linear coefficients in front of the Lagrange multipliers α in Eq (5.11) correspond to the separation margin for each training instance in SVM. In the adapted problem, the margin is rescaled according to the global model $f^s(\vec{x}_t)$'s prediction on the adaptation data: if the global model can well separate the adaptation pair $\vec{x}_t$, i.e., $f^s(\vec{x}_t) > 0$, the margin decreases,

indicating the adapted model can tolerate more error on this case; if the global model fails to correctly predict the order for this pair, i.e., $f^s(\vec{x}_t) \leq 0$, the margin increases, and $\vec{x}_t$ becomes a more important instance in adaptation for this particular user. This precisely interprets the effect of model-based adaptation: we only update the global model when it makes mistake on the adaptation data; otherwise keep it intact. Second, the proposed linear transformations induce two new kernels in a compressed space: $K_1(\vec{x}_t, \vec{x}_s)$, corresponding to the scaling operation, defines a compound polynomial kernel over the ranking features projected by the global ranking model w^s; and $K_2(\vec{x}_t, \vec{x}_s)$, corresponding to the shifting operation, defines another compound polynomial kernel over the original ranking features. Both kernels work in a compressed K-dimensional space determined by the feature group mapping $g(\cdot)$, and are interpolated by the balance parameter σ between the regularizations for shifting and scaling. As a result, non-linearity is introduced to the original linear RankSVM model, and such non-linearity helps the model to leverage the observations from seen features to the unseen ones in the same group.

5.3.4 Feature Grouping

In the proposed framework, a feature grouping function $g(\cdot)$ is used to organize the ranking features so that shared transformation is performed on the parameters of features in the same group. Such grouping can be given *a priori* according to the design of ranking features, or be determined by data-driven approaches based on a given set of queries and documents. In this work, we proposed and compared three possible ways of creating such feature groups.

The first grouping method is based on the name of ranking features. Ranking features are usually described by the way they are generated, e.g., *BM25 of Body*, *BM25 of Title* [83], such that the name of a ranking feature provides informative indication of its functionality. Given the naming scheme of features in a collection, we can manually define patterns to cluster the features into groups. We denote such grouping method as *Name*.

The second method is based on the co-clustering algorithms developed in document analysis. Similar to [40], we first project the document-feature matrix into a lower dimensional space by singular value decomposition (SVD), and then perform k-means algorithm to group the features into K clusters based on this low dimensional representation. We name such grouping method as

SVD.

The third method groups features by the corresponding learned parameters in the ranking models. We first evenly split the training collection into N non-overlapping folds, and train a single ranking model, e.g., RankSVM, on each fold. Then, we create a $V \times N$ matrix by putting the learned parameters of those N independent models together, on which k-means algorithm is applied to extract K feature groups. We name such grouping method as *Cross*.

The *Name* method requires the collection to have a reasonable feature naming scheme; if the ranking features are arbitrarily named, e.g., named by ID, such method cannot be used. The *SVD* method is generally applicable since it only requires a collection of documents represented by the ranking features. In the *Cross* method, besides a set of documents, relevance judgment of each document is also needed to estimate the grouping of features. In particular, for RankNet with multiple layers, the *Cross* method can be used to estimate the grouping function for each neuron in the network based on the learned weights of connections.

5.3.5 Discussion

The advantages of the proposed adaptation framework are four folds. First, it is a general framework for ranking model adaptation, which is applicable to a majority of existing learning-to-rank algorithms [84]. Second, the proposed adaptation framework is model-based, unlike the instance-based and feature-based adaptation methods, it does not need to operate on the numerous data from source domain, which makes the per-user basis ranking model adaptation feasible. Third, the same optimization technique for the original learning algorithm can be directly applied with little change, such that it does not increase the complexity of solving the adaptation problem. And in the adaptation phase, we only need to solve the optimization problem over a small amount of adaptation data, which ensures the computational efficiency for performing the adaptation on the scale of all search engine users. Fourth, most importantly, transformation is shared across features in a group-wise manner. According to Eq (5.2), the same linear transformation is applied onto the parameters of features in the same group, which renders several important properties in the adapted ranking models: in RankNet, the gradients for scaling (i.e., Eq (5.6)) and shifting (i.e., Eq (5.7)) operations are estimated based on all the observations in the same group; while in RankSVM, two

new non-linear kernels are induced over the original linear function space. As a result, even though we might not observe a specific feature occurring in the adaptation data, we can still propagate the information from other features in the same group to update it properly.

5.4 Experimental Results

In order to evaluate the proposed adaptation framework, we performed a series of experiments on a large scale search data set sampled from the query logs from a major commercial Web search engine. A set of state-of-the-art ranking model adaptation methods have been included as the baselines to validate the effectiveness of the proposed method.

5.4.1 Dataset and Settings

We extracted five days' search logs from May 27 2012 to May 31 2012 from a major commercial Web search engine for our experiments. During this period, a subset of users were randomly selected and all their search activities were collected, including the anonymized user ID, query string, timestamp, top 10 returned document lists and the corresponding clicks. The queries were ordered by their timestamp for each user, and the documents were listed by their original order returned by the search engine under each query.

To apply the proposed adaptation method and compare with the baselines according to user's click feedback, we can only use the queries with clicks. Therefore, in our experiment we filtered out the queries without clicks and required each user to have at least two queries with clicks, i.e., one for adaptation and one for testing.

We also sampled a large set of manually annotated query logs from our existing data collection as the user-independent training set for adaptation. Each query-document pair in this annotation set is labeled with a five-grade relevance score, i.e., from 0 for "bad" to 4 for "perfect." Documents in both of the selected user data set and annotation data set are represented by a set of 1,830 ranking features selected from their overlapped feature set, including frequently used ranking features such as BM25, language model score and PageRank. Using the language of domain adaptation, we treat the collection of annotated queries as our source domain and each user's queries with clicks as target domain. This setting provides a good simulation for real Web search scenario, where the generic

Table 5.1: Statistics of annotation and user data set.

	# Users	# Queries	# Documents
Annotation Set	-	49,782	2,320,711
User Set	34,827	187,484	1,744,969

rankers in use are usually trained on offline annotated data, and thus it helps us compare the effectiveness of different ranking model adaptation methods. The basic statistics of the annotation set and selected user set are summarized in Table 5.1.

Preference pairs are extracted from user's clicks to reflect her unique search requirements. In order to ameliorate the positional biases inherent in click data [3], we followed Joachims et al.'s method to extract the click preference pairs [70]. In particular, we employed two click heuristics: for a given query q with a ranked document list $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ returned by the search engine,

1. "*Click* $\succ$ *Skip Above*": extract preference pair $x_i \succ x_j$ for all pairs $1 \leq j < i$ with $y_i > y_j$.

2. "*Click* $\succ$ *Skip Next*": extract preference pair $x_i \succ x_{i+1}$ for all $y_i > y_{i+1}$.

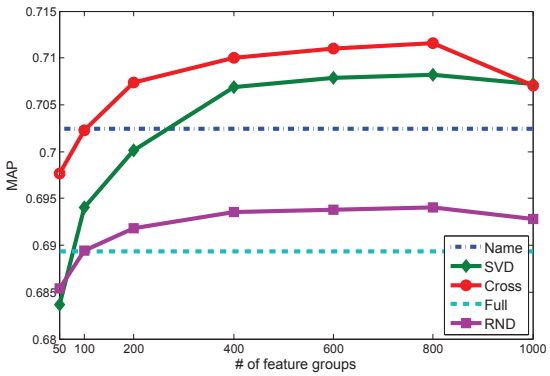
In order to avoid defining different feature grouping functions for different ranking algorithms we selected to adapt, e.g., in RankNet each neuron in the hidden layers needs a possibly different grouping function but in RankSVM only one grouping function is needed for the original features, we decided *not* to use hidden layers in the neuron networks for RankNet and LambdaRank in our experiment. As a result, the same grouping function defined on the original ranking features can be directly used in RankNet, LambdaRank and RankSVM. Based on this setting, a LambdaRank model optimizing NDCG@10 is trained on the annotation set and used as the source model for adaptation in the following experiments[1]. The trade-off parameter λ (in Eq (5.3)) and σ (in Eq (5.5)) in our method are selected by 5-fold cross validation on the whole user set in advance.

To quantitatively compare different adaptation methods' performance, we employed a set of standard IR evaluation metrics: by treating all the clicked documents as relevant, we calculated Mean Average Precision (MAP), Precision at 1 (P@1), Precision at 3 (P@3) and Mean Reciprocal Rank (MRR). Definitions of these metrics can be found in [9].

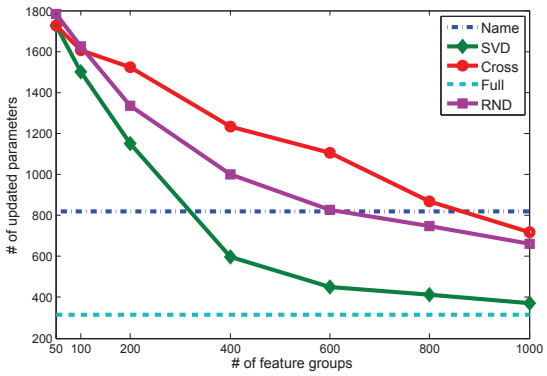
[1]Since we only used a subset of annotated queries and features, the results here do not reflect the actual performance of the search engine.

5.4.2 Analysis of Feature Grouping

In the proposed adaptation framework, the grouping of features has a substantial impact on the adaptation performance, since transformation will be shared for the parameters of features in the same group. Ideally, we should put parameters that need to be updated synchronously in the same group. In this experiment, we evaluated the three feature grouping methods, i.e., *Name*, *SVD*, and *Cross*, proposed in Section 5.3.4. For comparison purposes, we also included two trivial grouping methods: 1) "*Full*," which creates a group for every single feature, i.e., no transformation is shared across features; 2) "*RND*," which randomly allocates features into K groups.



(a) Adaptation performance with different K



(b) Number of updated parameters with different K

Figure 5.1: Analysis of feature grouping in RankNet adaptation.

In our data set, 413 feature groups are extracted by the *Name* method. The two data-driven approaches, *SVD* and *Cross*, were applied on the annotation set to identify the feature grouping, but we have to specify the group size K in advance. To analyze the effect of group size K in our proposed adaptation framework, we evaluated the adaptation performance of RankNet by varying the setting of K. To control the number of adaptation queries in each user, which influences the adaptation performance, we selected a subset of users, where each user has at least six queries with clicks (close to the average number of queries with clicks per user in our collection), and used the first three queries for adaptation and last three queries for testing in each user. This leads to a collection of 8,879 users with 112,069 queries.

The MAP ranking performance of RankNet with different feature grouping methods is shown in Figure 5.1 (a). First, it is clear that a properly set K is crucial for both *SVD* and *Cross* methods.

The more groups we set, the more adaptation parameters we need to estimate based on the limited adaptation data. But if we set too few feature groups, the discriminations among the features will be lost due to inaccurate feature grouping and hence we may mistakenly update some features which are irrelevant to the user's preference. Besides, Figure 5.1 (a) also shows that the adaptation performance is less sensitive to K around its optimal value, i.e., the performance as indicated by MAP is stable in a wide range of K from 400 to 800, for both *SVD* and *Cross*.

Another observation in Figure 5.1 (a) is that *Cross* performed consistently better than the other grouping methods under the same setting of K. Because in the *Cross* method features with similar contributions (i.e., linear weights) to document ranking are grouped together, they tend to update synchronously. Hence, sharing transformations among such features is more desirable. In contrast, other grouping methods cannot exploit such relationship among the features, e.g., *SVD* only exploits the co-occurrence relation between features, and thus they achieved worse results.

In order to understand the in-depth effect of feature grouping in our adaptation framework, we computed the average number of updated parameters in the adapted ranking model for each user with respect to different group size K and illustrated the results in Figure 5.1 (b). We can note that on average only 316 features (with a standard deviation of 214) can be observed (with non-zero value) in the adaptation data according to the result of *Full* method. However, because of sharing adaptation transformations across features in our framework, the number of parameters that have been actually adjusted is much larger. For example, with 800 groups, about 870 parameters (with a standard deviation of 220) on average are effectively updated by the *Cross* method, indicating that more than 60% of updated parameters are adapted without actual observation. On the other hand, when K becomes smaller, the number of updated parameters increases rapidly. Consequently, using too fewer groups forces less relevant features get updated by the shared transformation, which in turn degrades the overall adaptation performance.

Similar results are also observed in LambdaRank and RankSVM. In the following experiments, to avoid selecting K for each individual user and the variation of performance introduced by this factor, we fix K to be 800 for both *SVD* and *Cross*.

5.4.3 Comparison of Adaptation Performance

Algorithms for Comparison: To make a thorough evaluation of the proposed adaptation method, we included several state-of-the-art ranking model adaptation methods as baselines, covering instance-based, feature-based and model-based methods, for comparisons. We describe the employed baselines briefly in below.

TransRank [27] is an instance-based ranking model adaptation method, in which a utility function is defined to select the top k important queries from source domain into target domain for model training. IW-RankSVM [53] is another instance-based adaptation method, which re-weights the instances in source domain by measuring its distance to the classification hyperplane between source and target domain, and only uses those re-weighted instances from source domain for target-domain model training. CLRank [26] is a feature-based adaptation method, which constructs a new joint feature representation for both source and target domain to reduce the distributional difference between these two domains.

However, it would be prohibitively expensive if we directly applied these baselines for every user, because such methods have to access all the offline training data during adaptation. To make these methods applicable in our application scenario, we pooled all the user's adaptation data together to form a combined user collection, on which the above baseline methods are applied. In addition, we also trained a new LambdaRank model optimizing MAP on this integrated user collection as a baseline, and named it Target-Only.

RA-RankSVM [54] is a model-based adaptation method, which treats the ranking model from source domain as an additional regularization for model training in the target domain. Based on RA-RankSVM, we used the same regularization term in RankNet and LambdaRank to get the corresponding RA-RankNet and RA-LambdaRank baselines. Besides, without knowledge about the global model, we estimated a ranking model only based on each individual user's adaptation data, and denoted such method as Tar, e.g., Tar-RankSVM, accordingly.

All baseline methods' hyper-parameters, e.g., trade-off parameter C in RA-RankSVM, are tuned by 5-fold cross validation on the whole user data set in advance.

Table 5.2: Comparison of per-user basis ranking model adaptation performance.

	MAP	P@1	P@3	MRR
Tar-RankSVM	0.6240	0.4905	0.2335	0.6282
RA-RankSVM	0.6366	0.4809	0.2510	0.6410
Name-RankSVM	0.6544	0.5078	0.2546	0.6585
SVD-RankSVM	**0.6643**	**0.5209**	**0.2579**	**0.6687**
Cross-RankSVM	0.6638	0.5200	0.2574	0.6681
Tar-RankNet	0.6342	0.5051	0.2360	0.6384
RA-RankNet	0.6577	0.5267	0.2481	0.6619
Name-RankNet	0.6709	0.5425	0.2535	0.6750
SVD-RankNet	0.6751	0.5412	0.2581	0.6796
Cross-RankNet	**0.6781**	**0.5450**	**0.2593**	**0.6826**
Tar-LambdaRank	0.6436	0.5182	0.2384	0.6477
RA-LambdaRank	0.6616	0.5341	0.2479	0.6657
Name-LambdaRank	0.6814	0.5556	0.2569	0.6859
SVD-LambdaRank	0.6878	0.5590	0.2616	0.6925
Cross-LambdaRank	**0.6922**	**0.5662**	**0.2629**	**0.6969**

Adaptation Accuracy

We performed the experiment on all user data in our collection, in which the first 50% of queries from each user are used for adaptation and the rest are used for testing.

• **Comparison in per-user basis adaptation:** we fisrt compared the ranking performance of our proposed adaptation methods (under all the three grouping methods) with the model-based adaptation baseline methods, e.g., RA-RankSVM, RA-RankNet and RA-LambdaRank, and the baseline methods solely depend on the adaptation data, i.e., Tar-RankSVM, Tar-RankNet and Tar-LambdaRank. These are the only baselines applicable in the scenario of per-user basis ranking model adaptation. In particular, MAP metric is chosen to be optimized in the adapted LambdaRank models.

In Table 5.2, we can observe significant improvement of ranking performance from the model-based adaptation methods, i.e., our methods and RA methods, against the methods solely depending on the adaptation data, i.e., Tar methods. As discussed before, sparsity is a serious problem in per-user basis model estimation. Tar methods cannot estimate the parameters for the unseen features, and thus its ranking performance is poor. RA methods alleviate such deficiency by using the global model as back-off: for features not observed in the adaptation data, the parameters from the global model would be used. In our proposed method, besides back-off to the global

model when no observation is available (as shown in Eq (5.5)), we also propagate the observations from seen features to unseen features by transformation sharing to help the model better estimate the parameters of those unseen features. As a result, our adaptation methods, under all grouping methods, outperformed the corresponding RA adaptation methods (all the improvements are significant with $p\text{-}value<0.01$ under paired two-sample $t\text{-}test$).

Another observation in Table 5.2 is that the adapted LambdaRank performed consistently better than the adapted RankSVM and RankNet within our framework. In LambdaRank the lambda function helps the model to directly optimize the IR-related metrics, e.g., MAP in our case, while RankSVM and RankNet can only minimize pairwise loss. LambdaRank has shown better performance than those pairwise learning-to-rank algorithms in many classical ranking tasks [20]. In our adaptation framework, such advantage of LambdaRank is preserved since the same lambda function definition and optimization procedure are used as in the original LambdaRank. This demonstrates the flexibility of our adaptation framework, in which we can choose to adapt any specific ranking algorithm according to the requirement of the task.

• **Comparison with integrated adaptation:** according to the results in Table 5.2, we compared our best performing method *Cross*-LambdaRank with the instance-based and feature-based ranking model adaptation methods, and list the results in Table 5.3.

First of all, we can notice that in Table 5.3 the global ranking model trained on the annotation set did not perform well on the user testing set; while the model trained on the integrated user data improved most of the ranking metrics over 10%. This indicates evident distributional difference between the generic annotation set and user click set. Through instance re-weighting, i.e., IW-RankSVM and TransRank, or feature construction, i.e., CLRank, all baseline adaptation methods achieved improved ranking performance against the global model. For these baseline methods, since we have pooled all the users' adaptation data together, sparsity is no longer a serious problem. However, individual user's specific ranking preference will be overwhelmed once we pooled different users' clicks together. In our adaptation method, e.g., *Cross*-LambdaRank, the global model is adapted for each individual user towards maximizing the search utility based on their own adaptation data. As a result, *Cross*-LambdaRank outperformed all these baseline adaptation methods, which are originally designed for domain adaptation, in this user-oriented evaluation.

Table 5.3: Comparison of adaptation performance.

	MAP	P@1	P@3	MRR
Source-Only	0.5637	0.3356	0.2503	0.5679
Target-Only	0.6258	0.4667	0.2468	0.6298
IW-RankSVM	0.6427	0.4865	0.2506	0.6470
TransRank	0.6468	0.5202	0.2400	0.6512
CLRank	0.6590	0.5090	0.2561	0.6594
Cross-LambdaRank	**0.6922**	**0.5662**	**0.2629**	**0.6969**

• **Query-/User-level improvement analysis:** the results shown in Table 5.2 and Table 5.3 are averaged over all the users' testing queries. It is necessary to further investigate to what extend and what types of users/queries can benefit from the proposed adaptation method. We analyzed the detailed ranking results given by *Cross*-LambdaRank against the results from the global ranking model and RA-LambdaRank, which is the best baseline method according to Table 5.2 and Table 5.3.

Table 5.4: Ranking performance gain against the global model from *Cross*-LambdaRank and RA-LambdaRank on repeated and non-repeated queries.

Query Type	Method	ΔMAP	ΔP@1	ΔP@3	ΔMRR
Non-repeated	RA	-0.0337	-0.0050	-0.0292	-0.0342
	Cross	0.0204	0.0498	-0.0024	0.0206
Repeated	RA	0.2329	0.4082	0.0249	0.2331
	Cross	0.2375	0.4129	0.0273	0.2379

Query repetition is a common phenomenon in user's query log, and it is crucial for many memory-based personalization methods [109, 41]. First, we categorized the testing queries as repeated queries, if it occurred in the corresponding user's adaptation query set, and the rest as non-repeated ones; and then computed the improvement of ranking performance against the global model from *Cross*-LambdaRank and RA-LambdaRank on these two types of testing queries. As shown in Table 5.4 (all the differences are significant with p-*value*<0.01 under paired t-*test*), both methods achieved notably improvement against the global model on the repeated queries, but only *Cross*-LambdaRank attained improved results on the non-repeated queries. For the repeated queries, both methods can simply "memorize" and "promote" the documents clicked by the same user; while for the non-repeated queries, because RA-LambdaRank did not have any direct obser-

vations to adjust the relevant ranking features, it could not generalize well from the adaptation data. In *Cross*-LambdaRank, the unseen features can also be updated via transformation sharing, which renders the model better ranking capability on those non-repeated queries.

In addition, we also applied a proprietary multi-label classifier to annotate the query intent into 63 categories, e.g., navigational, commerce and etc., and found that the major improvement of our method against the global model comes from the navigational queries. In detail, comparing to the global model, 44.9% navigational queries get improved MAP results and only 10.2% of them become worse. "HowTo," "Health" and "Q&A" are the major categories of queries on which our method failed to generate better ranking than the global model. We investigated such kind of queries in the user data set and found the users' clicks are mostly for exploration purposes in these kinds of informational queries (diverse clicked documents), since they might not have a clear mind of answers for these queries yet. As a result, such clicks are not so reliable to update the ranking model in the adaptation phase.

To understand what types of users can benefit from our adaptation method, we categorized the users in our collection into three classes by the number of adaptation queries they have: 1) *heavy* user, who has more than 10 adaptation queries; 2) *medium* user, who has 5 to 10 adaptation queries; and 3) *light* user, who has less than 5 adaptation queries. We calculated the ranking performance gain from *Cross*-LambdaRank against the global model averaged over the users in these three classes in Table 5.5. Besides, we also included the improvement from RA-LambdaRank against the global model in the table for comparison.

Table 5.5: User-level ranking performance gain over global model from *Cross*-LambdaRank and RA-LambdaRank.

Method	User Class	ΔMAP	ΔP@1	ΔP@3	ΔMRR
RA	Heavy	0.1843	0.3309	0.0120	0.1832
	Medium	0.1102	0.2129	0.0025	0.1103
	Light	0.0042	0.0575	-0.0221	0.0041
Cross	Heavy	0.1998	0.3523	0.0182	0.1994
	Medium	0.1494	0.2561	0.0208	0.1500
	Light	0.0403	0.0894	-0.0021	0.0406

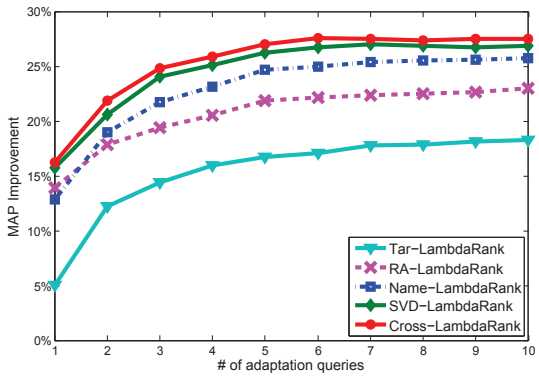
All the differences in Table 5.5, except the *Cross*/Light/ΔP@3 with the value of -0.0021, are significant with p-*value*<0.01 under paired t-*test*. We can observe that on the heavy users, who only

cover 6.8% population in our collection, *Cross*-LambdaRank and RA-LambdaRank achieved close improvement of ranking performance against the global model; while on the medium and light users, who consist 14.9% and 78.3% of the whole collection, *Cross*-LambdaRank achieved much more remarkably improvement than RA-LambdaRank. On the heavy users, both methods get relatively sufficient observations from each user to adapt the global model; while on the medium and light users, the observations become scattered and many features are not observed during adaptation. RA-LambdaRank failed to adjust the unseen features properly and only achieved modest improvement over the global model. By sharing transformation across features, *Cross*-LambdaRank better exploited the information embedded in the limited adaptation data, and attained better improvement against the global model. Besides, we also found that on the light users, both methods gave degraded P@3 results (the degradation of the *Cross* method is insignificant). We analyzed the results and found that with limited observations in this group of users, the adapted models tend to overfit the originally top ranked documents due to positional biases. As a result, the diversity of user preference might not be properly captured in this type of users.

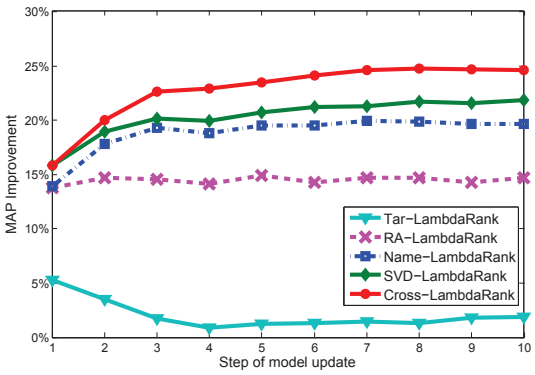
Adaptation Efficiency

The instance-based method, e.g., TransRank and IW-RankSVM, and feature-based method, e.g., CLRank, are infeasible to be applied in the per-user basis adaptation scenario, due to the need of numerously accessing the source-domain data during adaptation. Therefore, in this experiment, we will only compare with the model-based adaptation method, i.e., RA method. Besides, we also included the ranking model solely estimated on each user's adaptation data as a baseline. In particular, we will only illustrate the comparison results based on LambdaRank due to its superior ranking performance shown in Section 5.4.3. Since all the three methods share similar computational complexity, we evaluated their adaptation efficiency by varying the number of adaptation queries and examining which method can adapt to user's preference with fewer number of adaptation queries. To make the results comparable across different setting of adaptation queries, we selected a subset of users who have at least 15 queries in total, in which we fixed the last 5 queries in each user as testing queries. This gives us a collection of 2,743 users with 42,595 queries.

First, we gradually increased the size of adaptation queries from 1 to 10 in each user, and



(a) Batch adaptation for LambdaRank



(b) Online adaptation for LambdaRank

Figure 5.2: Adaptation efficiency comparisons.

re-estimated the adapted models every time accordingly. The relative improvement of MAP metric for all the methods against the global ranking model on the testing set are shown in Figure 5.2 (a).

As shown in Figure 5.2 (a), by leveraging knowledge from the global model, the adapted ranking models outperformed the model only estimated on the adaptation data; and, with only a small amount of adaption data, e.g., 1 or 2 queries, the adapted models can already achieve encouraging improvement (over 15%) against the global model. Comparing to RA method, our proposed adaptation method achieved more rapid improvement: *Cross*-LambdaRank achieved 25% improvement by using three queries, while RA-LambdaRank slowly climbed to 23% improvement even after ten queries. Such efficient adaptation credits to the transformation sharing across features in our framework, which helps the model better handle the sparsity problem during adaptation.

The settings in Figure 5.2 (a) simulate a situation in which batch update is performed, i.e., update the models for each user once we have collected sufficient adaptation data. However, in a more practical setting, we cannot wait too long to collect sufficient adaptation data, so that *online* updating is required. In this experiment, we used the same set of users as in Figure 5.2 (a), but updated the ranking models for every adaptation query we collected from the users in an online manner, in which we treated the previously updated model as the base model for the next iteration of model updating. The results are shown in Figure 5.2 (b).

We can clearly notice the advantage of our adaptation method against the baseline methods from the online adaptation results. Because Tar-LambdaRank cannot leverage any knowledge from the global model about the unseen features, its performance fluctuated due to the variance

of adaptation queries. Although RA-LambdaRank appeals to the global model for estimating the unseen features, in the online setting, the knowledge from global model get diminished as adaptation evolves, because it has to use the model from last iteration as regularization. As a result, its performance is worse than that in the batch mode (in Figure 5.2 (a)). In our method, observations in the adaptation data can be fully exploited via transformation sharing across features, so that unseen features can also get proper update during the online adaptation, which leads to consistent improvement of ranking performance in both batch and online settings.

5.5 Conclusions

In this chapter, we proposed a general ranking model adaptation framework for personalized search. A series of learned linear transformations, e.g., scaling and shifting, was performed onto the parameters of a generic ranking model on a per-user basis, such that the adapted model can better fit each individual user's search preferences. By sharing transformations across features in a group-wise manner, unseen features can also be properly updated given only limited number of adaptation queries. We instantiated the proposed framework with three frequently used learning-to-rank algorithms, i.e., RankNet, LambdaRank and RankSVM, which achieved significant improvement in, not only adaptation efficiency, but also adaptation accuracy, against several state-of-the-art ranking model adaptation methods in extensive experimentation.

In our current solution, the feature grouping function and transformation matrix are estimated independently. It would be meaningful to jointly estimate the two components for better adaptation performance. Besides, the proposed linear transformation based ranking model adaptation framework opens an interesting new direction for personalization: rich signals, e.g., user-specific profiles and features, could be included to affect the transformation in order to more fully reflect users' individual search interests.

Chapter 6

Modeling Action-level Satisfaction for Search Task Satisfaction Prediction

Search satisfaction is a property of a user's search process. Understanding it is critical for search providers to evaluate the performance and improve the effectiveness of search engines. Existing methods model search satisfaction holistically at the search-task level, ignoring important dependencies between action-level satisfaction and overall task satisfaction. In this chapter, we hypothesize that searchers' latent action-level satisfaction (i.e., whether they believe they were satisfied with the results of a query or click) influences their observed search behaviors and contributes to overall search satisfaction. We conjecture that by modeling search satisfaction at the action level, we can build more complete and more accurate predictors of search-task satisfaction. To achieve so, we develop a latent structural learning method, whereby rich structured features and dependency relations unique to search satisfaction prediction are explored. Using in-situ search satisfaction judgments provided by searchers, we show that there is significant value in modeling action-level satisfaction in search-task satisfaction prediction.

6.1 Introduction

Measuring search engine performance via behavioral indicators of search satisfaction has recently received considerable attention [2, 47, 50, 58]. In comparison with traditional relevance-based evaluations [9], such methods enable evaluation using real user populations, in naturalistic settings, and across a diverse set of information needs. It has been shown that users' search behaviors provide more accurate signals of search satisfaction than query-document relevance [50, 58].

The core problem in search-task satisfaction modeling is to understand whether users are satisfied with their search actions (i.e., whether they believe they were satisfied with the search results for a particular information need) when performing the task [2, 8, 43, 58]. Unfortunately, searchers'

detailed action satisfaction labels are *unobservable* in search log data; and they are difficult to obtain at scale from the searchers or reliably from third-party assessors. As a result, most prior search satisfaction models do not directly consider user satisfaction at the *action level*, or elect to only approximate that with specific assumptions. For example, most of existing methods consider search-task as the modeling unit, and extract holistic measures, such as total dwell time [47, 124] and search result clicks [50], to perform search satisfaction prediction. Other methods that consider action-level behaviors do not predict users' detailed satisfaction over those actions [2, 58, 59]. Instead they assume that all actions are satisfying in a satisfying task, and all actions are unsatisfying in an unsatisfying task. This masks the complex relationship between action-level satisfaction and overall search-task satisfaction: e.g., searchers can be ultimately satisfied by the search task, but most of her search actions might be quite unsatisfying [47]. Therefore, such a modeling assumption expropriates the model's ability to discriminate between different actions, i.e., satisfying vs. unsatisfying.

In this work, we hypothesize that users' perceived action-level satisfaction, even though unobservable in search logs, influences their observed search behaviors and contributes to overall search-task satisfaction. We conjecture that by modeling satisfaction at the individual action level, we can build more complete and more accurate predictors of search satisfaction. To achieve this, we consider the action-level user satisfaction as latent variables, and explicitly model their relationship to overall task satisfaction in a latent structural learning framework. By introducing the latent variables, expressive features and dependency relations unique to the search satisfaction problem can be incorporated to depict searchers' complex behavioral patterns. Knowledge about users' in-task search behaviors, e.g., consistency between action-level and overall task satisfaction, is naturally modeled in the proposed learning framework to guide satisfaction modeling.

Our research contributions can be summarized as follows:

- Explicitly model latent action-level satisfaction as part of search-task satisfaction modeling;
- Perform extensive experimental analysis of the proposed method whereby several state-of-the-art search satisfaction models are compared and significant performance improvement on different data sets is achieved;
- Demonstrate clear utility of the inferred action-level satisfaction labels by improved perfor-

mance in document relevance estimation and query suggestion.

6.2 Related Work

Recent advances in retrieval evaluation have focused on modeling search behaviors and exploiting implicit feedback [3, 70]. Qualitative studies showed that users' search behaviors are good indicators of retrieval system performance [101] and search-task difficulty [8]. Smith and Kantor found that users adapted their search behaviors to the deliberately degraded retrieval systems, e.g., increase the rate of query entry and decrease the occurrence of repeated queries [101]. Aula et al. reported that when facing with difficult search tasks, users tended to use more diverse queries and more advanced operations, and spend longer time on the search result pages [8]. Such studies shed lights on the potential of evaluating search performance via searchers' behaviors.

Satisfaction has been studied extensively in a number of areas such as psychology [85] and commerce [94]. In IR literature, search satisfaction is generally defined as the fulfillment of a user's information need [50, 58]. Fox et al. [50] used an instrumented browser to collect search activities and compared them against explicit user satisfaction judgments of full search sessions. They identified a strong association between users' search patterns and their explicit satisfaction ratings. Hassan et al. [58, 60] found that a user's search action sequence is sufficiently accurate to predict search satisfaction. Feild et al. [47] focused on the behavioral clues to detect search frustration, where various signals from query logs and physiological sensors were explored. In [75], Kim et al. introduced more sophisticated signals to calibrate click dwell time for better estimating click satisfaction.

Despite the wealth of research in this area, most prior studies regard search-tasks as the basic modeling unit, from which holistic measures, e.g, total dwell-time [124] and query-click ratio [47], are extracted for predicting search satisfaction. However, users' detailed *action-level* satisfaction was largely ignored in prior work, though it conveys important information about searchers' overall search satisfaction [8, 47]: searchers can be ultimately satisfied by the search task, but most of her search actions might be quite unsatisfying. Thus, methods that fail to consider satisfaction at the action-level may not be optimal for this prediction problem. To the best of our knowledge, Ageev et al.'s work in [2] was the first attempt to consider users' action-level satisfaction for

search-task satisfaction prediction. In their work, a controlled lab experiment is performed to track users' search activities during predefined search tasks. They approximated users' action-level satisfaction by using manual relevance judgments, and they identified distinct search paths among the satisfying/unsatisfying actions in the satisfying versus unsatisfying search tasks. Their study confirms our claim that it is necessary to distinguish and model users' action-level satisfaction in search-task satisfaction prediction. As their solution, a CRF model was adopted to predict search-task satisfaction based on a set of behavior features. However, because they asserted that all action labels equaled the task label, discrimination between different search actions was not possible. Therefore, their method is still unable to distinguish action-level satisfaction, as we do in this paper.

6.3 Problem Definition

In this section, we formally define the problem of search-task satisfaction prediction. We followed the definition of search task in Chapter 4, and we will assume such segmentation is given a priori in our task satisfaction prediction problem.

Specifically, the input of a search-task satisfaction prediction problem is a sequence of user u's search actions in a particular search task t_u, in which the actions are chronologically ordered, i.e., $A^{t_u} = \{a_1^{t_u}, \ldots, a_n^{t_u}\}$. We adopt the action type definition in [58], and consider the following types of search actions in this work:

- Q, issue a query to a search engine;
- SERP, hit BACK button to return to the search result page or refresh the search result page;
- PAGN, go to the next page of search results;
- SR, click on a returned document in search result page;
- BR, click on a hyperlink in the current document (not in a search result page);
- RL, click on a related search suggestion result;
- SP, click on the spelling correction link.

Each action a has an attribute $a.ref$ pointing to the previous action which leads to the current action.

The above action types cover most of the search actions a user typically performs during Web search. Additionally, to be consistent with our later description of the proposed method, we add two dummy actions into every search task, i.e., $a_0^{t_u} = \text{START}$ and $a_{n+1}^{t_u} = \text{END}$, indicating the start and end of a search task respectively. In particular, we denote $\mathcal{Q} = \{\text{Q}, \text{SERP}, \text{PAGN}, \text{RL}, \text{SP}\}$ as query-related actions, and $\mathcal{C} = \{\text{SR}, \text{BR}\}$ as click-related actions.

The output of a search-task satisfaction prediction problem is an overall satisfaction label y^{t_u} indicating whether the user u has been satisfied in the search task t_u. In this work, we follow Aula et al.'s criterion [8] to define search-task satisfaction as,

Definition (Search-Task Satisfaction) Given a user u's search task t_u, search-task satisfaction is a binary label y^{t_u}: $y^{t_u} = 1$, if the user's information need has been met and thus resulting a satisfying search task; otherwise $y^{t_u} = 0$.

In literature, there are different terms, e.g., "search success" [2, 58] and "frustration" [47], and perspectives, e.g., subjective [8, 58] or objective [34], used for defining a similar concept. We want to emphasize that our definition characterizes search satisfaction from a user's *subjective* perspective: a search-task is considered as satisfying, if, and only if, the searcher is satisfied with the search results and believes that they has found the answer (but the answer could be factually incorrect).

As a result, the problem of search-task satisfaction prediction is to estimate a function $f(\cdot)$ from the given search action sequence A^{t_u} to a search-task satisfaction label y^{t_u}, such that the predicted satisfaction label agrees with users' belief whether they have satisfied their information need.

Most of the previous approaches for search-task satisfaction prediction [2, 47, 50, 57, 58] fall into the above formalism. However, one important factor that has not yet been explicitly defined and explored in prior work is user u's satisfaction label $h_i^{t_u}$ related to a specific action $a_i^{t_u}$. Intuitively, $h_i^{t_u}$ characterizes the contribution of action $a_i^{t_u}$ towards user u's overall satisfaction of task t_u. Formally, we define a user's action-level satisfaction as,

Definition (Action-level Satisfaction) Action-level satisfaction $h_i^{t_u}$ is a binary outcome of a search action $a_i^{t_u}$ in task t_u, such that $h_i^{t_u} = 1$, if user u is satisfied with action $a_i^{t_u}$, e.g., found helpful information after clicking on a document; otherwise $h_i^{t_u} = 0$, e.g., a query action does not lead to any useful document.

It is worthwhile to note that despite defining $h_i^{t_u}$ as binary in the above definition, the potential label space for the variable $h_i^{t_u}$ is quite flexible, e.g., encoding it with multi-level ordinal labels to reflect users' complex information seeking behaviors (e.g., query refinement [8], exploring related information [123]). Our proposed method can be easily extended to the multi-label setting. In this work, we will follow this binary definition for simplicity and explicability.

6.4 Method

In this section, we describe the proposed latent structural model for search-task satisfaction prediction. We start with a real search task example to illustrate the necessity of modeling searchers' action-level satisfaction. Then we discuss our hypothesis about users' search behaviors with respect to action-level satisfaction. Based on such hypothesis, rich structured features and dependency relations unique to search-task satisfaction modeling are devised. In the end, we discuss how to incorporate domain-knowledge to guide the proposed model in learning the latent structures effectively.

6.4.1 Motivating Example

Table 6.1 presents a real example of a satisfying search task extracted from Ageev et al.'s public search data set [2]. We applied several state-of-the-art search satisfaction models, including Markov Model Likelihood (MML) method [58], logistic regression (LogiReg) model [47] and session-CRF model [2], and our proposed method on it. In particular, the MML and logistic regression method take a holistic view to directly predict task-level satisfaction, while the session-CRF and our method consider action-level satisfaction in the task. We trimmed the URLs of clicked documents to its domain in the table. The action-level predictions from session-CRF model (denoted as "CRF") and our method (denoted as "Ours") are illustrated in the last two columns of the table.

In this example, the searcher sought information on metals that can float on water. She rated this task as satisfying because she claimed the answer had been found after search. But it does not mean that she was satisfied with every action in the task. As we can observe, she first attempted three queries on Google, but was not satisfied with the search results: she kept reformulating the queries, spent a very short time on the clicked documents, and switched to Bing with the same

Table 6.1: Example of a satisfying search task. '+'/'-' indicates a predicted satisfying/unsatisfying action.

Search Actions	Engine	Time	CRF	Ours
Q: metals float on water	Google	10s	-	-
SR: wiki.answers.com		2s	-	-
BR: blog.sciseek.com		3s	-	-
Q: which metals float on water	Google	31s	-	-
Q: metals floating on water	Google	16s	-	-
SR: www.blurtit.com		5s	-	-
Q: metals floating on water	Bing	53s	-	-
Q: lithium sodium potassium float on water	Google	38s	-	+
SR: www.docbrown.info		15s	-	+

query. After spending quite some time on Bing's search result page, she issued a very specific query to Google and reached the correct answer (the answer was verified by a human editor).

Models based on task-level implicit measures, i.e., MML and LogiReg, mistakenly predicted that the searcher was unsatisfied with the task: dwell times on the clicked documents were generally short, along with a number of query reformulations and search engine switches. Due to the restrictive assumption in Ageev et al.'s session-CRF model, i.e., all actions have to be satisfying in a satisfying task, it made a wrong prediction for this task as well, since most actions were unsatisfying. But once we consider the searcher's action-level satisfaction, as predicted in our method's output, we could reach the correct conclusion that the task is satisfying. From this example, we can clearly realize the importance of recognizing a user's fine-grained satisfaction at action level for search-task satisfaction prediction.

6.4.2 Hypothesis and the AcTS Model

As was discussed in our motivating example in Table 6.1, the action-level satisfaction labels H[1] convey informative clues about overall search-task satisfaction. If H is known, sophisticated features about users' perceived satisfaction of search activities can be extracted, e.g., examining if the task ends with a satisfying action or measuring the ratio of time spent on satisfying actions versus unsatisfying ones, for better predicting the overall task satisfaction label y. Unfortunately, H is hidden in search log data; and it is also quite challenging to be manually annotated at scale.

[1]When no ambiguity is invoked, we will discard the user index u and task index t^u to simplify the notations.

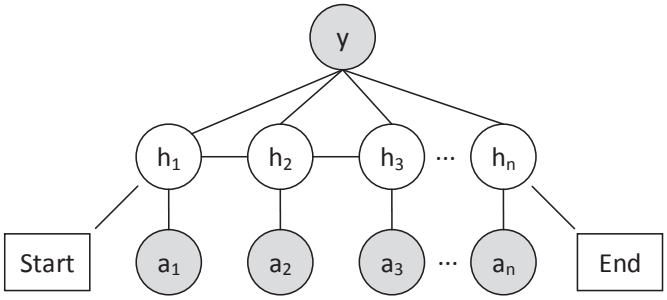


Figure 6.1: Structural dependency assumptions about a user's search behaviors postulated in AcTS model. Light circles represent latent variables and shadow circles represent observable variables. Lines indicate possible dependencies between the variables (the dependency between y and a is not shown to make the representation concise). In AcTS, a joint mapping of $f(A) \rightarrow H \times y$ is estimated.

This prevents previous works from directly utilizing such information for search-task satisfaction prediction.

To address this challenge, we devise a basic hypothesis about users' search behaviors:

Hypothesis. *The desire for satisfaction drives users' interaction with search engines and that the satisfaction attained during the search-task contributes to the overall satisfaction.*

This hypothesis makes two assumptions. First, users' overall search-task satisfaction depends on their satisfaction with the performed search actions, e.g., if all actions were satisfying, it is very likely that the user would end up with a satisfying search task. Second, users' search actions are mutually dependent via the latent action satisfaction labels. For example, if a query is unsatisfying, e.g., it is later resubmit to another search engine [56], the result clicks in the search engine's result page of first query can hardly be satisfying.

We consider H as latent variables and realize our hypothesis about a user's search behaviors in a structured prediction model. We name the proposed method as Action-aware Task Satisfaction (AcTS) model, and describe the structural dependencies imposed in AcTS model in Figure 6.1.

To formally encode the dependency assumptions in our hypothesis, we define a feature vector for the task satisfaction label y specified by the search action sequence A and corresponding hidden action satisfaction labels H as $\Phi(A, H, y)$. Based on this feature representation, AcTS predicts the

search-task satisfaction at testing time by,

$$(\hat{y}, \hat{H}) = \underset{(y,H)\in\mathcal{Y}\times\mathcal{H}}{\arg\max}\ w^{\mathsf{T}}\Phi(A, H, y). \tag{6.1}$$

In Eq (6.1), $\mathcal{Y}$ and $\mathcal{H}$ represent the sets of all possible values of y and configurations of H respectively. w is the parameter vector in our AcTS model, and it reflects the relative importance of features in predicting search-task satisfaction. In this paper, we refer to solving Eq (6.1) as the inference problem. In the solution of our inference problem, $\hat{y}$ becomes the output for the task-level satisfaction prediction and $\hat{H}$ is the inferred action-level satisfaction labels for the input search action sequence.

The inference problem of Eq (6.1) clearly distinguishes the proposed AcTS model from all the prior search satisfaction models. In order to make a prediction of the overall search satisfaction label y, we need to determine the latent action satisfaction labels H as well, which are mostly consistent with the observations in the input search actions A and support the predicted overall satisfaction label y in task t. Formally, we are estimating a joint mapping from input search action sequence A to task satisfaction label y and latent action satisfaction labels H, i.e., $f(A) \rightarrow H \times y$; while most prior works only estimate a binary mapping of $f(A) \rightarrow y$. Moreover, in the proposed AcTS model, a user's search actions A are no longer treated as independent, but instead, they are modeled as being correlated with each other via the latent action satisfaction labels H. As a result, expressive features about a user's search behaviors can be designed, such as measuring the transition between a user's satisfying and unsatisfying search actions and examining whether a user is satisfied with all the query actions.

More importantly, the inferred action-level satisfaction labels H not only provide informative signals for determining overall search satisfaction, but also reveal the utility of those actions towards a user's information need. For example, based on the identified labels in H, we can easily recognize which clicked document is helpful in satisfying a user's information need, and which query leads to the helpful documents. The estimated utilities are beneficial for a variety of search applications, e.g., document relevance estimation and query suggestion. Nevertheless, such information is not available in any of the existing search satisfaction models.

In the following, we will discuss in detail about our design of the structured features $\Phi(A, H, y)$

in Section 6.4.3, and the use of domain knowledge for learning the optimal feature weights w in Section 6.4.4.

6.4.3 Structured Features

Table 6.2: Structured behaviorial features for search-task satisfaction modeling in AcTS.

Type	Feature	Description
Short-range	$\phi_{switch}(y, A, h_i)$	if the user switches search engine after this query action
	$\phi_{reform}(y, A, h_i)$	edit distance between two consecutive queries
	$\phi_{question}(y, a_i, h_i)$	if the query is a question
	$\phi_{stopword}(y, a_i, h_i)$	proportion of stopwords in query
	$\phi_{rel}(y, a_i, h_i)$	query term matching in URL string of a_i
	$\phi_{pos}(y, a_i, h_i)$	display position of the clicked URL
	$\phi_{last}(y, h_n)$	if T ends up with a satisfying search action
Long-range	$\phi_{trans}(y, h_i, h_{i+1}, a_i, a_{i+1})$	first order transition between actions with respect to satisfaction labels
	$\phi_{allQ}(y, H, A)$	if all the query-related actions in T are satisfying
	$\phi_{existQ}(y, H, A)$	if there exists a satisfying query-related action in T
	$\phi_{allC}(y, H, A)$	if all the click-related actions in T are satisfying
	$\phi_{existC}(y, H, A)$	if there exists a satisfying click-related action in T

Previous work has developed a wide variety of behavioral features for search satisfaction prediction [2, 47, 50, 56, 124]. All of those features can be flexibly applied in our AcTS model. However, since most prior works only estimate a holistic mapping of $f(A) \rightarrow y$, their employed features (e.g., total dwell time [124] and number of result clicks [50]) cannot capture a user's action-level satisfaction. In this section, we focus on the newly developed structured features for AcTS, in which expressive signals about the dependency among search actions A, action-level satisfaction labels H and task-level satisfaction label y is explicitly explored via the latent variables. The devised features can be categorized into two classes: short-range features (specifying satisfaction label for a single action in task t) and long-range features (specifying satisfaction labels for a set of actions in task t).

• **Short-range features:** As shown in Figure 6.1, in our AcTS model, the features extracted from action a_i are directly used to predict the corresponding satisfaction label h_i and overall task satisfaction label y (i.e., $f(A) \rightarrow H \times y$). This is distinct from the features explored in most

existing search satisfaction models, where the action-level observations are aggregated to determine the task-level satisfaction label y [47, 50, 56].

In a user's query-related actions, although not especially common, search engine switching (i.e., the voluntary transition between different search engines) usually indicates searcher frustration [56]. We encode this as $\phi_{switch}(y, A, h_i) = \delta(y, a_i, h_i)\delta(a_i.Engine \neq a_j.Engine)$, where a_j is the next query action following the current query action a_i and $\delta(\cdot)$ is an indicator function. Similarly, query reformulation also indicates the user is not satisfied with the search results of the current query [8]. We formalize this by measuring the similarity between two consecutive queries: $\phi_{reform}(y, A, h_i) = \delta(y, a_i, h_i)sim(a_i.Query, a_j.Query)$, where a_j is the query action following the current query action a_i, and $sim(X, Y)$ is the edit distance between query string X and Y. Besides, we also examine if the query is in a question form by $\phi_{question}(y, A, h_i)$ and calculate the proportion of stopwords in the query by $\phi_{stopword}(y, A, h_i)$ to estimate satisfaction for the query-related actions.

Among a user's click-related actions, the relevance quality of a clicked document to the given query can be an important criterion to measure user satisfaction [64]. Because we do not assume the availability of document content in our problem since that is usually unavailable in search logs, we can only measure relevance of the clicked documents according to their URL strings. In particular, we define $\phi_{rel}(y, A, h_i) = \delta(y, a_i, h_i)c(a_i.URL, a_k.Query)$, where $c(URL, Query)$ counts the number of query terms occurred in the URL string, and a_k is the query action that leads to the current click action a_i. In addition, the original rank position of the clicked URL in search-result page is also a good indicator of its relevance quality [64]. We encode it as $\phi_{pos}(y, a_i, h_i) = \delta(y, a_i, h_i)a_i.Pos$.

Besides, previous studies have demonstrated that a user's last search action is closely related to her search-task satisfaction [50]. We encode this as $\phi_{last}(y, h_n) = \delta(y = h_n)$, i.e., examine whether the satisfaction label of the user's last action agrees with her overall task satisfaction.

• **Long-range features:** We devise the first order transition feature $\phi_{trans}(y, h_i, h_{i+1}, a_i, a_{i+1})$ to capture a user's sequential search behaviors with respect to the latent action satisfaction labels. For example, in a satisfying search task, an unsatisfying query is more likely to be reformulated into a satisfying query rather than another unsatisfying one. In particular, we define $\phi_{trans}(y, h_i, h_{i+1}, a_i, a_{i+1}) = \delta(y = y', h_i = h', h_{i+1} = h'', a_i = a', a_{i+1} = a'')$, where (y', h', h'', a', a'') takes all the possible values for task satisfaction label, action satisfaction labels and action types.

We should note that our transition features are different from those introduced in [57, 58]: in that work, only the transitions between different action types are modeled, e.g., from Q to SR; while in our model, we distinguish search action transitions with respect to the latent action satisfaction labels, e.g., from satisfying Q to satisfying SR.

Beyond exploring the behaviorial patterns within adjacent search actions, a set of long-range features are introduced to capture dependency at the whole task level by examing: I. if all the query-related actions are satisfying: $\phi_{allQ}(y, H, A) = \delta(\sum_{a_i \in \mathcal{Q}} h_i = \sum_{a_i \in \mathcal{Q}} 1)$; II. if there exists a satisfying query action: $\phi_{existQ}(y, H, A) = \delta(\sum_{a_i \in \mathcal{Q}} h_i > 0)$; III. if all the click-related actions are satisfying: $\phi_{allC}(y, H, A) = \delta(\sum_{a_i \in \mathcal{C}} h_i = \sum_{a_i \in \mathcal{C}} 1)$; and, IV. if there exists a satisfying click: $\phi_{existC}(y, H, A) = \delta(\sum_{a_i \in \mathcal{C}} h_i > 0)$.

We need to emphasize that the above long-range features can only be exploited by our AcTS model, since it explicitly models the users' action-level satisfaction across different actions in a search task. None of existing methods can utilize such information for search-task satisfaction prediction.

In addition to the above newly introduced structured features, we also included the action-level behavior features from [2] and [50] in our AcTS model, such as action dwell time and query-click ratio. The list of features[2] used in this work appears in Table 6.2.

6.4.4 Training AcTS with Weak Supervision

Because the ground-truth labels for a user's action-level satisfaction are unobservable in the search log data, we have no direct supervision to guide the model in learning about such latent structures. Fortunately, there is plenty of work in cognitive science and information science exploring users' search behaviors and strategies in performing a successful search task [8, 93, 123]. Such studies shed light on the insights of users' detailed in-task search behavior patterns. In this section, we propose the use of *structured loss functions* [25] to inject such domain knowledge as weak supervision for AcTS training (i.e., learning the weight vector w in Eq (6.1)).

To regularize the training of AcTS model with domain knowledge, we derive our learning algorithm for AcTS model from the latent structural SVMs framework [25]. For a given set of

[2]Details of features from [2, 50] are not listed in the table.

search tasks with only task-level search satisfaction labels, i.e., $\{(A_m, y_m)\}_{m=1}^{M}$, AcTS training is formalized as the following optimization problem:

$$\min_{w,\xi} \ \frac{1}{2}||w||^2 + C\sum_{m=1}^{M}\xi_m^2 \tag{6.2}$$
$$s.t. \forall m, \max_{H\in\mathcal{H}} w^{\mathsf{T}}\Phi(A_m, H, y_m) \geq \max_{(\hat{y},\hat{H})\in\mathcal{Y}\times\mathcal{H}} [w^{\mathsf{T}}\Phi(A_m, \hat{H}, \hat{y}) + \Delta(y_m, \hat{y}, \hat{H}, A)] - \xi_m.$$

In Eq (6.2), $\Delta(y_m, \hat{y}, \hat{H}, A)$ measures the distance between the predicted labels $(\hat{y}, \hat{H})$ and the ground-truth (y_m, H_m^*), where H_m^* is the unobservable ground-truth of action-level satisfaction labels. $\{\xi_m\}_{m=1}^{M}$ is a set of slack variables to allow errors in the training data, and C controls the trade-off between empirical training loss and model complexity.

$\Delta(y_m, \hat{y}, \hat{H}, A_m)$ indicates the prediction error between $(\hat{y}, \hat{H})$ and (y_m, H_m^*); and thus it drives model learning. As H_m^* is unknown in the training data, we have no supervision to guide AcTS model in learning about such latent structures. As our solution, weak supervision about users' search behaviors is injected via the design of $\Delta(y_m, \hat{y}, \hat{H}, A_m)$. Intuitively, we should increase $\Delta(y_m, \hat{y}, \hat{H}, A_m)$, i.e., penalize the prediction, when the inferred $\hat{H}$ contradicts our knowledge about a legitimate configuration of H. In this work, we define a set of structured loss functions $\sigma(\hat{y}, \hat{H}, A)$ to realize the knowledge about $\hat{H}$ in $\Delta(y_m, \hat{y}, \hat{H}, A_m)$ from different perspectives.

First, a good configuration of $\hat{H}$ has to be consistent with the predicted overall search-task satisfaction label $\hat{y}$. We measure this by:

$$\sigma_{sat}(\hat{y}, \hat{H}) = \begin{cases} 1 & \hat{y} = 1, \ \sum_{\hat{h}_i\in\hat{H}} \hat{h}_i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

i.e., all the actions should not be unsatisfying in a satisfying task. And, vice versa,

$$\sigma_{dsat}(\hat{y}, \hat{H}) = \begin{cases} 1 & \hat{y} = 0, \ \sum_{\hat{h}_i\in\hat{H}} \hat{h}_i = \sum_{\hat{h}_i} 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

Second, the configuration of $\hat{H}$ itself should be consistent. For example, an unsatisfying query

cannot result in any satisfying search-result clicks [2], i.e.,

$$\sigma_{clk}(\hat{H}, A) = \begin{cases} 1 & \text{exist } a_i = Q, a_j = SR, a_j.\text{ref} = a_i, \hat{h}_i < \hat{h}_j \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

And when the user performs duplicated actions in the same task, e.g., submit the same query twice to the same search engine, their inferred satisfaction labels should be the same,

$$\sigma_{dup}(\hat{H}, A) = \begin{cases} 1 & \text{exist } a_i = a_j, \hat{h}_i \neq \hat{h}_j \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

The suggested query from a search engine's spelling correction, e.g., correcting the misspelt query "amazone" into its correct form "amazon," should not hurt user satisfaction,

$$\sigma_{sp}(\hat{H}, A) = \begin{cases} 1 & \text{exist } a_i = Q, a_j = SP, a_j.\text{ref} = a_i, \hat{h}_i > \hat{h}_j \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

Based on the above estimated distance between $\hat{H}$ and H^*_m, we can define the margin in Eq (6.2) as,

$$\Delta(y_m, \hat{y}, \hat{H}, A_m) = \delta(y_m = \hat{y}) + \textstyle\sum_i \lambda_i \sigma_i(\hat{y}, \hat{H}, A_m). \tag{6.8}$$

where λ_i is a trade-off between task-level 0/1 loss and action-level loss defined by the structured loss functions $\sigma(\hat{y}, \hat{H}, A)$ as described in Eq (6.3) to Eq (6.7).

The margin function defined above encodes the knowledge about a user's latent action-level satisfaction labels within a search task as week supervision for latent structure learning [25]. It bridges the qualitative studies of users' search behaviors [8, 93, 123] and quantitative modeling approaches. We should note that the structured loss functions $\sigma(\hat{y}, \hat{H}, A)$ might be violated in a particular user's real search actions, and λ_i controls our confidence of such loss functions.

The optimization problem in Eq (6.2) can be efficiently solved by the iterative algorithm proposed in [25]. One thing we should note is that due to the long-range dependency introduced by the structured features proposed in Section 6.4.3, e.g., $\phi_{allQ}(y, H, A)$ and $\phi_{existQ}(y, H, A)$, the inference problems defined in Eq (6.1) and Eq (6.2) become computationally intractable. We address these

inference problems via integer linear programming (ILP), which will be discussed in Section 6.4.5 with more details.

6.4.5 Inference Algorithm

The inference problem defined in Eq (6.1) becomes computationally intractable, due to the long-range dependency introduced by the structural features proposed in Section 6.4.3, e.g., $\phi^t_{allQ}(y, H, A)$ and $\phi^t_{existQ}(y, H, A)$. We apply the method from [77] to effectively address the inference problem via integer linear programming (ILP).

To convert our inference problem into an ILP problem, we introduce a set of auxiliary variables $q(x^j_c)$, where c enumerates every set of connected variables in the AcTS model, i.e., cliques in Figure 6.1, and j enumerates all the possible assignments of variables in clique c. To distinguish the newly introduced auxiliary variables, we refer to the variables of interest in AcTS model (i.e., y and H) as original variables. For example, eight auxiliary variables $\{q(x^1_{c_0}), q(x^2_{c_0}), \ldots, q(x^8_{c_0})\}$ are introduced for the clique $c_0 = (h_0, h_1, y)$ in AcTS, corresponding to eight different assignments of original variables in the clique c_0. These auxiliary variables take binary values, and $q(x^j_c) = 1$ if and only if the original variables in clique c take the assignment specified by j. For example, $q(x^3_{c_0}) = 1$ indicates $h_0 = 1$, $h_1 = 1$ and $y = 0$ (original variables are enumerated lexicographically). Then, we define a new weight vector $\boldsymbol{\beta}$, whose length equals the total number of auxiliary variables. Each dimension of $\boldsymbol{\beta}$, i.e., β^j_c, corresponds to a particular auxiliary variable $q(x^j_c)$. The value of β^j_c represents the contribution of clique c taking the specific assignment j to the objective function defined in Eq (6.1).

Based on the auxiliary variables $q(x^j_c)$ and new weight vector $\boldsymbol{\beta}$, we defined the ILP problem as,

$$\begin{aligned} \max \quad & \sum_{c,j} \beta^j_c q(x^j_c) \\ s.t. \quad & \forall c, c', j, \quad q(x^j_c) \in \{0, 1\}, \quad \textstyle\sum_j q(x^j_c) = 1 \\ & \sum_{k \in \mathrm{Val}(c/s_{c,c'})} q(x^k_c) = \sum_{l \in \mathrm{Val}(c'/s_{c,c'})} q(x^l_{c'}) \end{aligned} \tag{6.9}$$

where $s_{c,c'}$ is the intersection of variables in clique c and c', $c/s_{c,c'}$ is the set of variables in c but

not in $s_{c,c'}$, and Val(c) is the enumeration of all the possible assignments to the variables in set c.

In this ILP formulation, the first equality constraint enforces the mutual exclusivity of assignments to the original variables within a clique. The second equality constraint enforces the consistency of assignments to the original variables shared by two different cliques. Readers can refer to [77] for more details about this ILP transformation.

Once the ILP problem is solved, optimal solution of the original inference problem in Eq (6.1) can be immediately decoded from auxiliary variables. And based on this ILP reformulation, the two additional inferences in AcTS training can be effectively solved in a similar way. To perform the loss-augmented inference, we need to include the margin term $\Delta(y_m, \hat{y}, \hat{H}, A_m)$ into the object function of Eq (6.9). And to perform the latent variable completion inference, we only need to fix $y = y_m$ and all the rest procedures stay the same as in Eq (6.9).

The above ILP problems might be massive given the potentially large number of variables involved in a clique, e.g., all the query actions in the same task will be connected via the feature $\phi^t_{allQ}(y, H, A)$. But considering the average size of real search tasks, e.g., on average 5 to 7 actions per task in our evaluation data sets, the exact solution of ILP can still be effectively found with off-the-shelf ILP solvers.

6.5 Experiments

In this section, we first quantitatively evaluate the effectiveness of the proposed AcTS model that models users' action-level satisfaction as latent variables, whereby several state-of-the-art search satisfaction models are compared over the in-situ task satisfaction labels from previous studies [2, 59]. Then we assess the quality of the inferred action-level satisfaction labels via their utilities in facilitating other information retrieval studies, where understanding users' detailed action-level satisfaction is important.

6.5.1 Data Sets

Hassan et al. [59] developed a toolbar plugin for the Internet Explorer browser to collect search activities and explicit search satisfaction ratings from the searchers. The authors explicitly asked the searchers to rate their search tasks immediately upon termination. This data set provides

Table 6.3: Basic statistics of evaluation data sets.

Data set	# User	# Task	Action/Task	T^+:T^-
toolbar	153	7306	5.2(±6.6)	6.84:1
contest	156	1487	6.2(±5.9)	6.70:1
search log	2,406,841	7,650,047	7.1(±11.8)	-

reliable first-hand annotation of search-task satisfaction. We refer to this as "toolbar data" in our experiments.

Ageev et al. [2] designed a game-like online contest for crowdsourcing search behavior studies. In their study, users were required to perform several predefined informational tasks via a Web search interface and submit the answers they found to the system. All users' search behaviors were logged and annotated by the authors. According to our search-task satisfaction definition described in Section 6.3, we treat the tasks in which the user has submitted an answer as satisfying (the answer might be incorrect with respect to the predefined information need). We refer to this as "contest data" in our experiments.

To investigate the utility of the proposed method in predicting search satisfaction in real-world search engine logs, we extracted large-scale query logs sampled from a commercial Web search engine. In a four-month period, from December 2012 to March 2013, a subset of users were randomly selected. The search logs recorded their search activities, including the anonymized user ID, query string, timestamp, returned URL sets and the corresponding user clicks. These logs were segmented into search tasks by the method developed in Chapter 4. This data set does not contain task-level nor action-level satisfaction labels. We refer to this as "search log data," and describe its usage in Section 6.5.3.

Basic statistics of these data sets appear in Table 6.3.

6.5.2 Search-Task Satisfaction Prediction

To investigate the effectiveness of modeling users' action-level satisfaction as latent variables in AcTS model, we first quantitatively compare the performance of the proposed model with several state-of-the-art methods in predicting overall search-task satisfaction.

Baselines

Several methods have been proposed to predict search satisfaction based on users' search behaviors [2, 47, 57, 58]. We adopt several best-performing models from the previous works as our baseline methods.

Hassan et al. [58] proposed a Markov Model Likelihood (MML) method to predict search satisfaction. In MML, two sets of first order transition probabilities are estimated from the search action trails in satisfying and unsatisfying tasks. At testing time, MML calculates the likelihood ratio of an input search action sequence between the satisfying and unsatisfying models to determine the task satisfaction label. We followed the specification of MML in [57] to implement the model (they used the same set of action types as ours). Maximum a posterior estimator with Dirichlet priors is used to estimate the transition probabilities in MML. To model click dwell time in MML, we add two new action types, SR_long and BR_long, which represent the click actions (SR and BR) with dwell time longer than 30s, in all the methods in our experiment.

Feild et al. [47] used a logistic regression model to predict search frustration, where features extracted from both query logs and physiological sensors are employed. We built a logistic regression model based on the features described in Section 6.4.3. The short-range features are aggregated in each task by action type, e.g., average the click position features $\phi_{pos}(y, a_i, h_i)$ over all SR actions in the same task. The long-range features, e.g., $\phi^t_{allQ}(y, H, A)$, are not included, since logistic regression cannot handle latent variables. We refer to this method as "LogiReg."

Ageev et al. proposed a session-CRF model [2] to predict search-task satisfaction. Although search actions were explicitly modeled, they asserted that action-level satisfaction labels equaled to the task-level label. Mathematically, this assumption makes their session-CRF degenerate to a logistic regression model. This obscures the complex dependency between task satisfaction and detailed action satisfactions in session-CRF. As a result, it cannot as effectively model the action-level user satisfaction as our model does. We took the same implementation of session-CRF as used in [2].

Table 6.4: Search task success prediction performance on the toolbar data set.

	avg-f_1	T^+-f_1	T^--f_1	Accuracy
MML	0.707	0.897	0.518	0.830
LogiReg	0.740	0.918	0.563	0.861
session-CRF	0.728	0.910	0.545	0.850
AcTS	**0.761***	**0.938***	**0.584***	**0.893***
AcTS_0	0.739	0.924	0.554	0.868

$^*p$-value$<$0.05

Table 6.5: Search task success prediction performance on the contest data set.

	avg-f_1	T^+-f_1	T^--f_1	Accuracy
MML	0.658	0.901	0.414	0.831
LogiReg	0.682	0.930	0.435	0.875
session-CRF	0.685	0.921	0.449	0.862
AcTS	**0.701***	**0.934**	**0.469***	**0.882**
AcTS_0	0.687	0.925	0.449	0.868
labeled-AcTS	0.649	0.945	0.352	0.899

$^*p$-value$<$0.05

Effectiveness of the Latent Structure Model

As illustrated in Table 6.3, the distribution of task satisfaction labels in both toolbar and contest data are highly unbalanced: about 85% of the tasks are labeled as satisfying. In such an unbalanced data set, accuracy alone is inadequate to compare the performance of different methods. In our evaluation, we compute f_1 scores for both satisfying (T^+-f_1) and unsatisfying tasks (T^--f_1). Following the metric used in [2], we also report the average f_1 between T^+-f_1 and T^--f_1. In order to avoid bias introduced by training/testing split, we performed five-fold cross-validation in each method by sampling tasks into different folds, and repeated it three times with different random seeds. As a result, we report the average performance of all methods from 15 different trials on the toolbar and contest data sets in Table 6.4 and Table 6.5. Paired two sample t-test is performed to validate the statistical significance of the improvement from AcTS model against the best-performing baseline, LogiReg, under each of the performance metrics. In particular, we set the trade-off parameters λ_i to one in Eq (6.8) for AcTS model in all our experiments.

We can clearly observe the significant improvement from the proposed AcTS model over all baselines in both data sets. MML, which only models the sequential patterns in a user's search actions, performed the worst among all the methods. This indicates that a user's sequential search

behaviors alone are insufficient to capture the overall search satisfaction. session-CRF behaved similarly as LogiReg. Although action-level labels are explicitly modeled in session-CRF, its restrictive assumption about the labels degrades the model's capability in distinguishing the action-level satisfaction labels, e.g., unsatisfying actions will not be allowed in a satisfying task in session-CRF. We accredit the encouraging performance improvement of the proposed AcTS model to its unique capability of modeling the action satisfaction labels as latent variables. By explicitly modeling a user's action-level satisfaction, AcTS can naturally include all the signals used in the baseline methods and explore richer structured information, as specified in our long-range features, which cannot be handled in any baseline method.

Besides exploring more expressive structured features for search-task satisfaction prediction, another unique advantage of modeling the action-level satisfaction labels as latent variables in AcTS is to incorporate domain-knowledge for model training via the structured loss functions. To investigate this aspect, we test a special setting of AcTS, in which we set the trade-off parameters λ_i to zero in Eq (6.10). As a result, we are training a AcTS model with only task-level supervision. We name this model as AcTS_0 and include its performance on both data sets in Table 6.4 and Table 6.5.

Without the structured loss functions, AcTS's performance dropped significantly: it performed similarly as the LogiReg and session-CRF baselines. The reason is that the task-level satisfaction label alone cannot guide the model in learning about the latent structures of H. As a result, the inferred labels of H in AcTS_0 becomes arbitrary, and provides little help in predicting task satisfaction. This result confirms the need to explicitly model the *dependency* between action-level and task-level satisfaction in search satisfaction modeling.

In addition, since action-level manual annotations are available in the contest data set, we can treat those labels as "ground-truth" action satisfaction labels, and train our AcTS model with a known structure. To incorporate these labels into AcTS training, we define a new margin for Eq (6.2) as,

$$\Delta(y, \hat{y}, H^*, \hat{H}) = \delta(y = \hat{y}) + \textstyle\sum_i \delta(h_i^* = \hat{h}_i) \tag{6.10}$$

i.e., we are computing the Hamming distance between two labeled search sequences. We name this new model as labeled-AcTS, and list its performance in Table 6.5.

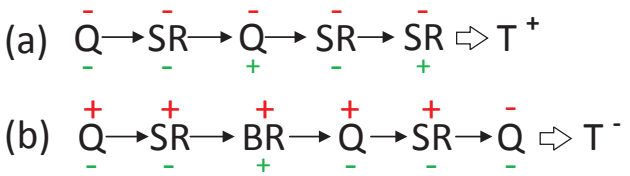


Figure 6.2: Case study of two manually annotated search sequences in the contest data set. Red labels on top of each action are the editor's annotations from [2], and green labels at the bottom are AcTS's predicted labels. T^+ and T^- indicate the task satisfaction labels provided by the users.

Surprisingly, the labeled-AcTS model did not outperform the original AcTS model with latent structures; and it performed significantly worse when predicting the unsatisfying tasks. To analyze the degraded performance, we examined the annotated search tasks in this data set and found many disagreements between the editor's judgements and searchers' actual behaviors. The discrepancy mainly stems from the inconsistent criteria between third-party editors and real users; and to understand it, we illustrate two typical inconsistent search sequences in Figure 6.2.

In Figure 6.2(a), all the clicked documents are judged to be irrelevant for this task. However the searcher still rated the task as satisfying. A reasonable explanation is that the searcher believed that she had found the correct answer, so was satisfied. While in Figure 6.2(b), according to the editor's judgment, the searcher has already issued several good queries and found the relevant documents for the question. However, the user rated it as unsatisfying in the end. The reason might be that the user did not notice the relevant passage(s) in the clicked documents, so was not satisfied with all of her search actions. The main reason for such disagreements is the annotation criterion devised in this data set [2]: Ageev et al. labeled a URL as a good URL if it contains correct answer to the predefined question in the search task; and a query is judged to be good if it leads to a good URL in its search result page. Nevertheless, from a real searcher's perspective, because she does not have any knowledge about the questions beforehand, she cannot fully judge the helpfulness of search results in an objective way. As a result, the editor's objective judgments in this data set cannot precisely reflect a user's perceived satisfaction during search, which, however, is the goal of prediction in our task satisfaction prediction problem. Such discrepancy explains the degraded performance of labeled-AcTS: in Eq (6.10), we overly penalized the predictions in model training due to the inappropriate manual annotations.

Meanwhile, as shown in Figure 6.2, the inferred action labels from AcTS better aligned with

the final task satisfaction labels in both cases: in Figure 6.2(a), the last query and last click action are predicted as satisfying; and in Figure 6.2(b), most of the actions are predicted as unsatisfying. Those inferred labels are more consistent with our above hypothetical analysis of users' search behaviors.

6.5.3 Action-Level Satisfaction Modeling

As an output of our structured inference problem defined in Eq (6.1), the inferred action-level satisfaction labels H have shown their ability in helping to predict the overall search-task satisfaction. Nevertheless, because the ground-truth labels of such output are not available in our evaluation data sets, we cannot directly evaluate the quality of the predicted action-level labels. In this section, we assess the quality of the inferred action satisfaction labels via their utilities in facilitating other information retrieval applications, where understanding users' action-level satisfaction is important.

Document Relevance Estimation

Accurately interpreting users' clickthroughs and extracting relevance signals for search engine optimization is an important topic in IR studies [3, 43]. The action-level satisfaction labels from AcTS can serve as a proxy to estimate the utility of clicked documents. In this section, we evaluate how the estimated document relevance can be used to improve the training of general learning-to-rank algorithms.

We chose LambdaMART [18] as our base learning-to-rank algorithm, and evaluate its ranking performance improvement by adding the features derived from AcTS model's output. A large set of manually annotated query-URL pairs are collected to create the evaluation data set. In this annotation set, each query-URL pair is labeled with a five-point relevance score, i.e., from 0 for "bad" to 4 for "perfect." And each pair is represented by a set of 398 standard ranking features, e.g., BM25, language model score and PageRank. We refer to this collection as the "annotation set."

In this experiment, we train an AcTS model based on all the search tasks in the toolbar data, and apply the learned model on the four-month search log data. We group the inferred satisfaction

labels under each unique query-URL pair, and calculate the corresponding median, mean and standard deviation as the additional relevance features derived from AcTS. To reduce noise in this estimation, we ignore the query-URL pairs occurred less than five times in this corpus. In the end, we joined the query-URL pairs extracted from the search log data with those in the annotation set, and obtained 3,311 annotated queries and 128,120 query-URL pairs with additional features derived from AcTS.

The same feature generation strategy is applied to the session-CRF's output. However, the MML and LogiReg baselines are not directly applicable in this evaluation, since they cannot make predictions of individual search actions. To compare with them, we followed Hassan et al.'s method [59] to estimate document utility based on the predicted overall task satisfaction labels. In their method, the utility of a clicked document is assumed to be proportional to its dwell time. To distinguish document utilities between satisfying and unsatisfying tasks, they separated such scores into "utility" (for satisfying tasks) and "despair" (for unsatisfying tasks), which were used as two different relevance features. To make their relevance feature representation consistent with that from our AcTS model, i.e., one utility score per query-URL pair, we unified "utility" and "despair" by simply treating "despair" as negative "utility." As a result, we can apply the same aggregation strategy over all the query-URL pairs based on MML's and LogiReg's output to generate new relevance features from those methods.

In addition, we also include a session-based click model, i.e., Session Utility Model (SUM) [43], as a baseline in this experiment. SUM aims to extract intrinsic relevance of documents to the given query from users' click behaviors in search sessions (tasks). However, it assumes all the search tasks are satisfying when modeling clicks. Thus, it is necessary to investigate if modeling search-task satisfaction is necessary for estimating document relevance from user clicks.

In our experiment, we fix the total number of trees in LambdaMART to 100, each of which has 15 nodes. The learning rate is set to be 0.1. Five-fold cross-validation is used in evaluation, where we use one fold of data for testing, one fold for validation and the rest for training. We compute four standard IR evaluation metrics: by treating all the labels above "fair" as relevant, we calculated P@1, MAP and MRR. NDCG@5 is computed based on the five-point relevance scale. The improvements of LambdaMART's ranking performance with different additional relevance

Table 6.6: Ranking performance improvements of LambdaMART with additional document relevance features estimated from different methods.

%	P@1	MAP	NDCG@5	MRR
MML	+4.926	+3.482	+3.573	+2.650
LogiReg	+5.110	+3.352	+3.783	+2.776
session-CRF	+4.752	+3.402	+3.896	+2.616
SUM	+5.101	+3.405	+3.946	+2.807
AcTS†	**+5.366**	**+3.819**	**+4.278**	**+2.955**

†: p-value<0.01 in all the metrics.

features against the original features are listed in Table 6.6.

The new relevance features from AcTS significantly improved LambdaMART's performance against the original features under all the metrics (p-value<0.01). We examined the learned tree models in LambdaMART and found all the features generated by AcTS model are selected as important splitting factors (i.e., among the top 10 important features). The features from AcTS also significantly improved the MAP and NDCG@5 metrics (p-value<0.05) comparing to those from MML and SUM, which are the second best methods under these two metrics respectively. Since no baseline search satisfaction models can distinguish the fine-grained action-level satisfaction, their estimated relevance features are not as accurate as those from the inferred action-level labels of our AcTS model. Comparing to SUM, although it distinguishes the utility of different clicked documents, it does not consider overall task satisfaction when modeling user clicks, and thus the relevance features from AcTS led to better improved ranking performance. This result validates the need to distinguish overall task satisfaction in modeling clicks for document relevance estimation.

Query Reformulation Quality Estimation

Search tasks provide rich context for performing log-based query suggestion [46, 80]. Liao et al. [80] reported that a Log Likelihood Ratio (LLR) based query similarity metric achieved the best performance in their task-based query suggestion experiment. In this section, we investigate how the identified action-level user satisfaction labels can be used to further improve LLR in task-based query suggestion.

Given two queries q_a and q_b, assuming q_b is issued after q_a, LLR makes the null hypothesis H_0 as: $P(q_b|q_a) = p_0 = P(q_b|\neg q_a)$, i.e., q_a and q_b are independent; and the alternative hypothesis H_1

as: $P(q_b|q_a) = p_1 \neq p_2 = P(q_b|\neg q_a)$, i.e., q_a and q_b are dependent. Likelihood ratio test is used, in which the test statistic is defined as $-2\ln\frac{\max_{p_0} L(H_0)}{\max_{p_1,p_2} L(H_1)}$, to determine the dependency between q_a and q_b. If the value of test statistic is larger than a predefined threshold, the null hypothesis is rejected, i.e., q_a and q_b are determined to be dependent, and q_b will be selected as a suggestion for q_a.

In Liao et al.'s work, the probabilities of p_1, p_2 were estimated by the occurrences of consecutive queries in the same task, without considering the quality of query reformulations. For example, if a satisfying query q_a is frequently reformulated into an unsatisfying query q_b, even though they are strongly correlated according to LLR, we should not suggest q_b for q_a to users. To take the inferred action-level satisfaction into account, we weight the consecutive query pairs according to their inferred satisfaction labels by,

$$c(q_a, q_b) = \exp(h_b - h_a) \tag{6.11}$$

i.e., we emphasize the pair of queries in LLR calculation, where the follow-up query improved user satisfaction; and downgrade the reformulations that hurt user satisfaction. Based on this weighting scheme, the same LLR test statistics are computed for measuring correlation between queries.

The LLR test statistics for all consecutive query pairs in a task are computed based on the first three-month search logs. The same threshold, 100 as used in [80], is applied to filter the suggestion candidates. The fourth-month search logs are used as the testing set to examine whether the suggested queries will be issued by users after the target query [103]. Such evaluation measures utility of the suggested queries in real usage context. In particular, the next three consecutive queries following the target query in the same search task are regarded as relevant in computing the evaluation metrics of P@3, MAP and MRR. To make the evaluation results comparable between the baseline (LLR without query weight-ing) and our method (LLR with query weighting), we only evaluated the overlapped target queries in both methods.

Beside this automatic evaluation method, we also collected a set of manual annotations to assess the quality of the suggested queries. We ordered all the target queries from the first three-month search logs according to their frequency, and treated the first third of queries as high frequency queries, the second third as medium, and the rest as low. 100 queries were randomly selected from

Table 6.7: Query suggestion performance improvements with query reformulation quality estimated by AcTS.

%	P@3	MAP	MRR
Query log†	+7.18	+8.60	+6.42
Annotation	+2.58	+6.79	-0.75

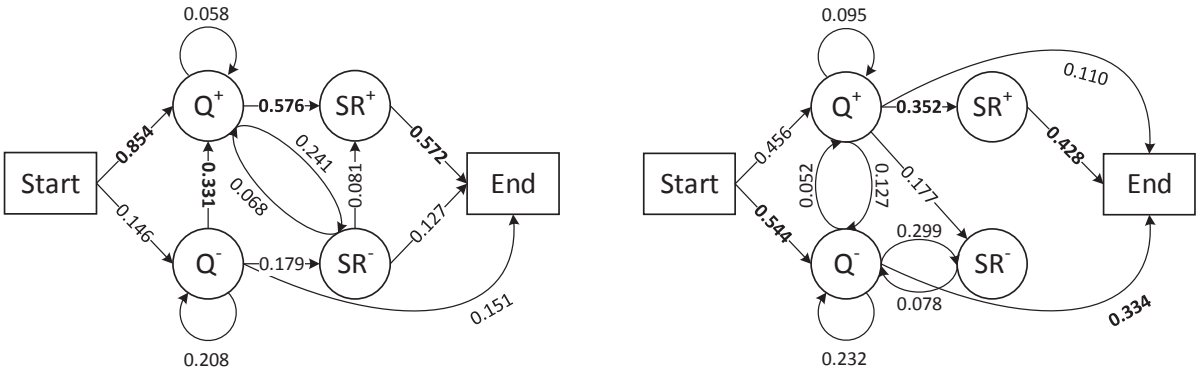
†: p-value<0.01 in all the metrics.

each category. For each selected target query, the top five suggestions from both methods were selected and interleaved before being presented to the annotators, in order to reduce annotation bias. Six human annotators were recruited to label the suggestion results. They were instructed to judge if the suggestions are relevant to the given target query with binary labels. Annotators were separated into two groups, each of which was required to annotate 150 target queries selected evenly from the above three categories. The final relevance judgment was obtained by majority vote. We list the improvement of the LLR-based query suggestion performance from the new query weighting scheme on these two testing sets in Table 6.7.

As shown in the results, the new query weighting scheme greatly improved the original co-occurrence based query suggestion performance. In the log-based automatic evaluation, all the performance metrics are significantly improved. According to the manual judgments, the major improvement is derived from the low frequency queries: P@3 and MAP are improved by 14.8% and 15.1% accordingly (p-value<0.01). In Liao et al.'s reported result, their task-based LLR performed poorly on this category, which they attributed to the sparsity of query co-occurrence. Therefore, we can find clear benefit of distinguishing the quality of reformulated queries when performing query suggestion for those low frequency queries. And the inferred action satisfaction labels from AcTS provide such a reliable quality estimator for further improving the query suggestion performance.

Analysis of Search Behavior Patterns

Analyzing user search behavior patterns is important, since it helps us understand how the users use search engine to solve search problems. Ageev et al. [2] analyzed search paths in different types of users and tasks, and identified distinct users' in-task behavioral patterns. In particular, they approximated the search paths based on the manually annotated search actions. However, such manual judgments are not generally available and expensive to acquire at scale. In contrast, our



(a) Search paths in satisfying tasks

(b) Search paths in unsatisfying tasks

Figure 6.3: Search paths estimated by the inferred action satisfaction labels from AcTS in toolbar data. Edges with transition probability less than 0.05 are discarded. Bold depicts highest outgoing transition probability.

model is capable of performing such analysis of user search activities *without manual annotations*.

We performed this analysis on the toolbar data, where we do not have action-level annotations. The first-order transition probabilities between different action types with respect to the inferred action satisfaction labels are estimated. In Figure 6.3, we demonstrated two subgraphs of the identified search paths in satisfying and unsatisfying tasks. We ignored the edges with transition probability less than 0.05 and used bold font to highlight the outgoing edges with the highest transition probability from each node in the figure. Since we only showed a sub-graph from the original graph, the illustrated outgoing transition probabilities of some nodes may *not* sum up to one (e.g., $p(\text{SERP}^-|\text{SR}^-) = 0.558$ is not included in Figure 6.3(b)).

According to the search paths estimated from the inferred action satisfaction labels, users exhibit quite different behavior patterns in the satisfying and unsatisfying search tasks. In a satisfying task (as shown in Figure 6.3(a)), users usually start with a satisfying query ($p(\text{Q}^+|\text{START})$=0.854), which will very likely result in a satisfying click ($p(\text{SR}^+|\text{Q}^+)$=0.576); while in an unsatisfying task (as shown in Figure 6.3(b)), users are more likely to begin with an unsatisfying query ($p(\text{Q}^-|\text{START})$=0.544), and move to some unhelpful documents. An interesting search pattern we observed in the estimated search paths is that in a satisfying task, once users issue an unsatisfying query, they can quickly correct it and reformulate a satisfying one ($p(\text{Q}^+|\text{Q}^-)$=0.331); while in an unsatisfying task, users tend to get stuck in a sequence of unsatisfying queries ($p(\text{Q}^-|\text{Q}^-)$=0.232) and end up with a failed search task ($p(\text{END}|\text{Q}^-)$=0.334). These are examples of the types of in-

sights that our automated method can yield, without having to apply expensive manual labeling.

6.6 Conclusions

In this chapter, we explicitly modeled searchers' satisfaction at the action level for search-task satisfaction prediction. A latent structural learning framework was developed to model the unobservable action-level satisfaction labels, which enabled us to explore rich structured features and dependency relations unique to search satisfaction modeling. Significant performance improvements in extensive experimental comparisons against several state-of-the-art search satisfaction models confirmed the value of modeling action-level satisfaction in search-task satisfaction prediction. Moreover, we demonstrated clear benefit of the inferred action satisfaction labels in other search applications such as document relevance estimation and query suggestion.

As the next step for this line of work, it will be interesting to investigate how to apply the developed framework for predicting search-task satisfaction in real time, action-by-action. If we can detect task failure early in the search process, search engines can adjust their ranking strategies or search support offered, before the users abandon their search. In addition, exploring the applications of action-level satisfaction labels in additional contexts is also an interesting direction to pursue.

Chapter 7

Conclusions and Research Frontiers

Modern information service systems, such as search engines and recommender systems, have become an indispensable tool in most people's daily life when accessing information and making decisions. Among all the components within a typical information service system, the user modeling module is arguably one of the most essential part, which enables in-depth understanding of end users and provides invaluable insight for system design and optimization.

In this thesis, we focus on developing computational user intent models via broad exploration of user-generated data - from users' opinionated review text data, i.e., their explicit opinions towards a product or a service, to their interactive behaviors recorded in the information service system, e.g., clicking on a document or reformulating the previous query. Novel computational models are proposed to analyze different types of user-generated data for capturing their underlying latent intents. A positive feedback loop is therefore formed between the end users and systems via the proposed user modeling approaches: users make informed decisions based on the synthesized information from the system, and the system refines its service strategy according to users' feedback. Both users and system collaboratively optimize their own objectives within such a feedback loop. The utility of the proposed user modeling approaches has been proved by the extensive experimentations based on practical information service systems (e.g., commercial search engine systems) and a prototype system ReviewMiner, which is developed based on the work of latent user intent identification for opinionated text data.

7.1 Conclusions

In this thesis, we have proposed a general framework for modeling users' behaviors, and developed effective computational models to analyze and understand users' underlying intents by taking a

comprehensive view of their generated data. Two specific forms of problem formalization, from the perspectives of probabilistic modeling and max-margin principle, are discussed at the beginning of this thesis, and detailed instantiations of the proposed framework are illustrated with different types of user behaviors and underlying user intent of interest.

- **Modeling opinionated text data for latent user intent identification.** In the first part of this thesis, to enable deep understanding of opinionated text data, a new opinionated text mining problem called Latent Aspect Rating Analysis (LARA) is proposed and studied. Clearly distinct from all previous work in opinion analysis that mostly focuses on integrated entity-level opinions, LARA *for the first time* reveals individual users' latent sentiment preferences at the level of topical aspects in an unsupervised manner. Discovering such detailed user preferences (which are often hard to obtain by a human from simply reading many reviews) enables applications beyond traditional sentiment analysis studies. First, such analysis facilitates in-depth understanding of user intents. For example, by mining the product reviews, LARA recognizes which aspect influences a particular user's purchase decision the most. Second, by identifying each user's latent aspect preference in a particular domain (e.g., hotel booking), personalized result ranking and recommendation can be achieved. Third, discovering the general population's sentiment preferences over different aspects of a particular product or service provides a more effective way for businesses to manage their customer relationship and conduct market research.

- **Modeling interactive behavior data for system optimization.** In the second part of this thesis, users' interaction patterns recorded in search engine logs (e.g., their issued queries and clicked documents) are explored for understanding and modeling users' longitudinal information seeking behaviors. First, A novel learning-based method is developed to associate users' scattered search behaviors over time to reflect their underlying search intents in Chapter 4. Capitalizing on the unique patterns of users' search behaviors identified in this work, the proposed method is able to achieve performance gain while reducing the reliance on costly human annotations. Based on the identified long-term search tasks, in Chapter 5, individual users' historical search behaviors in the same task are leveraged to personalize the system's output according to the identified preference within the task. The unique advantage of the

proposed ranking model adaptation method is its adaptation efficiency: the adaptation is achieved via a series of simple linear transformations over the global model according to each user's click feedback, which enables online adaptation on a per-user basis. In addition, the proposed method is general: it can be easily applied to adapt many state-of-the-art ranking algorithms and achieve promising personalized ranking performance. The last but not the least, it is important for the system to accurately assess the users' satisfaction with regarding to its output so as to adjust its service strategy in the future. In Chapter 6, a latent structural learning method, whereby rich structured features and dependency relations unique to search satisfaction prediction are explored, is proposed to predict user search-task satisfaction. In particular, the searchers' latent action-level satisfaction (i.e., whether they believe they were satisfied with the results of a query or click) is explicitly modeled for predicting their overall search satisfaction. Not only better task-level satisfaction prediction performance is improved by the proposed structural modeling approach, but also helpful insight is achieved by explicitly modeling the users' detailed action-level satisfaction. This modeling approach gives us a more comprehensive understanding of a user's information seeking process.

The research work in this thesis has generated concrete real-world impact.

- **ReviewMiner system.** Based on the techniques developed in solving the LARA problem, a system named ReviewMiner [1] for automated opinion mining and business intelligence is developed in this thesis. ReviewMiner provides multi-modal opinion analysis and decision making. Opinionated review documents collected under different categories, e.g., hotel reviews from TripAdvisor (www.tripadvisor.com), product reviews from Amazon (www.amazon.com), and medication reviews from WebMD (www.webmd.com), are indexed and analyzed. The system provides end users with easy access of analyzed aspect-level opinions in textual, temporal and spatial dimensions. Based on the techniques developed in the second part of this thesis, ReviewMiner automatically adapts to different users' aspect preferences from their usage history in the system to perform personalized recommendation and result ranking. Currently, no commercial review sites, e.g., Amazon (www.amazon.com) or Newegg (www.newegg.com), can provide such in-depth analysis of user opinions and preferences when assisting users to make

[1] http://timan100.cs.uiuc.edu:8080/ReviewMiner/

decisions. In addition, ReviewMiner provides functionalities for business analytic researchers to keep track of customer feedback and to understand customer opinions of products and services. For example, ReviewMiner can recognize the inquired item's mostly commented aspects in the customer reviews, identify the corresponding relative emphasis the users have expressed over those aspects and track the temporal dynamics of user opinions and emphasis over those aspects. Such analysis can hardly be achieved in any other existing opinion mining or business analytic systems.

- **Application in commercial search engine systems.** The long-term search task identification method proposed in Chapter 4 has been deployed in the commercial search engine Bing's internal tool chain. The proposed method can be easily parallelized, and it can process 14-days Bing search logs in only 30 minutes. Both researchers and engineers in Bing can now use this method for analyzing search logs and understanding users long-term information seeking behaviors.

7.2 Possible Extensions

We have discussed the closely related future work of the proposed computational methods in the corresponding chapters before. In the end of this thesis, we would like to give a more systematic discussion about the possible extensions of those studies, by introducing more types of signals for analysis and making more explicit connections among the proposed modeling approaches.

In the proposed solutions of latent aspect rating analysis in Chapter 2, no information about individual users is considered; instead, the same prior distribution for aspect weights and word sentiment polarities are shared across users. The main reason for sharing the parameters in the proposed solution is the lack of observations for multiple reviews from the same user: in the hotel data set, most of reviewers only contribute one review document. To address this sparsity issue, we can consider to perform user clustering together with latent rating analysis [121]. In this approach, each user will associate with a latent user group (e.g., business trip traveler v.s. casual vacation traveler), and each group will have its own prior distribution of the aspect weights and word sentiment polarities. By performing such joint modeling, we can not only identify different

users' preferences over aspects more precisely, but also group the users according to their behavior patterns, which can be used as an additional signal to profile users.

In addition to exploring various signals for better modeling users' rating behaviors, the learned model of LARA can also be used to assess the trustworthiness of the review content. Given the proposed LARA model captures the generation process of ordinary users' rating behaviors and a spammer's behavior is dramatically different from ordinary users' [67], the generation likelihood given by LARA model on different review document will be a good signal for detecting spam reviews.

The models proposed in the second part of this thesis, i.e., the model for identifying users' long-term search task studied in Chapter 4, the model for personalized ranking model adaptation studied in Chapter 5 and the model for task satisfaction prediction 6, act as separated components in a pipeline. That is: search actions are first segmented into tasks, then personalization is performed based on the recognized tasks, in the end search utility is evaluated by the predicted user satisfaction. However, it is possible to connect those components in a more unified way by exploring the online learning paradigm [16, 76], such that they could reinforce each other at real time. For example, when a user finishes a search action within an ongoing user task, the system can promptly evaluate her satisfaction of this action, pass the feedback back to the personalization and task segmentation modules (see in Figure 1.1). If it is judged to be positive feedback, it indicates both the personalization and task segmentation modules made good predictions and therefore their model parameters can be enhanced; while if it is negative feedback, it indicates either the personalization module or task segmentation module (or both of them) made a mistake, they need to refine their model parameters according to such negative feedback. When it is hard to resolve the reason for the negative feedback, the system can record the case for later manual inspection. This real-time model learning would enable direct interaction between users and system, and make it possible for the system to actively acquire immediate feedback from the users.

From a practical perspective, it would be also interesting to explore such direct interaction between users and system in the developed ReviewMiner system. In the current ReviewMiner system, each registered user is assigned to a unique ID for recording her behaviors, different types of user-generated data in the system can therefore be analyzed by the models proposed in this thesis

(from both part I and part II) for constructing a complete understanding of the user. By providing function support to allow users to write reviews in ReviewMiner system, the LARA model can be used to identify their preferences over the aspects from their generated opinionated reviews (note: right now the system users in ReviewMiner are not authors of the analyzed reviews); by associating users' searching, browsing, and commenting behaviors in the system together, more comprehensive user-tasks can be identified; and the personalization and task satisfaction model can be deployed in the system to better assist users in their information seeking process.

More importantly, the developed ReviewMiner system enables us to acquire in-situ user feedback for all the computational models we have developed in this thesis. For the LARA work, no aspect weight is available at any existing online review website, and we have to model it as latent variables in our solution. Nevertheless, in ReviewMiner, once the user publishes a review in the system, we can explicitly ask her to confirm the aspect weight predicted by our LARA model on her review. Such explicit feedback can then be utilized to estimate better parameters in the proposed LARA models. The same strategy can be used to acquire explicit feedback on the task segmentation module, personalization module and satisfaction prediction module as well. For example, asking the user if she is on a predicted task, or if she is satisfied with the retrieved items. Active learning strategies can be explored [112, 30] in this line of study in order to reduce the frequency of acquiring explicit feedback from the users.

7.3 Research Frontiers

The research studied in this thesis catalyzes a new frontier for data mining and knowledge discovery.

We are now living in the era of big data, where a large amount of data is accumulated at an accelerated speed. It is worth noting that a large portion of such data is actually produced and consumed by humans, and it will continue to fuel exponential growth. For example, every day there are 5 million transactions processed on eBay [7], 340 million tweets posted on Twitter [15], and 5.13 billion searches performed on Google [17]. Huge amount of invaluable knowledge is buried in such human-generated data. By tapping into it, decision makers - from ordinary users who make daily purchase decisions to senior officers in international organizations who make strategic plans - would all benefit from having access to the knowledge discovered from such data.

Computational user models studied in this thesis have explored several main categories of human-generated data, e.g., opinionated text document and interactive behaviors, and build a solid foundation for modeling many types of human-generated data and extracting actionable knowledge from it. The future work along this line of research is to *put humans into the loop of mining big data*. Figure 7.1 outlines the research of human-centric data mining and knowledge discovery, which is a generalized vision of the positive feedback loop that we have discussed in Chapter 1.

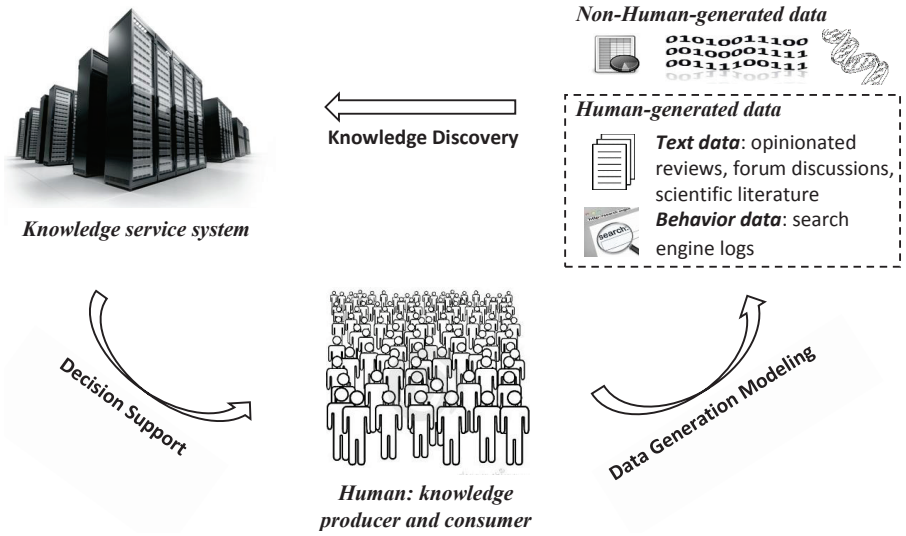


Figure 7.1: Putting humans into the loop of mining big data.

In Figure 7.1, humans play a dual role as both data producers and knowledge consumers. First, as the producer of data, humans constantly generate two kinds of unique data, both are extremely valuable from the perspective of knowledge discovery and machine learning. 1) As a consequence of human communications in natural language, people routinely generate enormous amounts of unstructured natural language text data, e.g., scientific literature, online forum discussion and social network communication, which embeds rich semantic information about users' latent intents. 2) As users of knowledge service systems, people interactively create continually growing behavior data, which reflects their latent information need and decision making process. Second, as the consumer of the mined knowledge, different users have distinct requirement of types of data type for analysis (e.g., DNA sequence for biologists and historic stock quotes for financial analysts) and preferences of information for decision making, thus accurately understanding their respective information needs and decision preferences is crucial for providing effective decision support. Blindly mining patterns

or statistics from data will inevitably hide the big picture from us, and cannot guarantee optimized system utility in assisting users.

Comparing to the concept of positive feedback loop between users and information service systems discussed in Chapter 1, the research direction of "human-centric data mining and knowledge discovery" is more general and fundamental. The positive feedback loop takes a *micro* view for analyzing the data generated when users interact with system; while the "human-centric data mining and knowledge discovery" takes a *macro* view, where both human-generated and non-human-generated data is analyzed for maximizing the utility of the mined knowledge for users' decision making process.

The computational user models developed in this thesis build a technical foundation for the more general exploration of "human-centric data mining and knowledge discovery." For example, the latent aspect rating analysis work studied in Chapter 2 can be generalized to study the generation process of other types of human-generated data, e.g., forum discussion and movement trajectory, for extracting the underlying latent user intents. And the method developed for identifying users' long-term search tasks in Chapter 4 can be generalized to a user's other types of behaviors, e.g., location update and online purchase, for discovering in-depth knowledge about human behaviors in general.

The ideal outcome of research in the line of *put human into the loop of mining big data* is an intelligent knowledge service system which directly *talks to the users and learns from the users*. Such a system is beyond the traditional information retrieval systems (e.g., the commercial Web search engines like Bing and Google), and it is a fusion of systems for information extraction, integration, retrieval and operation optimization. The system would record every action the user has taken when interacting with the system (e.g., acquiring knowledge, browsing results, and posting contents), correctly responds to the user's inquiry by analyzing his or her intention at that particular time, and actively updates its service strategy when the user's interest changes. As a result, such a system now should directly provide knowledge, rather than raw information, to the users, i.e., it actively helps user to perform decision making instead of just passively providing information. With such an intelligent system, users will have no need to be trained for any specific interaction scheme or to exhaust the combination of query keywords to describe their information

need to the system (as they have to do in the current generation of information service systems). Instead, they could easily obtain the needed knowledge for all kinds of tasks through the support enabled by the deployed techniques of data mining and user intent modeling.

To achieve such a long-term research goal, there are three major challenges to be solved: namely, knowledge discovery and representation, user understanding, and intelligent knowledge service techniques.

Knowledge Discovery and Representation: The types of data that convey invaluable knowledge are far beyond those have been explored in this thesis. Exploring all these types of data provides us a broad and comprehensive view in decision making and operation optimization. For example, mining personal health data helps doctors to find more effective and affordable treatments for each patient; analyzing vehicle GPS sensor data helps alleviating public traffic congestion; and social media data provides a unique mine for sociologists to study social influence and dynamics.

However, analyzing data in an ad-hoc manner is far from ideal, which will inevitably isolate the mined knowledge and hamper its usage across different applications. As all human-generated data is governed by the underlying intent of human behaviors, there are associations among different types of such data and all mining algorithms should be aware of these associations when modeling the data. As in the work of LARA, the latent user intents are abstracted as his or her emphasis over aspects of items in a specific domain. By knowing a user's preference in one domain, we could infer his or her preferences in similar domains, e.g., preferring a lower price in hotel booking might indicate the emphasis of price in car rental and flight tickets as well. Along the line of exploring richer types of human-generated data, a general knowledge representation scheme, e.g., an ontology structure, is necessary to organize the mined knowledge about human behaviors, so that the learned knowledge can be used in multiple application scenarios transparently.

User Understanding: Problems related to human activities have their unique properties: the human behaviors are largely determined by their own goals and preferences, rather than the instructions of system designers. As a result, effectively mining actionable knowledge from human activities requires a fusion of solutions from both computational methodologies and behavioral and psychology studies (e.g., human-computer interaction). This thesis mainly concentrates on developing data-driven computational methods for user understanding. It would be fascinating to

seek evidence and support for those computational methods from the behavioural and psychology studies, and incorporate the established theories from those fields to guide model designing and knowledge mining. Such a study will never be just an one-sided effort; the developed method and mined knowledge about users will also benefit the studies in psychology, cognitive science, design, and more.

In addition, nowadays each individual's behavior is no longer "individual," but a result of social interactions. In social science, this is named "social dynamics": the behavior of groups that results from the interactions of individual group members. Therefore, in order to understand users in a more precise and comprehensive view, we should no longer treat them independently, but put our analysis into the context of social interactions. Introducing the social connections and influence into the user modeling framework will help us to explore user behaviors and interactions within their social networks in a broader view to advance the level of user understanding and knowledge service system optimization.

Intelligent Knowledge Service Techniques: Currently, most knowledge service systems respond to users' inquires in a passive way, e.g., predefined service strategies or offline trained prediction models are used to generate the output. Such a strategy is only sub-optimal given that it cannot adapt to individual user's information need or the shift of public interest quickly. In order to build an intelligent knowledge service system, the system should not only quietly assist the users, but also actively learn from the users (e.g., probe users for additional feedback). That is, let the system and users collaborate for knowledge acquisition and decision making. Game theory provides a nice theoretic foundation to perform such study, but the research on the border of computer science and game theory is still in its infancy. The real problems in data mining for knowledge discovery are complex, dynamic, and involve a lot of uncertainty; thus, it will not be a straightforward practice of applying existing solutions in game theory to those real-life problems. Most of the research that has been conducted in the past focused on restrict assumptions about the game and players; little is known about more general settings. By explicitly modeling the interaction between users and systems, we can explore the application of game-theoretic models for building the system that actively talks to the users and learns from the users in a more intelligent way.

References

[1] Onix text retrieval toolkit stopword list. http://www.lextek.com/manuals/onix/stopwords1.html. 37

[2] Mikhail Ageev, Qi Guo, Dmitry Lagun, and Eugene Agichtein. Find it if you can: a game for modeling different types of web search success using interaction data. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 345–354. ACM, 2011. 110, 110, 111, 112, 114, 114, 115, 115, 119, 121, 121, 123, 125, 126, 127, 127, 127, 128, 130, 130, 135

[3] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10. ACM, 2006. 85, 99, 112, 131

[4] Eugene Agichtein, Ryen W White, Susan T Dumais, and Paul N Bennet. Search, interrupted: understanding and predicting search task continuation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 315–324. ACM, 2012. 61, 63

[5] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM. 1

[6] Peter Anick. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 88–95. ACM, 2003. 63

[7] Answers. http://wiki.answers.com/Q/How_many_transactions_does_eBay_process_per_day?#slide2, 2013. 143

[8] Anne Aula, Rehan M Khan, and Zhiwei Guan. How does search behavior change as search becomes more difficult? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 35–44. ACM, 2010. 110, 112, 112, 112, 114, 114, 115, 120, 121, 123

[9] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999. 89, 99, 110

[10] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596. Springer, 2005. 1

[11] Paul N Bennett, Ryen W White, Wei Chu, Susan T Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 185–194. ACM, 2012. 86, 89

[12] Christopher M Bishop et al. *Pattern recognition and machine learning*. springer New York, 2006. 7

[13] D.M. Blei and J. McAuliffe. Supervised topic models. *Advances in Neural Information Processing Systems*, 20:121–128, 2008. 38

[14] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003. 33, 34, 35, 38

[15] Twitter Engeering Blog. Twitter turns six. http://blog.twitter.com/2012/03/twitter-turns-six.html, 2012. 143

[16] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17:9, 1998. 142

[17] Statistic Brain. Google annual search statistics. http://www.statisticbrain.com/google-searches, 2013. 143

[18] Chris Burges. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report MSR-TR-2010-82*, 2010. 131

[19] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005. 14, 87, 89, 92

[20] Chrisstopher J.C. Burges, Robert Ragno, and Viet Le. Learning to rank with nonsmooth cost functions. *Proceedings of the Advances in Neural Information Processing Systems*, 19:193–200, 2007. 14, 87, 89, 94, 104

[21] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998. 41

[22] D. Cai, X. He, X. Wang, H. Bao, and J. Han. Locality preserving nonnegative matrix factorization. In *IJCAI'09*, pages 1010–1015, 2009. 77

[23] L.D. Catledge and J.E. Pitkow. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN systems*, 27(6):1065–1073, 1995. 63

[24] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 41

[25] Ming-Wei Chang, Vivek Srikumar, Dan Goldwasser, and Dan Roth. Structured output learning with indirect supervision. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 199–206, 2010. 7, 66, 70, 71, 121, 121, 123, 123

[26] D. Chen, Y. Xiong, J. Yan, G.R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2010. 87, 88, 102

[27] Depin Chen, Jun Yan, Gang Wang, Yan Xiong, Weiguo Fan, and Zheng Chen. Transrank: A novel algorithm for transfer of rank learning. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 106–115. IEEE, 2008. 86, 87, 87, 102

[28] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2005. 88

[29] William W Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480. ACM, 2002. 65, 75, 75

[30] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994. 143

[31] Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, volume 4, pages 293–300, 2004. 72

[32] Hang Cui, Vibhu Mittal, and Mayur Datar. Comparative experiments on sentiment classification for online product reviews. In *Twenty-First National Conference on Artificial Intelligence*, volume 21, page 1265, 2006. 20

[33] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics, 2004. 21

[34] Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, 2007. 114

[35] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003. 20

[36] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006. 53

[37] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990. 1

[38] Google Developers. Google maps api. https://developers.google.com/maps/. 55

[39] A. Devitt and K. Ahmad. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of ACL'07*, pages 984–991, 2007. 17, 20

[40] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001. 96

[41] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web*, pages 581–590. ACM, 2007. 88, 89, 105

[42] Kevin Duh and Katrin Kirchhoff. Learning to rank with partially-labeled data. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 251–258. ACM, 2008. 87

[43] Georges Dupret and Ciya Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 181–190. ACM, 2010. 110, 131, 132

[44] A. Esuli and F. Sebastiani. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, 2006. 43

[45] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005. 21

[46] Henry Feild and James Allan. Task-aware query recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 83–92. ACM, 2013. 133

[47] Henry A Feild, James Allan, and Rosie Jones. Predicting searcher frustration. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 34–41. ACM, 2010. 110, 111, 111, 112, 112, 112, 114, 114, 115, 119, 120, 127, 127

[48] Thomas Finley and Thorsten Joachims. Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pages 217–224. ACM, 2005. 65, 65, 75, 75

[49] Apache Software Foundation. Apache lucene. `http://lucene.apache.org/`. 52

[50] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS)*, 23(2):147–168, 2005. 57, 110, 110, 111, 112, 112, 114, 119, 119, 120, 120, 121, 121

[51] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems (TOIS)*, 9(3):223–248, 1991. 1

[52] Jianfeng Gao, Qiang Wu, Chris Burges, Krysta Svore, Yi Su, Nazan Khan, Shalin Shah, and Hongyan Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 505–513. Association for Computational Linguistics, 2009. 86, 88, 88

[53] Wei Gao, Peng Cai, Kam-Fai Wong, and Aoying Zhou. Learning to rank only using training data from related domain. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169. ACM, 2010. 86, 87, 87, 102

[54] B. Geng, L. Yang, C. Xu, and X.S. Hua. Ranking model adaptation for domain-specific search. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):745–758, 2012. 86, 88, 88, 102

[55] A.B. Goldberg and X. Zhu. Seeing stars when there arent many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*, 2006. 20

[56] Qi Guo, Ryen W White, Yunqiao Zhang, Blake Anderson, and Susan T Dumais. Why searchers switch: understanding and predicting engine switching rationales. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 335–344. ACM, 2011. 117, 119, 120, 120

[57] Ahmed Hassan. A semi-supervised approach to modeling web search satisfaction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 275–284. ACM, 2012. 114, 121, 127, 127

[58] Ahmed Hassan, Rosie Jones, and Kristina Lisa Klinkner. Beyond dcg: user behavior as a predictor of a successful search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 221–230. ACM, 2010. 110, 110, 110, 111, 112, 112, 113, 114, 114, 114, 115, 121, 127, 127

[59] Ahmed Hassan, Yang Song, and Li-wei He. A task level metric for measuring web search satisfaction and its application on improving relevance estimation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 125–134. ACM, 2011. 111, 125, 125, 132

[60] Ahmed Hassan and Ryen W White. Personalized models of search satisfaction. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2009–2018. ACM, 2013. 112

[61] Daqing He, Ayşe Göker, and David J Harper. Combining evidence for automatic web session identification. *Information Processing & Management*, 38(5):727–742, 2002. 63, 63, 63

[62] Xiaodong He and Wu Chou. Minimum classification error linear regression for acoustic model adaptation of continuous density hmms. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 1, pages I–397. IEEE, 2003. 14, 86, 89

[63] Jeff Huang and Efthimis N Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 77–86. ACM, 2009. 1

[64] Scott B Huffman and Michael Hochster. How well does result relevance predict session satisfaction? In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 567–574. ACM, 2007. 120, 120

[65] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. 65, 66

[66] Nitin Jindal and Bing Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 244–251. ACM, 2006. 17

[67] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM, 2007. 142

[68] Yohan Jo and Alice H. Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824, New York, NY, USA, 2011. ACM. 21

[69] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002. 14, 87, 89, 94

[70] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM, 2005. 57, 85, 99, 112

[71] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 699–708. ACM, 2008. 61, 61, 61, 62, 63, 63, 63, 75, 76

[72] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396. ACM, 2006. 72

[73] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. 32

[74] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics, 2004. 20

[75] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 193–202. ACM, 2014. 112

[76] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004. 142

[77] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. 124, 125

[78] Alexander Kotov, Paul N Bennett, Ryen W White, Susan T Dumais, and Jaime Teevan. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 5–14. ACM, 2011. 61, 62, 63, 64, 75, 76, 82

[79] CJ Leggetter and PC Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech and language*, 9(2):171, 1995. 14, 86, 89

[80] Zhen Liao, Yang Song, Li-wei He, and Yalou Huang. Evaluating the effectiveness of search task trails. In *Proceedings of the 21st international conference on World Wide Web*, pages 489–498. ACM, 2012. 62, 62, 74, 75, 133, 133, 134

[81] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009. 21

[82] Fang Liu, Clement Yu, and Weiyi Meng. Personalized web search for improving retrieval effectiveness. *Knowledge and Data Engineering, IEEE Transactions on*, 16(1):28–40, 2004. 4

[83] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, pages 3–10, 2007. 96

[84] T.Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. 1, 90, 91, 97

[85] Shane J Lopez and Charles R Snyder. *The Oxford handbook of positive psychology*. Oxford University Press, 2011. 112

[86] L. Lovász and M.D. Plummer. *Matching theory*. Elsevier Science Ltd, 1986. 39

[87] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM, 2009. 20, 40

[88] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. Identifying task-based sessions in search engine query logs. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 277–286. ACM, 2011. 61, 61, 62, 63, 63, 64, 65, 75, 82

[89] Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32. Association for Computational Linguistics, 2005. 76, 76

[90] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003. 21

[91] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM, 2007. 21

[92] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. 21

[93] Raquel Navarro-Prieto, Mike Scaife, and Yvonne Rogers. Cognitive strategies in web searching. In *Human Factors & the Web*, pages 43–56, 1999. 121, 123

[94] Richard L Oliver. *Satisfaction: A behavioral perspective on the consumer*. ME Sharpe, 2010. 112

[95] S.J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010. 86

[96] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005. 20, 20

[97] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM, 2005. 61, 63, 64

[98] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138. ACM, 2006. 69

[99] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50. ACM, 2005. 88

[100] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, volume 33, pages 6–12. ACM, 1999. 61, 63

[101] Catherine L Smith and Paul B Kantor. User adaptation: good results from poor systems. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 147–154. ACM, 2008. 112, 112

[102] B. Snyder and R. Barzilay. Multiple aspect ranking using the good grief algorithm. In *Proceedings of NAACL HLT*, pages 300–307, 2007. 20

[103] Yang Song, Dengyong Zhou, and Li-wei He. Query suggestion by constructing term-transition graphs. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 353–362. ACM, 2012. 134

[104] David Sontag, Kevyn Collins-Thompson, Paul N Bennett, Ryen W White, Susan Dumais, and Bodo Billerbeck. Probabilistic models for personalizing web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 433–442. ACM, 2012. 85, 88

[105] A. Spink, M. Park, B.J. Jansen, and J. Pedersen. Multitasking during web search sessions. *Information Processing & Management*, 42(1):264–275, 2006. 63, 65

[106] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM, 2004. 4

[107] Bin Tan, Xuehua Shen, and ChengXiang Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 718–723. ACM, 2006. 85, 88

[108] Jaime Teevan, Susan T Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456. ACM, 2005. 88

[109] Jaime Teevan, Daniel J Liebling, and Gayathri Ravichandran Geetha. Understanding and predicting personal navigation. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 85–94. ACM, 2011. 86, 88, 105

[110] I. Titov and R. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL '08*, pages 308–316, 2008. 20, 21

[111] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM, 2008. 21

[112] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002. 143

[113] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004. 7

[114] V. Vapnik. *The nature of statistical learning theory*. springer, 1999. 70

[115] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998. 8

[116] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001. 63, 65, 72, 75

[117] Hongning Wang, Xiaodong He, Ming-Wei Chang, Yang Song, Ryen White, and Wei Chu. Personalized ranking model adaptation for web search. In *Proceedings of the 36st Annual ACM SIGIR Conference*. ACM, 2013. 14

[118] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM, 2010. 11, 11, 19, 51

[119] Hongning Wang, Yue Lu, and ChengXiang Zhai. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–626. ACM, 2011. 11, 12, 19, 51

[120] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen White, and Wei Chu. Learning to extract cross-session search tasks. In *Proceedings of the 22st international conference on World Wide Web*. ACM, 2013. 13

[121] Hongning Wang, ChengXiang Zhai, Feng Liang, Anlei Dong, and Yi Chang. User modeling in search logs via a nonparametric bayesian approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 203–212. ACM, 2014. 141

[122] Ryen W White and Steven M Drucker. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*, pages 21–30. ACM, 2007. 74

[123] Ryen W White and Dan Morris. Investigating the querying and browsing behavior of advanced search engine users. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 255–262. ACM, 2007. 115, 121, 123

[124] Ya Xu and David Mease. Evaluating web search using task completion time. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 676–677. ACM, 2009. 111, 112, 119, 119

[125] Yiming Yang and Jan O.Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML'97*, pages 412 – 420, 1997. 24

[126] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM, 2009. 66, 70

[127] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106, 2003. 21

[128] Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65, 2010. 21

[129] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50. ACM, 2006. 17