ANALYZING AND MODELING GROUPS

BY JINYUN YAN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

S. Muthukrishnan

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2015

ABSTRACT OF THE DISSERTATION

Analyzing and Modeling Groups

by Jinyun Yan

Dissertation Director: S. Muthukrishnan

A group is a collection of humans. Members within a group often share certain characteristics, interests and preferences, along with their individual differences. Such collections of members lead to interesting collective behavior. In this work, we analyze and model the behavior of groups when part of three real-world applications: group recommendation, personalized search and reference group identification.

Group recommendation is a variation of the classical problem of recommending items, but where the client is a group rather than an individual. We are interested in the setting where individuals, part of the same group or not, interact regularly with the recommender system. There are two challenges in group recommendation in this setting: 1) historical information about member and group is often missing; 2) members' presence when they ask for a recommendation may be different at different times. We formulate this problem as a group multi-armed bandit problem and design policies for two types of group feedback. We develop a demo system to collect member and group feedback on recommendations to group events and observe the existence of member influence when the group wants to reach consensus.

Personalized search results rely heavily on individuals' search and click history. However, a large portion of queries submitted by users each day is new. It is hard to improve search relevance on these queries. We analyzed queries and clicks at group

level and observed that individuals' click preferences align well with groups' preferences. With this in mind, we propose cohort models that model each user through groups of users who are similar in one or more dimensions, and facilitate personalized search through cohort's search intent and click preference. Experiments show that cohort models can achieve significant improvement on search relevance, particularly when personal historical data is insufficient.

A good way to assess a person is to look at her reference group. A person is considered to be equal or near equal to people in her reference group. We study the problem of finding a group of comparable people for any given researcher, so that we can better represent and understand the researcher in query. To do so, we build researchers' research trajectory with year, publication and venue. Then, we use a trajectory matching algorithm to determine how similar they are and identify relevant candidates. Our algorithm can be easily modified to find more senior researchers whose early stage of their career is comparable to a given junior researcher. We also provide a map-reduce version of our matching algorithm to make it scale well with data.

Acknowledgements

I am immensely fortunate to be advised by Professor S. Muthukrishnan. He is one of the smallest and most hard working people I know. His reasoning, thoughts and broad knowledge have profound impact on my research and on my life. By working with him, I learned the beauty of asking questions, formulating problems and expressing ideas. He encouraged me to purse what I want to do and has been very supportive during my Ph.D.. This thesis would not have been possible without him.

I am very grateful to work with Graham Cormode. His expertise and in-depth guidance helped me a lot. He is a turn-to person whenever I have questions. He gave me many unselfish help and valuable suggestions. I would thank Professor Tomasz Immielinski, who guided me through my first research step. Special thanks to Professor Badri Nath for serving on my committee and for his valuable feedback on my work.

I have spent memorable summers at Technicolor Labs, Microsoft Research and Bing. More than two of thirds of the thesis got done during these internships. Thank Stratis Ioanndis, Jose' Bento, Ryen White, Wei Chu for great insights and mentoring. In addition, I am very thankful to members of MassDAL, all fellow students in the CS department of Rutgers and colleagues during internships. I benefited a lot from working with and learning from these smart people.

The last but not the least, I thank my husband Qi, my parents, younger brother and my parents-in-law for their enduring love and support.

Dedication

To my daughter Anna Chen.

Table of Contents

Abstract ii

Acknowledgements iv

Dedication v

List of Tables ix

List of Figures x

1. Introduction 1

1.1. Recommendation Targeting to Groups 1

1.2. Enhance Personalized Search by Groups 4

1.3. Identify Reference Group for Researchers 5

2. Recommendation to Groups 7

2.1. MAB Background and Problem Formulation 8

2.1.1. Multi-armed Bandits Background 8

2.1.2. Group-MAB Formulation 9

2.1.3. A Naive Approach and Its Regret Bound 11

2.2. Two Settings of Reward Process 12

2.2.1. Receiving Individual Rewards 13

2.2.2. Receiving Group Reward 14

2.3. Related Work 22

2.3.1. Group Recommendation 22

2.3.2. Stochastic Contextual Bandits 24

2.4. Proofs 27

2.4.1. Proof for Theorem 1 27

2.4.2. Proof for Theorem 3 30
2.5. Numerical Results 35
2.5.1. Group-UCB 35
2.5.2. Contextual ϵ-Greedy 38
2.6. A Group Recommender Demo System 39
2.6.1. System Overview 41
2.6.2. Recommendation Model 42
2.6.3. User Study and Model Evaluation 46
2.7. Conclusion 47

3. Cohort Modeling for Enhanced Personalized Search 48
3.1. Introduction 48
3.1.1. Background and Related Work 48
3.1.2. Contributions 51
3.2. Cohort Modeling 52
3.3. Empirical Datasets 58
3.3.1. All Queries 58
3.3.2. New Queries in User Search History 59
3.3.3. Popularity of Queries 60
3.3.4. Query Entropy 60
3.3.5. Acronym Queries 61
3.4. Experiment Results and Findings 62
3.4.1. Ranking Models and Evaluation Metrics 62
3.4.2. Research Questions and Findings 64
3.5. Learned Cohort Models 69
3.5.1. Clustering Method 70
3.5.2. Evaluating Clustered Cohorts 71
3.5.3. Preference Analysis 73
3.6. Discussion and Conclusion 74

4. Comparable Groups 77
4.1. Introduction 77
4.2. Problem Setting 79
4.3. Data and Analysis 80
4.3.1. Exploratory Data Analysis 80
4.4. Algorithms to Comparable Researchers 83
4.4.1. On Venue Score 85
4.4.2. On Topic Similarity 88
4.4.3. On Prefix Matching 90
4.4.4. On Parallel Computation 91
4.5. Conclusions and Future Work 92

5. Summary and Future Directions 95

References 97

List of Tables

2.1. Member information in the group 36

2.2. Prediction performance on test set, prediction threshold = 0.5 46

3.1. Data sets for cohort modeling experiments, All dates from 2013. 59

3.2. Gains in MAP and MRR over baseline(± SEM). 65

3.3. Gains in MAP and MRR over baseline for acronym queries(± SEM). . . 69

3.4. Gains in MAP and MRR over baseline for acronym queries(± SEM). . . 72

4.1. Dataset statistics 80

4.2. Sequence example of a researcher 83

4.3. An optimal sequence matching 87

4.4. Case study of edit distance 88

4.5. Case Study of Topic Edit Distance 94

List of Figures

2.1. (a) Random groups up to 10 users. (b) Average slope of the regret with regard to group size. 36

2.2. (a)Regrets for groups of people with same zip code. (b)Regrets for equal weight vector and influence weight vector. 37

2.3. (a) Distributions of scaling coefficients for households of size 2 and 3. (b) Distributions of convergence times for households of size 2 and 3. . . . 38

2.4. Scatter plot showing correlation between scaling coefficients and convergence times. 39

2.5. (a) Regret over T when x_t is from i.i.d. (b) Regret over T when x_t is not from i.i.d. 40

2.6. Group decision making framework 40

2.7. Create event and recommendation for the group 42

2.8. Voting and other members' decisions 43

2.9. Information stored by the system 44

3.1. Gains in MRR over baseline for each cohort type for new and old queries from each user (±SEM) 66

3.2. Gains in MRR over baseline for each cohort type for difference in the popularity of the query(±SEM) 67

3.3. Gains in MRR over baseline for each cohort type for different query entropy bins(±SEM) 68

3.4. Gains in MRR over the baseline for clustered cohorts versus predefined cohorts for different k for a selected query set(±SEM). 71

3.5. Average DiffTop weight for each cohort (±SEM). 75

4.1. Tables and data schemas 79

4.2. Start and end years. 81
4.3. Research period length. 81
4.4. Burst speed and half speed 83
4.5. Correlations with #publications 84
4.6. Correlations with h-index and g-index 86

Chapter 1

Introduction

Evolutionary theory suggests humans evolved to function in groups for survival. People consciously form, join or leave groups, for instance, families and unions. They also belong to multiple groups by traits, demographics, or interests. Members in the group often exhibit common characteristics, shared interests and similar preferences, as well as their individual differences. Such mixture leads to interesting and complex collective behavior.

With the bloom of internet, social networking sites and mobile apps, in addition to current trend of moving offline activities to online, tons of data are generated. There is extensive research on mining such data to understand individuals, in turn to provide good personalized services. However, groups are less explored. In this work, we focus on analyzing and modeling groups in three real-world problems: recommendation targeting to groups, personalized search enhanced by groups and identification of reference groups for researchers through scholarly data.

1.1 Recommendation Targeting to Groups

Recommender systems have become an important research topic. There are many practical applications equipped with recommendation to help users to deal with information overload and get personalized experiences. Examples include recommending products at Amazon [51] and movies at MovieLens [57] and Netflix [12]. The recommendation problem is commonly reduced to rating prediction problem that estimates ratings for items that the user has not seen nor rated. Once we obtain the estimated ratings, we can recommend items with highest ratings to the user.

Formally, the recommendation problem can be formulated as follows: let U be the

set of all users, and I be the set of items, let f be the utility function $f : U * I \rightarrow R$ that measures the relevance of the item $i \in I$ to the user $u \in U$. Then for each user u, we can recommend the item that maximizes the utility: $i^* = \arg\max_{i \in I} f(u, i)$ The utility of an item is usually represented by the rating, which is in certain range, e.g. [1, 5], and indicates the user's satisfaction to the item. Each user u can associate with a profile which can include demographic features or interests. Similarly, items can also associate with a set of characteristics.

However, in many real-life cases, recommendations are consumed by groups of users rather than individuals, for instance, watching movies with family, dining out with friends and team building with colleagues. In fact, seldom is a person alone in many of their daily activities. We are interested in the recommendation problem that targets groups. Let G be the set of all groups, the utility function $f : G * I \rightarrow R$ then measures the relevance of the item i to the group $g \in G$. Existing work on group recommendation focuses on the definition of the utility function, then reduces the problem to traditional recommendation to individuals. Examples of the group utility function include taking averaging, minimum or maximum over members' utility functions.

Yet there are many challenges for group recommendation. Firstly, groups are dynamic. Even for persistent groups, e.g. families, that recurrently attend certain group activities, it is hard to guarantee every member will participate in each event. The recommendation algorithm should be flexible in order to handle the challenge of dynamic presence of members. Secondly, information about the group and members can be missing. There could be new groups joining the system and new users joining a group. While individuals' and groups' preferences are crucial to make customized recommendation, it is extremely expensive to collect them. The recommender system then often learns preferences through historical behaviors, more precisely, previous ratings to items. When encountering new groups or new users in the group, which is named as cold start problem, the algorithm needs to be capable to generate good recommendations and learn preferences quickly.

To address above challenges, we approach the group recommendation problem as a variation of multi-armed bandits (MAB) problem. MAB [62] is a sequential decision

problem with the exploration and exploitation tradeoff, that balances the decision between staying with the option which gave the highest payoff in the past and exploring new options that might give higher payoffs in the future. Formally, let A be the set of K arms, which payoffs are from underlying distributions. The learner does not have any prior knowledge about the payoffs. It plays an arm $a \in A$ at each time t, and receives the payoff a_t. The objective is to maximize the expected payoffs over T plays. There are many policies for the MAB problem, such as ϵ-greedy [83], and UCB [6], that work well in practice and have theoretic guarantees. This makes MAB a good fit to solve cold start problem in recommendation [47].

In our setting, each arm has underlying distribution of payoffs for groups, rather than a single user. At each time t, a group g_t shows up and the learner picks an arm to the group. The learn then receives the feedback (payoff) to the selected arm by the group. We consider two types of feedback process: 1) each present member gives feedback; 2) the group gives a single group feedback. For the first type, a metric is desired to transfer individual feedback to group feedback. For the second type, individual preference needs to be learned to effectively handle dynamic presence of members. We design policies for each type of feedback and prove that they are efficient in terms of computational time and space.

In traditional recommender systems targeting to individuals, the post-recommendation feedback can be collected on the individual level, from explicit ratings to implicit ratings derived from clicks or page views etc.. However, the post-recommendation feedback is harder to collect on the group level. Usually the group needs a group decision making process to finalize the option out of recommended items. The final decision reflects the group feedback to the item, and members' opinions in the decision making process reflects individual feedback conditioned on the group. We design a demo system that records and facilitates group consensus on recommended items. The system allows users to organize group events. It generates recommendations to groups and provides voting and chatting functionalities to members to reach consensus. A user study is conducted through the demo for lunch dining out events among colleagues in a research lab. We observed member influence during decision making process. Based on this, we propose

an influence cascade model to predict the pairwise influence, which is in turn used to improve the estimation of individuals' decision on recommended items for group events.

1.2 Enhance Personalized Search by Groups

Personalization of search results has been investigated in detail in domains such as Web search and beyond [70, 74, 78]. The ability to tailor search results to a particular individual enables a wealth of opportunity to better satisfy their particular information needs. Personalization models are typically learned from observed short- and long-term search behavior (such as queries and result clicks), which is either used directly [79] or is converted into a different representation (e.g., a set of topical categories) to build more general models and improve personalization coverage [15, 70]. Despite the value of personalization, one drawback is that it requires sufficient user information to perform effectively; users must be willing to share their search history and the search engine must attain sufficient information on user interests to build accurate profiles. Even short-term personalization depends on the long-term behavior for the first query in the session when no other behavior has been observed [15].

It is known that users frequently submit the same query to find the information they have searched previously. Teevan *et al.* [77] found that approximately 33% of query instances in their case study were an exact repeat of a query submitted by the same user at a previous time. For such re-finding queries, the results clicked by individual users in their search history provide a strong signal to identify the correct result for each user when that query is repeated [79]. The remaining 67% queries are new, and it can be challenging to improve their search quality via personalization given limited history.

One way in which these issues can be addressed is by finding cohorts of searchers who share one of more attributes with the current searcher. Attributes that could be used to form cohorts include location, topical interest, and domain preferences, all easily accessible to search engines via users' long-term search histories. Given a user, we can leverage the search behavior of other members of their cohort(s) to enhance

personalization by providing signals if sufficient information is unavailable or as an additional signal to build richer personalization models if they already exist. Cohorts have been used effectively in applications such as collaborative filtering (CF) [30], where groups of similar users (based on factors such as liking the same item [19]) can yield relevant recommendations. Cohorts have also shown some limited utility in retrieval settings. Groupization has shown promise in laboratory settings [80], task models to find those engaged in similar tasks have yielded strong results [88], and there have even been attempts to use CF more directly in search result ranking [71]. However, there has been no detailed study of applying cohort models to enhance Web-scale search personalization. We address that shortcoming in this work.

We propose the construction and application of user cohorts to enhance web search personalization. Our initial method creates pre-defined cohorts on three types: topic, location, and top level domain preference (e.g., .gov, .edu). Rather than limiting to these pre-defined sets, we also propose clustering methods capable of learning the cohorts and dynamically assigning users to one or more clusters. We demonstrate through extensive experimentation with search engine log data that our cohort modeling methods can yield significant relevance improvements over a production ranker that already included personalization targeting the current searcher. We show that these gains are even larger when we target particular queries (e.g., those with high ambiguity) and particular users (e.g., those with no query-relevant history).

1.3 Identify Reference Group for Researchers

For those few scientists who win Nobel prizes or Turing awards, their standing in their research community is unquestionable. For the rest of us, it is more complex to understand where we stand and who we are alike. One common approach is to compare people. Movie stars, CEOs, authors, and singers, are all compared on a number of dimensions. It is common to hear a new artist being introduced in terms of other artists that they are similar to or have been influenced by. In academia, it is also common to look for comparable researchers. Recommendation letters and tenure cases usually suggest other researchers who are comparable to the individual in question. In

discussing whether someone is suitable to collaborate with, we might ask who they are similar to in their research work. These comparisons can have significant influence by indicating that researchers compare favorably to others, and by providing a starting point for detailed discussions of the individual's strengths and weaknesses.

We study this problem of finding the comparable group for researchers, aiming to help researchers understand themselves. It is challenging since there is no simple strategy that allows a similar researcher to be found. Natural first attempts, such as looking at co-authors, or scouring the author's preferred publication venues (conferences or journals), either fail to find good candidates, or swamp us with too many possibilities.

In this work, we are particularly interested in comparing any pair of two researchers given their research output, as embodied by their publications over years. There are several challenges to address here. Firstly, we need suitable data and metrics. The comparison may be based on the research impact, teaching performance, funding raised or students advised. For some of these, we lack the data to support automatic comparison. Secondly, the whole career of a researcher may last several decades. During this career, she/he may be productive all the time, may take time off for a while, or switch research interests and topics. It can be difficult to find a perfect match for the whole career, and we may wish to focus on periods of greatest activity or influence.

To solve this problem, we propose a trajectory matching algorithm that compares a pair of researchers by matching their research trajectories. The trajectory is defined in three dimensions: topic, quality and time. Topic is reflected through researchers' publications and quality is measured through venue ranks. In addition to match full career path, we introduce prefix matching, to find senior researchers that are comparable at their early career stage. We apply the algorithm on scholarly data extracted from DBLP and arXiv. Empirical results show that our algorithm can identify relevant comparable groups. To make the algorithm scale well with the large data, we provide a map-reduce version of the algorithm which is further improved by grouping authors.

Chapter 2

Recommendation to Groups

Traditional recommender systems have focused on finding relevant items for a single user. However there is an increasing need to recommend items to groups of people. For instance, families may share accounts for e-commerce or movie streaming services; colleagues may dine out for lunch on almost every weekday. The problem of generating relevant items to a group of users is called group recommendation.

To date, research in group recommendation has followed two main approaches. One is aggregating results of recommendations to each member in the group [50], the other is aggregating members' preference to group preference then applying traditional recommendation techniques on the merged pseudo-user [52]. Both approaches requires sufficient data of individual and group historical behavior. When new groups join the system and new user joins the group, they fail to produce relevant recommendation.

We address two challenges in group recommendation: cold start and dynamic presence of members. More specifically, the system does not have any prior knowledge about the group nor members in the group, and members in the group can have dynamic presence in each group event. We utilize Multi-armed Bandits (MAB) framework and propose the group variant MAB. Unlike traditional MAB with a single player, our setting has a group of players with dynamic presence at each session. The learner picks an arm at each session to the group then receives the group payoff to the picked arm. More relevant the arm is to the group, higher payoffs the learner receives. We aim at designing policies to maximize the payoff from the group in sequential plays.

We consider two types of feedback: 1) each present member gives individual feedback; 2) the group gives a single aggregated feedback. We design *Group-UCB* and ϵ-*Greedy* policies for corresponding reward setting. We prove that our policies deviate

from the maximum expected cumulative payoff by at most $O(\log T)$. The rest of the chapter is structured as follows. Section 2.1 introduces the background of MAB and the problem formulation of group recommendation. Section 2.2 describes the two types of feedback, corresponding policies we designed and the main results for each policy. Section 2.3 lists related work to group recommendation and MAB. Section 2.4 has the proofs of two main results in Section 2.2. Section 2.5 presents the simulation of our two settings and their numeric results. In Section 2.6, we show a consensus-oriented group recommender demo system, and discuss the data collected from the user study through the demo system. Section 2.7 concludes the chapter.

2.1 MAB Background and Problem Formulation

In this section, we introduce the background of traditional Multi-armed bandit and the formulation of group-MAB. We also discuss a simple extension of traditional MAB to groups.

2.1.1 Multi-armed Bandits Background

A MAB algorithm is a learner who plays in sequential sessions the following game with a user u. At time t the learner has to pick an option to suggest to the user. The options are called arms and belong to a set A with K elements. The arm selected by the learner at time t is $a_t \in A$. When the user receives that arm, she returns a reward r_{a_t} which comes from an underlying stochastic satisfaction model that user has for the arm a_t. This model is unknown to the learner. We define $\theta_{u,a} = \mathbb{E}[r_{u,a}]$. The learner has to improve its arm-selection policy based on the new observation (a_t, r_{a_t}) and all previous observations which it keeps in memory. Feedback from unpicked arms is not observed. In the process, the total satisfaction achieved is defined as $\sum_{t=1}^{T} r_{a_t}$. This is called reward.

The optimal expected rewards for the user is defined as $\mathbb{E}[\sum_{t=1}^{T} r_{a^*}]$ where a^* is the arm with maximum expected reward. The goal is to design a policy of arm picking in order to maximize the expected total rewards. Equivalently, we can design a policy

to minimize the regret, $R(T)$, which is defined as the deviation in the reward from choosing the optimal arm, formally:

$$R(T) = \sum_{t=1}^{T} \theta_{a^*} - \sum_{t=1}^{T} \theta_{a_t}. \tag{2.1}$$

In the context of recommendation, we may view items to recommend as arms. The reward can be interpreted as a rating score given to the suggested item. This rating can be measured indirectly as click through rate or conversion rate depending on the definition of reward. The goal is to maximize these quantities in expectation. The bandit algorithm can learn user's preference quickly without any prior information and thus is a good fit to solve the cold-start problem that arises in recommender systems.

2.1.2 Group-MAB Formulation

We now formulate group recommendation as a group-MAB problem.

Group Representation

Let G be a group of $d = |G|$ users, whose members recurrently come together for group events. At any given time t, $S_t \subset G$ is the set of users that are together. At time t, each user u has associated a presence-weight $x_{u,t}$. The vector $x_t = \{x_{u,t}\}_u$ takes values in a finite set $\mathcal{X} \subset \mathbb{R}_+^d$ and models user's presence as well as relative influence on overall group satisfaction. Just like in traditional MAB, at every time t, each user, which is a member in the group, gives a reward $r_{u,a}$ of a suggested arm a to the learner. The rewards are random according to a model unknown to the learner. The expected value of the reward is $\theta_{u,a} = \mathbb{E}[r_{u,a}] \in \mathbb{R}_+$. Without loss of generality, we normalize $\theta_{u,a}$ and $r_{u,a}$ such that they belong to $(0, 1]$. We denote $\theta_a = \{\theta_{a,u}\}_u$ and $r_a = \{r_{a,u}\}_u$.

The *group reward* is then an aggregation of the members' satisfaction/rewards. There are multiple ways to aggregate members' rewards according to social choice theory [29], such as average strategy, plurality voting, borda count, least misery, most pleasure and so on. We adopt a weighted average strategy, which computes group reward as a linear function of rewards from members. In particular, $r_{x_t} = \sum_{u \in S_t} x_{u,t} r_{u,a_t}$,

where $x_{u,t} \in \mathbb{R}_+$ is a non-negative weight value.

In general, the vector x_t of users' weights at the time t takes values in a finite set $\mathcal{X} \subset \mathbb{R}_+^d$ and satisfies two conditions. First, absent users have zero weight, *i.e.*, if $u \notin S_t$ then $x_{u,t} = 0$. Second, $M_1 = \sup_{x \in \mathcal{X}} \|x\|_1 < \infty$, thus, weights are bounded. Under this notation, the expected group reward is given by the inner product $x_t^\dagger \theta_{a_t} = \sum_{u \in G} x_{u,t} \theta_{u,a_t}$.

This model of a group reward is fairly general. Consider, for example,

$$x_{u,t} = \begin{cases} \frac{w_u}{\sum_{v \in S_t} w_v}, & \text{if } u \in S_t \\ 0, & \text{o.w.} \end{cases} \tag{2.2}$$

where w_u are positive for all $u \in G$. If $w_u = 1$ for all $u \in G$, the group reward becomes a simple average over the rewards of users present at the t-th session. Alternatively, a highly influential individual u, whose reward greatly affects group satisfaction, can be modeled with a big weight w_u. Note that, in this case, the set of possible weight vectors $\mathcal{X}$ is finite and contains 2^d elements and $M_1 = \sup_{x \in \mathcal{X}} \|x\|_1 = 1$.

Problem Definition

The learner plays in T sequential sessions $t = 1, 2, 3, \ldots, T$. In each session, it observes the group x_t, and picks an arm a_t for the group based on information learned from past sessions. The group gives back either a single group reward r_{x_t} or a vector of individual rewards $r_{x_t} = \{r_{u,a_t}\}_u$. It will be clear from the context which definition of r_{x_t} we are using. The tuple (x_t, a_t, r_{x_t}) is then used in later sessions to improve the arm-selection policy. Let $a_t \in A$ be the arm selected by the policy at the t-th session. We call the sequence $\{a_t\}_{t=1}^T$ the *policy*.

Our goal is to design an *adaptive policy* to maximize group rewards over time. Note that the group is dynamic. Given the group represented by a vector $x \in \mathcal{X}$, an *optimal arm* a_x^* is the one that maximizes the expected group rewards, *i.e.*, $a_x^* = \arg\max_{a \in A} x^\dagger \theta_a$.

Given a policy $\{a_t\}_{t=1}^T$, we define its *regret* after T sessions to be

$$R(T) = \sum_{t=1}^{T} x_t^\dagger \theta_{a^*_{x_t}} - \sum_{t=1}^{T} x_t^\dagger \theta_{a_t} \tag{2.3}$$

where $a^*_{x_t}$ is the optimal arm for x_t. As in the classic multi-armed bandit problem, the regret is the difference between expected rewards of an optimal policy and the policy $\{a(t)\}_{t=1}^T$. In contrast to the classic setup, the optimal arm may change with each session, as it depends on the weight vector x_t.

Note that, under the above assumption, the learner *observes* the weight vector x_t, and then makes its selection a_t. For example, in the case of weights given by (2.2), the learner has access to the relative weights w_u as well as the composition of the group of present users S_t. On the other hand, the reward of users at time t is revealed *only after* the arm is suggested to the group, and thus can only be used in future arm selections.

2.1.3 A Naive Approach and Its Regret Bound

There is an immediate, but inefficient, approach to reducing the multi-armed bandit problem with groups that we have proposed to the classic setup. This amounts to treating every session at which a weight vector $x \in \mathcal{X}$ appears separately from all other sessions. In particular, consider all the sessions t for which $x_t = x$, for $x \in \mathcal{X}$. Conditioned on the arm a displayed, the group reward is independent of rewards at previous sessions. Thus, the subgroup x can be treated as an individual player, and the optimal arm for this player can be found using *e.g.*, the UCB policy by Auer *et al.* [8].

Interestingly, applying this naïve method yields logarithmic regret. In particular, the following holds.

Lemma 1. *Let $R_{\mathcal{A}}(T)$ be the regret of a (possibly random) policy $\mathcal{A}$ for the classic multi-armed bandit problem. Then, applying $\mathcal{A}$ at times $\{t : x_t = x\}$, where $x \in \mathcal{X}$, yields a regret $R(T) = \sum_{x\in\mathcal{X}} R_{\mathcal{A}}(n(x))$, where $n(x)$ is the number of times that $x_t = x \in \mathcal{X}$ up to and including time T. Moreover, there exists a sequence $\{a_t\}_{t=1}^T$ such that the regret satisfies*

$$2^d R_{\mathcal{A}}(\lfloor T/2^d \rfloor) \leq R(T) \leq 2^d R_{\mathcal{A}}(\lfloor T/2^d \rfloor + 1).$$

where d is the size of the group.

Proof. The regret can be written as

$$R(T) = \sum_{x \in \mathcal{X}} \mathbb{E}[\sum_{t=1}^{T} \mathbb{1}(x_t = x)(x^\dagger \theta_{a_x^*} - \sum_{t=1}^{T} x^\dagger \theta_{a_t})]$$
$$= \sum_{x \in \mathcal{X}} R_{\mathcal{A}}(n(x))$$

Let x_t be the average weight vector defined in (2.2) with $w_u = 1$, for all $u \in G$. Then, let $S_t \subset G$ be periodic, with period 2^d, taking all possible values in the power set of G in some predefined order. Then, $n(x)$ is either $\lfloor T/2^d \rfloor$ or $\lfloor T/2^d \rfloor + 1$, and the lemma follows. □

Thus, given a policy $\mathcal{A}$ with logarithmic regret $R_A(t)$ (as in [8]), the naïve approach also yields logarithmic regret; nevertheless, the leading constant of the regret will be the number of distinct subgroups, which can be of the order of 2^d, *i.e.*, exponential on the group capacity. This is clearly unsatisfactory. Intuitively, treating different groups separately hinders learning: rewards given by a user during her participation in one subgroup S_t are ignored when she participates in a different group.

Given that logarithmic regret is tenable, our goal is thus to produce a policy with a regret bound whose leading constant is small. If the sequence of sessions is $|S_t| = 1$ for all t, *i.e.*, all users appear alone, no policy can do better than the naïve policy, as rewards across subgroups are independent. For this case, the leading constant of the regret is $\Omega(d)$; this follows from Lemma 1.

2.2 Two Settings of Reward Process

At each session, the learner picks an arm then receives rewards from the current group to the selected arm. There are two types of reward processes. The first type is that every present member gives their rewards to the learner, thus r_{x_t} is a vector. For example, in Meetup [1], groups can organize offline events and members can submit explicit ratings to

[1] `www.meetup.com`

events they participated. The second type is that the learner receives a group reward, and r_{x_t} is a scalar. Such reward is often recognized as implicit ratings and derived from group behaviors. For example, group purchasing in Groupon [2] or LivingSocial [3], group checkin in Whrrl [4] , group listening in Last.fm [5] can be considered as positive rewards.

Next, we will discuss policies designed for each of these two reward processes and their regret bounds.

2.2.1 Receiving Individual Rewards

Group-UCB Algorithm

Algorithm 1 Group-UCB

$n_u \leftarrow 0$; $n_{u,a} \leftarrow 0$; $\hat{\theta}_{u,a} \leftarrow 0$
for $t = 1$ to T **do**
  Observe present users S_t and the weight vector x_t
  for a=1 to K **do**
    for u =1 to d **do**
      if $n_{u,a} = 0$ then $p_{u,a} \leftarrow \infty$
      else $p_{u,a} \leftarrow \hat{\theta}_{u,a} + \sqrt{\frac{2 \ln n_u}{n_{u,a}}}$
    end for
    $p_a \leftarrow \sum_u x_{u,t} * p_{u,a}$
  end for
  choose arm $a \leftarrow \arg\max_a p_a$ (break ties arbitrarily)
  observe reward r_u from each user $u \in S_t$, then update:
  $n_{u,a} \leftarrow n_{u,a} + 1$ for all $u \in S_t$
  $n_u \leftarrow n_u + 1$ for all $u \in S_t$
  $\hat{\theta}_{u,a} \leftarrow \hat{\theta}_{u,a} \cdot \frac{(n_{u,a}-1)}{n_{u,a}} + r_u \cdot \frac{1}{n_{u,a}}$
end for

In Algorithm 1, we propose the policy *Group-UCB* for the setting when we receive each present member's reward after each recommendation. The learner maintains the estimates of the quantities $\theta_{u,a}$, for all $u \in G$ and $a \in A$. The reward $r_{u,a}$ of user u to the arm a is estimated from the empirical average: $\hat{\theta}_{u,a}(s) = \frac{1}{s} \sum_{\tau=1}^{s} r_{u,a}(\tau)$. Moreover, the learner keeps track of how many times a user has participated and how many times

[2] `www.groupon.com`

[3] `www.livingsocial.com`

[4] `www.whrrl.com`

[5] `www.lastfm.com`

a particular arm has been chosen. Formally, let $\mathbb{1}(E)$ be the characteristic function of an event E (1 if E is true and zero otherwise). The learner keeps track of the number of times a user u shows up and an arm a is suggested at that time:

$$n_{u,a}(T) = \sum_{t=1}^{T} \mathbb{1}(x_{u,t} > 0 \text{ and } a_t = a),$$

as well as the number of times the user u has been present up to session T:

$$n_u(T) = \sum_{t=1}^{T} \mathbb{1}(x_{u,t} > 0).$$

Using the above quantities, the learner selects an arm as follows. At the t-th session, the learner first observes the present composition of group S_t and the associated weight vector x_t. The arm selected is given by

$$a_t = \arg\max_{a \in A} \sum_{u \in G} x_{u,t} \left(\hat{\theta}_{u,a} + \sqrt{\frac{2 \ln n_u}{n_{u,a}}} \right). \tag{2.4}$$

Then the subgroup provides the learner with individual rewards, which are then used to update the estimates $\hat{\theta}_{u,a}$ for the arm $a = a_t$ and for users $u \in S_t$.

Regret Bound for Group-UCB

The regret under Group-UCB can be bounded according to the following theorem. Its proof will be presented in Section 2.4.

Theorem 1. *Given $x \in \mathcal{X}$, denote by $B_x \subset A$ the set of suboptimal arms under x,* i.e.*, $B_x = \{a \in A : x^\dagger \theta_{a_x^*} > x^\dagger \theta_a\}$ Moreover, let $\Delta^a_{\min} = \inf_{x \in \mathcal{X}: a \in B_x} x^\dagger \theta_{a_x^*} - x^\dagger \theta_a$. Then, $R(T) \leq \sum_{a \in A} \frac{8 M_1^3 d}{(\Delta^a_{\min})^2} \ln T + 4KdM_1$ for Group-UCB.*

The proof for Theorem 1 is given in 2.4.1.

2.2.2 Receiving Group Reward

We now discuss the scenario when a single group reward is observed, that means r_{x_t} is a scalar. We treat the group as the contextual variable and turn to contextual MAB

setting. The weight vector x_t of the subgroup S_t is the observed stochastic contextual variable to the learner. We prove our main result (Theorem 3) in the stochastic setting where x_t are drawn i.i.d. from an unknown multivariate probability distribution $\mathcal{D}$. In addition, we require that the set of contexts is finite *i.e.*, $|\mathcal{X}| < \infty$. We define $\Sigma_{\min} > 0$ to be the smallest non-zero eigenvalue of the covariance matrix $\Sigma \equiv \mathbb{E}\{x_1 x_1^\dagger\}$.

After observing a context x_t and selecting an arm a_t, the learner receives a reward r_{a_t,x_t} which is drawn from a distribution p_{a_t,x_t} independently of all past contexts, actions and payoffs. We assume that the expected payoff is a linear function of the context. In other words,

$$r_{a_t,x_t} = x_t^\dagger \theta_a + \epsilon_{a,t} \tag{2.5}$$

where $\{\epsilon_{a,t}\}_{a\in\mathcal{A},t\geq 1}$ are a set of independent random variables with zero mean and $\{\theta_a\}_{a\in\mathcal{A}}$ are unknown parameters in $\mathbb{R}^d$. Note that, w.l.o.g, we can assume that $Q = \max_{a\in\mathcal{A}} \|\theta_a\|_2 \leq 1$. This is because if $Q > 1$, as payoffs are linear, we can divide all payoffs by Q; the resulting payoff is still a linear model, and our results stated below apply. A sub-gaussian random variable Z has the property that $\mathbb{E}\{e^{\gamma Z}\} \leq e^{\gamma^2 L^2}$ where L is a constant. In particular, sub-gaussianity implies $\mathbb{E}\{Z\} = 0$. We make the following technical assumption.

Assumption 1. *The random variables $\{\epsilon_{a,t}\}_{a\in\mathcal{A},t\geq 1}$ are sub-gaussian random variables with constant $L > 0$.*

Given a context x, the optimal arm is $a_x^* = \arg\max_{a\in\mathcal{A}} x^\dagger \theta_a$. The expected cumulative regret the learner experiences over T steps is defined by 2.3. The objective is to design a policy that achieves as low expected cumulative regret as possible, and also has low computational complexity. We define $\Delta_{\max} \equiv \max_{a,b\in\mathcal{A}} ||\theta_a - \theta_b||_2$, and $\Delta_{\min} \equiv \inf_{x\in\mathcal{X},a:x^\dagger\theta_a<x^\dagger\theta_{a_x^*}} x^\dagger(\theta_{a_x^*} - \theta_a) > 0$. Observe that, by the finiteness of $\mathcal{X}$ and $\mathcal{A}$, the defined infimum is attained (*i.e.*, it is a minimum) and is indeed positive.

When there exists a gap between optimal and suboptimal rewards, several algorithms have been proposed that achieve $O(\log T)$ regret after T time steps. However, proposed methods either have a computation complexity per iteration that scales linearly with T or achieve regrets that grow linearly with the number of contexts $|\mathcal{X}|$. We

propose an ϵ-greedy type of algorithm that solves both limitations. In particular, when contexts are variables in $\mathbb{R}^d$, we prove that our algorithm has a constant computation complexity per iteration of $O(\text{poly}(d))$ and can achieve a regret of $O(\text{poly}(d)\log T)$ even when $|\mathcal{X}| = \Omega(2^d)$. In addition, unlike previous algorithms, its space complexity scales like $O(Kd^2)$ and does not grow with T.

Contextual ϵ-Greedy Algorithm

We now present a simple and *efficient* on-line algorithm that, under the above assumptions, has expected *logarithmic* regret. Specifically, its computational complexity, at each time instant, is $O(Kd^3)$ and the expected memory requirement scales like $O(Kd^2)$. As far as we know, our analysis is the first to show that a simple and *efficient* algorithm for the problem of linearly parametrized bandits can, under reward separation and i.i.d. contexts, achieve logarithmic expected cumulative regret that simultaneously can scale like $polylog(|\mathcal{X}|)$ for natural scenarios.

Before we present our algorithm in full detail, let us give some intuition about it. Part of the job of the learner is to estimate the unknown parameters θ_a based on past actions, contexts and rewards. We denote the estimate of θ_a at time t by $\hat{\theta}_a$. If $\theta_a \approx \hat{\theta}_a$ then, given an observed context, the learner will more accurately know which arm to play to incur small regret. The estimates $\hat{\theta}_a$ can be constructed based on a history of past rewards, contexts and arms played. Since observing a reward r for arm a under context x does not give information about the magnitude of θ_a along directions orthogonal to x, it is important that, for each arm, rewards are observed and recorded for a rich class of contexts. This gives rise to the following challenge: If the learner tries to build this history while trying to minimize the regret, the distribution of contexts observed when playing a certain arm a will be biased and potentially not rich enough.

We address this challenge using the following idea which also appears in the epoch-Greedy algorithm of [44]. We partition time slots into *exploration* and *exploitation epochs*. In exploration epochs, the learner plays arms uniformly at random, independently of the context, and records the observed rewards. This guarantees that in the history of past events, each arm has been played along with a sufficiently rich set of

contexts. In exploitation epochs, the learner makes use of the history of events stored during exploration to estimate the parameters θ_a and determine which arm to play given a current context. The rewards observed during exploitation are not recorded.

More specifically, when exploiting, the learner performs two operations. In the first operation, for each arm $a \in \mathcal{A}$, an estimate $\hat{\theta}_a$ of θ_a is constructed from a simple ℓ_2-regularized regression, as in [6] and [20]. In the second operation, the learner plays the arm a that maximizes $x_t^\dagger \hat{\theta}_a$. Crucially, in the first operation, only information collected during exploration epochs is used. In particular, let $\mathcal{T}_{a,t-1}$ be the set of exploration epochs up to and including time $t-1$ (*i.e.*, the times that the learner played an arm a uniformly at random (u.a.r.)). Moreover, for any $\mathcal{T} \subset \mathbb{N}$, denote by $r_\mathcal{T} \in \mathbb{R}^n$ the vector of observed rewards for all time instances $t \in \mathcal{T}$, and $X_\mathcal{T} \in \mathbb{R}^{n \times d}$ is a matrix of $\mathcal{T}$ rows, each containing one of the observed contexts at time $t \in \mathcal{T}$. Then, at time t the estimator $\hat{\theta}_a$ is the solution of the following convex optimization problem.

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{2n} \|r_\mathcal{T} - X_\mathcal{T}\theta\|_2^2 + \frac{\lambda_n}{2} \|\theta\|_2^2. \tag{2.6}$$

where $\mathcal{T} = \mathcal{T}_{a,t-1}$, $n = |\mathcal{T}_{a,t-1}|$, $\lambda_n = 1/\sqrt{n}$. In other words, the estimator $\hat{\theta}_a$ is a (regularized) estimate of θ_a, based only on observations made during exploration epochs. Note that the solution to (2.6) is given by $\hat{\theta}_a = \left(\lambda_n I + \frac{1}{n} X_\mathcal{T}^\dagger X_\mathcal{T}\right)^{-1} \frac{1}{n} X_\mathcal{T}^\dagger r_\mathcal{T}$.

An important design choice is the above process selection of the time slots at which the algorithm explores, rather than exploits. Following the ideas of [73], we select the exploration epochs so that they occur approximately $\Theta(\log t)$ times after t slots. This guarantees that, at each time step, there is enough information in our history of past events to determine the parameters accurately while only incurring in a regret of $O(\log t)$. There are several ways of achieving this; our algorithm explores at each time step with probability $\Theta(t^{-1})$.

The above steps are summarized in pseudocode by Algorithm 2. Because there are K arms and for each arm $(x_t, r_{a,t}) \in \mathbb{R}^{d+1}$, the expected memory required by the algorithm scales like $O(Kd^2)$. In addition, both the matrix $X_\mathcal{T}^\dagger X_\mathcal{T}$ and the vector $X_\mathcal{T}^\dagger r_\mathcal{T}$ can be computed in an online fashion in $O(d^2)$ time: $X_\mathcal{T}^\dagger X_\mathcal{T} \leftarrow X_\mathcal{T}^\dagger X_\mathcal{T} + x_t x_t^\dagger$

Algorithm 2 Contextual ϵ -greedy

For all $a \in A$, set $A_a \leftarrow 0_{d\times d}$;$n_a \leftarrow 0$; $b_a \leftarrow 0_d$
for $t = 1$ to p **do**
  $a \leftarrow 1 + (t \mod K)$; Play arm a
  $n_a \leftarrow n_a + 1$; $b_a \leftarrow b_a + r_t x_t$; $A_a \leftarrow A_a + x_t x_t^\dagger$
end for
for $t = p + 1$ to T **do**
  $e \leftarrow$ Bernoulli(p/t)
  if $e = 1$ **then**
    $a \leftarrow$ Uniform$(1/K)$; Play arm a
    $n_a \leftarrow n_a + 1$; $b_a \leftarrow b_a + r_t x_t$; $A_a \leftarrow A_a + x_t x_t^\dagger$
  else
    for $a \in \mathcal{A}$ **do**
      Get $\hat{\theta}_a$ as the solution to the linear system: $\left(\lambda_{n_a} I + \frac{1}{n_a} A_a\right) \hat{\theta}_a = \frac{1}{n_a} b_a$
    end for
    Play arm $a_t = \arg\max_{a\in\mathcal{A}} x_t^\dagger \hat{\theta}_a$
  end if
end for

and $X_{\mathcal{T}}^\dagger r_{\mathcal{T}} \leftarrow X_{\mathcal{T}}^\dagger r_{\mathcal{T}} + r_t x_t$. Finally, the estimate of $\hat{\theta}_a$ requires solving a linear system (see Algorithm 2), which can be done in $O(d^3)$ time. Note that the algorithm contains a scaling parameter p, which is specified in 2.2.2.

Regret Bounds for Contextual ϵ-Greedy

The above analysis of the algorithm can be summarized in following theorems.

Theorem 2. *Algorithm 2 has computational complexity of $O(Kd^3)$ per iteration and its expected space complexity scales like $O(Kd^2)$.*

We now state our theorem that shows that Algorithm 2 achieves $R(T) = O(\log T)$.

Theorem 3. *Under Assumptions 1, the expected cumulative regret of algorithm 2 satisfies,*

$$R(T) \leq p\Delta_{\max}\sqrt{d} + 14\Delta_{\max}\sqrt{d}Ke^{Q/4} + p\Delta_{\max}\sqrt{d}\log T.$$

for any

$$p \geq \frac{CKL'^2}{(\Delta'_{\min})^2(\Sigma'_{\min})^2}. \tag{2.7}$$

Above, C is a universal constant, $\Delta'_{\min} = \min\{1, \Delta_{\min}\}$, $\Sigma'_{\min} = \min\{1, \Sigma_{\min}\}$ and $L' = \max\{1, L\}$.

The proof for Theorem 3 is given in 2.4.2. In Theorem 3, the bound on the regret depends on p - small p is preferred - and hence it is important to understand how the right hand side (r.h.s.) of (2.7) might scale when K and d grow. In below example, we show that, for a concrete distribution of contexts and choice of expected rewards θ_a, and assuming (2.7) holds, $p = O(K^3 d^5)$ [6] . There is nothing special about the concrete details of how contexts and θ_a's are chosen and, although not included here, for many other distributions, one also obtains $p = O(\text{poly}(d))$. We can certainly construct pathological cases where, for example, p grows exponentially with d. However, we do not find these intuitive.

Example of Scaling of p with d and K

Assume that contexts are obtained by normalizing a d-dimensional vector with i.i.d. entries as Bernoulli random variables with parameter w. Assume in addition that every θ_a is obtained i.i.d. from the following prior distribution: every entry of θ_a is drawn i.i.d. from a uniform distribution and then θ_a is normalized. Finally, assume that the payoffs are given by $r_{a,t} = x_t^\dagger \Theta_a$, where $\Theta_a \in \mathbb{R}^d$ are random variables that fluctuate around $\theta_a = \mathbb{E}\{\Theta_a\}$ with each entry fluctuating by at most F.

Under these assumptions the following is true:

- $\Sigma_{\min} = \Omega(d^{-1})$. In fact, the same result holds asymptotically independently of $w = w(d)$ if, for example, we assume that on average groups are roughly of the same size, M, with $w = M/d$;
- $L = O(\sqrt{d})$. This holds because $\epsilon_{a,t} = r_{a,t} - \mathbb{E}\{r_{a,t}\} = x_t^\dagger(\Theta_a - \theta_a)$ are bounded random variables with zero mean and $\|x_t^\dagger(\Theta_a - \theta_a)\}\|_\infty = O(\sqrt{d})$.
- $\Delta_{\min} = \Omega(1/(Kd\sqrt{w})$ with high-probability (for large K and d). This can be seen as follows, if $\Delta_{\min} = x^\dagger(\theta_a - \theta_b)$ for some x, a and b, then it must be true

[6] This bound holds with probability converging to 1 as K and d get large

that θ_a and θ_b differ in a component for which x is non-zero. The minimum difference between components among all pairs of θ_a and θ_b is lower bounded by $\Omega(1/(K\sqrt{d}))$ with high probability (for large K and d). Taking into account that each entry of x is $O(1/\sqrt{dw})$ with high-probability, the bound on $\Delta_{\min}$ follows.

If we want to apply Theorem 3 then (2.7) must hold and hence putting all the above calculations together we conclude that $p = O(K^3 d^5)$ with high probability for large K and p.

Computing p in Practice

If we have knowledge of an a priori distribution for the contexts, for the expected payoffs and for the variance of the rewards then we can quickly compute the value of $\Sigma_{\min}$, L and a typical value for $\Delta_{\min}$. An example of this was done above. There, the values were presented only in order notation but exact values are not hard to obtain for that and other distributions. Since a suitable p only needs to be larger then the r.h.s. of (2.7), by introducing an appropriate multiplicative constant, we can produce a p that satisfied (2.7) with high probability.

If we have no knowledge of any model for the contexts or expected payoffs, it is still possible to find p by estimating $\Delta_{\min}$, $\Sigma_{\min}$ and L from data gathered while running Algorithm 2. Notice again that, since all that is required for our theorem to hold is that p is greater then a certain function of these quantities, an exact estimation is not necessary. This is important because, for example, accurately estimating $\Sigma_{\min}$ is hard when matrix $\mathbb{E}\{x_1 x_1^\dagger\}$ has a large condition number.

Not being too concerned about accuracy, $\Sigma_{\min}$ can be estimated from $\mathbb{E}\{x_1 x_1^\dagger\}$, which can be estimated from the sequence of observed x_t. $\Delta_{\min}$ can be estimated from Algorithm 2 by keeping track of the smallest difference observed until time t between $\max_b x^\dagger \hat{\theta}_b$ and the second largest value of the function being maximized. Finally, the constant L can be estimated from the variance of the observed rewards for the same (or similar) contexts. Together, these estimations do not incur in any significant loss in computational performance of our algorithm.

Adversarial Setting

In the stochastic setting, the richness of the subset of $\mathbb{R}^d$ spanned by the observed contexts is related to the skewness of the distribution $\mathcal{D}$. The fact that the bound in Theorem 3 depends on $\Sigma_{\min}$ and that the regret increases as this value becomes smaller indicates that our approach does not yield a $O(\log T)$ regret for the adversarial setting, where an adversary choses the contexts and can, for example, generate $\{x_t\}$ from a sequence of stochastic processes with decreasing $\Sigma_{\min}(t)$.

In particular, the main difficulty in using a linear regression, and the reason why our result depends on $\Sigma_{\min}$, is related to the dependency of our estimation of $x_t^\dagger \theta_a$ on $\frac{1}{|\mathcal{T}_{a,t-1}|} X_{\mathcal{T}_{a,t-1}}^\dagger X_{\mathcal{T}_{a,t-1}}$. It is not hard to show that the error in approximating $x_t^\dagger \theta_a$ with $x_t^\dagger \hat{\theta}_a$ is proportional to

$$\sqrt{x_t^\dagger \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-2} x_t}. \tag{2.8}$$

This implies that, even if a given context has been observed relatively often in the past, the algorithm can "forget" it because of the *mean* over contexts that is being used to produce estimates of $x_t^\dagger \theta_a$ (the mean shows up in (2.8) as $\frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}$).

The effect of this phenomenon on the performance of Algorithm 2 can be readily seen in the following pathological example. Assume that $\mathcal{X} = \{(1,1),(1,0)\} \subset \mathbb{R}^2$. Assume that the contexts arrive in the following way: $(1,1)$ appears with probability $1/z$ and $(1,0)$ appears with probability $1 - 1/z$. The correlation matrix for this stochastic process is $\{(1,1/z),(1/z,1/z)\}$ and its minimum eigenvalue scales like $O(1/z)$. Hence, the regret scales as $O(z^2 \log T)$. If I is allowed to slowly grow with t, we expect that our algorithm will not be able to guarantee a logarithmic regret (assuming that our upper bound is tight). In other words, although $(1,1)$ might have appeared a sufficient number of times for us to be able to predict the expected reward for this context, Algorithm 2 performs poorly since the mean (2.8) will be 'saturated' with the context $(1,0)$ and forget about $(1,1)$.

One solution for this problem is to ignore some past contexts when building an estimate for $x_t^\dagger \theta_a$, by including in the mean (2.8) past contexts that are closer in direction to the current context x_t. Having this in mind, and building on the ideas of [6], we

Algorithm 3 Contextual UCB

for $t = 1$ to p **do**
  $a \leftarrow 1 + (t \mod K)$; Play arm a; $\mathcal{T}_{a,t} \leftarrow \mathcal{T}_{a,t-1} \cup \{t\}$
end for
for $t = p + 1$ to T **do**
  for $a \in \mathcal{A}$ **do**
    $c_{a,t} \leftarrow \min_{\mathcal{T} \subset \mathcal{T}_{a,t-1}} \frac{\log t}{|\mathcal{T}|} x_t^\dagger \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-2} x_t$
    $\mathcal{T}^* \leftarrow$ subset of $\mathcal{T}_{a,t-1}$ that achieves the minimum; $n \leftarrow |\mathcal{T}^*|$
    Get $\hat{\theta}_a$ as the solution to the linear system: $\left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right) \hat{\theta}_a = \left(\frac{1}{n} X_{\mathcal{T}}^\dagger r_{\mathcal{T}}\right)$
  end for
  Play arm $a_t = \arg\max_a x_t^\dagger \hat{\theta}_a + \sqrt{c_{a,t}}$; Set $\mathcal{T}_{a,t} \leftarrow \mathcal{T}_{a,t-1} \cup \{t\}$
end for

propose the UCB-type Algorithm 3.

It is straightforward to notice that this algorithm can not be implemented in an efficient way. In particular, the search for $\mathcal{T}^* \subset \mathcal{T}_{a,t-1}$ has a computational complexity exponential in t. The challenge is to find an efficient way of approximating $\mathcal{T}^*$ efficiently. This can be done by either reducing the size of $\mathcal{T}_{a,t-1}$ – the history from which one wants to extract $\mathcal{T}_{a,t-1}$ – by not storing all events in memory (for example, if we can guarantee that $|\mathcal{T}_{a,t}| = O(\log t)$ then the complexity of the above algorithm at time step t is $O(t)$), or by finding an efficient algorithm of approximating the minimization over the $\mathcal{T}_{a,t-1}$ (or both). It remains an open problem to find such an approximation scheme and to prove that it achieves $O(\log T)$ regret for a setting more general than the i.i.d. contexts considered in this paper.

2.3 Related Work

2.3.1 Group Recommendation

Prior work on group recommendation has been inspired by many different applications serving groups. Examples include recommending TV programs [52], sightseeing tours [49, 43], museum visit [37], restaurants [53], music listening in the gym [40] and so on. Past research on group recommendation has focused on identifying an "objective" function to model group satisfaction. The authors in [52] examined strategies inspired by social choice theory [29], such as "the Average", "the Average without Misery" and

"the Least Misery" and so on. Overall, "the average" is most often adopted strategy.

The authors in [68] proposed the semantic of group recommendation which is a combination of item relevance and the disagreement among members. Items with large relevance score and small disagreement are preferred. There are two main approaches of utilizing traditional recommendation techniques for group recommendation. One is aggregating individual's results [50], and the other is aggregating user profiles to a group profile, then applying recommendation to this pseudo-user [67].

To the best of our knowledge, no prior work addresses the issues of group recommendation we address, which are cold start problem and the dynamics of member presence of a group over time. We adopted the multi-arm bandit approach to address these issues. The theory of multi-arm bandits is extensive, with myriad versions studied from many arms, to delays, dependence among the arms, and so on.[7] Still, our variation, with users appearing in multiple, different groups over time, seems new, as is our algorithm and its analysis.

Our problem formulation of group MAB is closely related to so-called *linear* or *contextual* bandits [7, 26, 48, 45], which have recently been applied to personalized recommendation [48] (though aimed at individual users and not groups). In contextual bandits, the reward of an arm can be expressed as an inner product of an observable *context vector* and a set of *latent variables*. This is the case for the "group rating" we define here: the "weight vectors" $x(t)$ introduced in our model serve the role of context, while individual ratings correspond to latent variables.

Our Group-UCB policy departs from contextual bandits by having access not only to the final group rating, but also to individual rating of users, which are latent and not observed in the standard contextual bandit model. This in turn, allows us to obtain better bound in the regret (which is typically not logarithmic for contextual bandits). Our MAB setting when receiving group reward is contextual bandits with stochastic context. In next section, we will discuss related work to this setting in detail.

---

[7] A recent tutorial is at `http://techtalks.tv/talks/54451/`.

2.3.2 Stochastic Contextual Bandits

Langford and Zhang [44] propose an algorithm called *epoch-Greedy* for general contextual bandits. Their algorithm achieves an $O(\log T)$ regret in the number of time steps T in the *stochastic* setting, in which contexts are sampled from an unknown distribution in an i.i.d. fashion. Unfortunately, the proposed algorithm and subsequent improvements [27] have high computational complexity. Selecting an arm at time step t requires making a number of calls to a so-called *optimization oracle* that grows polynomially in T. In addition, the cost of an implementation of this optimization oracle can grow linearly in $|\mathcal{X}|$ in the worst case; this is prohibitive in many interesting cases, including the case where $|\mathcal{X}|$ is exponential in the dimension of the context. In addition, both algorithms proposed in [44] and [27] require keeping a history of observed contexts and arms chosen at every time instant. Hence, their space complexity grows linearly in T.

In the paper [44], it assumes that the context $x \in \mathcal{X}$ is sampled from a probability distribution $p(x)$ and that, given an arm $a \in \mathcal{A}$, and conditioned on the context x, rewards r are sampled from a probability distribution $p_a(r \mid x)$. A significant challenge in their setup is that, though contexts x are sampled independently, they are not independent conditioned on the arm played: an arm will tend to be selected more often in contexts in which it performs well. Hence, learning the distributions $\{p_a(r \mid x)\}_{a \in \mathcal{A}}$ from such samples is difficult. The epoch-Greedy algorithm [44] deals with this by separating the exploration and exploitation phase, effectively selecting an arm uniformly at random at certain time slots (the exploration "epochs"), and using samples collected only during these epochs to estimate the payoff of each arm in the remaining time slots (for exploitation).

We show that the challenges above can be addressed when rewards are linear. In the above contextual bandit set up, this means that $\mathcal{X}$ is a subset of $\mathbb{R}^d$, and the expected reward of an arm $a \in \mathcal{A}$ is an unknown linear function of the context x, *i.e.*, it has the form $x^\dagger \theta_a$, for some unknown vector θ_a. Our algorithm is inspired by the work of [5] on the ϵ-greedy algorithm and the use of linear regression to estimate the parameters θ_a. The main technical innovation is the use of matrix concentration bounds to control the

error of the estimates of θ_a in the stochastic setting. We believe that this is a powerful realization and may ultimately help us analyze richer classes of payoff functions.

Our algorithm uses the same separation in "epochs" as in [44]. The paper establishes an $O(T^{2/3}(\ln|\mathcal{X}|)^{1/3})$ bound on the regret for epoch-Greedy in their stochastic setting. They further improve this to $O(\log T)$ when a lower bound on the gap between optimal and suboptimal arms in each context exists, *i.e.*, under *arm separation*. Unfortunately, the price of the generality of the framework in [44] is the high computational complexity when selecting an arm during an exploitation phase. In a recent improvement [27], this computation requires a $poly(T)$ number of calls to an optimization oracle.

Most importantly, even in the linear case we study here, there is no clear way to implement this oracle in sub-exponential time in d, the dimension of the context. As Dudik *et al.* [27] point out, the optimization oracle solves a so-called cost-sensitive classification problem. In the particular case of linear bandits, the oracle thus reduces to finding the "least-costly" linear classifier. This is hard, even in the case of only two arms: finding the linear classifier with the minimal number of errors is NP-hard [39], and remains NP hard even if an approximate solution is required [11]. As such, a different approach is warranted under linear rewards.

Contextual bandits with linear rewards is a special case of the classic linear bandit setup [6, 20, 47, 63]. In this setup, the arms themselves are represented as vectors, *i.e.*, $\mathcal{A} \subset \mathbb{R}^d$, and, in addition, the set $\mathcal{A}$ can change from one time slot to the next. The expected payoff of an arm a with vector x_a is given by $x_a^\dagger\theta$, for some unknown vector $\theta \in \mathbb{R}^d$, *common among all arms*.

There are several different variants of the above linear model. Auer [6], Li *et al.* [47], and Chu *et al.* [20] study this problem in the adversarial setting, assuming a finite number of arms $|A|$. In the adversarial setting, contexts are not sampled i.i.d. from a distribution but can be an arbitrary sequence, for example, chosen by an adversary that has knowledge of the algorithm and its state variables. Both algorithms studied, LinRel and LinUCB, are similar to ours in that they use an upper confidence bound and both estimate the unknown parameters for the linear model using a least-square-error type method. In addition, both methods apply some sort of regularization. LinRel

does it by truncating the eigenvalues of a certain matrix and LinUCB by using ridge regression. In the adversarial setting, and with no arm separation, the regret bounds obtained of the form $O(\sqrt{T}polylog(T))$.

Dani *et al.* [24], Rusmevichientong and Tsitsiklis [63], and Abbasi-Yadkori *et al.* [1] study contextual linear bandits in the stochastic setting, in the case where $\mathcal{A}$ is a fixed but possibly uncountable bounded subset of $\mathbb{R}^d$. Dani *et al.* [24] obtain regret bounds of $O(\sqrt{T})$ for an infinite number of arms; under arm separation, by introducing a gap constant Δ, their bound is $O(d^2(\log T)^3)$. Rusmevichientong and Tsitsiklis [63] also study the regret under arm separation and obtain a $O(\log(T))$ bound that depends exponentially on d. Finally, Abbasi-Yadkori *et al.* [1] obtain a $O(\text{poly}(d)\log^2(T))$ bound under arm separation.

Our problem can be expressed as a special case of the linear bandits setup by taking $\theta = [\theta_1; \ldots; \theta_K] \in \mathbb{R}^{Kd}$, where $K = |\mathcal{A}|$, and, given context x, associating the i-th arm with an appropriate vector of the form $x_{a_i} = [0 \ldots x \ldots 0]$. As such, all of the bounds described [6, 47, 20, 24, 63, 1] can be applied to our setup. However, in our setting, arms are uncorrelated; the above algorithms do not exploit this fact. Our algorithm indeed exploits this to obtain a *logarithmic* regret, while also scaling well in terms of the dimension d.

Several papers study contextual linear bandits under different notions of regret. For example, Dani *et al.* [23] define regret based on the worst sequence of loss vectors. In our setup, this corresponds to the rewards coming from an arbitrary temporal sequence and not from adding noise to $x^\dagger\theta_a$, resembling the 'worst-case' regret definition of [9]. Abernethy *et al.* [2] assume a notion of regret with respect to a best choice fixed in time that the player can make from a fixed set of choices. However, in our case, the best choice changes with time t via the current context. This different setup yields worse bounds than the ones we seek: for both stochastic and adversarial setting the regret is $O(\sqrt{T}polylog(T))$.

Recent studies on multi-class prediction using bandits [41, 33, 22] have some connections to our work. In this setting, every context x has an associated label y that a learner tries to predict using a linear classifier of the type $\hat{y} = \arg\max_a x^\dagger\theta_a$. Among

algorithms proposed, the closest to ours is by Crammer and Gentile [22], which uses an estimator for $\{\theta_a\}$ that is related to LinUCB, LinRel and our algorithm. However, the multi-class prediction problem differs in many ways from our setting. To learn the vectors θ_a, the learner receives a one-bit feedback indicating whether the label predicted is correct (*i.e.*, the arm was maximal) or not. In contrast, in our setting, the learner directly observes $x^\dagger\theta_a$, possibly perturbed by noise, without learning if it is maximal.

Finally, bandit algorithms relying on experts such as EXP4 [10] and EXP4.P [16] can also be applied to our setting. These algorithms require a set of policies (experts) against which the regret is measured. Regret bounds grow as $\log^C N$, where N is the number of experts and C a constant. The trivial reduction of our problem to EXP4(.P) assigns an expert to each possible context-to-arm mapping. The 2^d contexts in our case lead to K^{2^d} experts, an undesirable exponential growth of regret in d; a better choice of experts is a new problem in itself.

2.4 Proofs

2.4.1 Proof for Theorem 1

There are several important observations regarding Theorem 1. To begin with, the bound holds *for arbitrary sequences* $x_t \in \mathcal{X}$: irrespectively of which subsets of users show up, and how the ratings of these users are weighted, the regret will be logarithmic and the bound applies. Compared to the bound of the regret of UCB for the classic bandit problem appearing in [8], Theorem 1 differs by a multiplicative factor of $dM_1^3/\Delta^a_{\min}$. The constant M_1 is a bound on the sum of weights of all users. For practical purpose, this ought to be small; for example, when the group rating is a weighted average as in (2.2), $M_1 = 1$. Moreover, the constant $\Delta^a_{\min}$ captures the gap in expected group rewards between optimal and suboptimal arms. Note that, since $\mathcal{X}$ is finite, $\Delta^a_{\min}$ are bounded away from zero, for all $a \in A$. Similar quantities also appear in the bound of [8], but not in quadratic form. Intuitively, the smaller such gaps are, the more trials the policy needs to differentiate an optimal arm from suboptimal arms; this indeed manifests in the bound through the quantity $\Delta^a_{\min}$.

Recall that $B_x = \{a \in A : x^\dagger \theta_{a_x^*} > x^\dagger \theta_a\}$, and $\Delta^a_{\min} = \inf_{x \in \mathcal{X}: a \in B_x} x^\dagger \theta_{a_x^*} - x^\dagger \theta_a$. Let $\{a_t\}_{t=1}^T$ be the sequence of arms selected by the Group-UCB policy at sessions $t = 1, \ldots, T$. The regret $R(T)$ over the sequence $\{a_t\}_{t=1}^T$ can be written as

$$R(T) = \mathbb{E}\big[\sum_{t=1}^{T} \sum_{a \in A} \mathbb{1}(a \in B_{x_t}, a_t = a)(x_t^\dagger \theta_{a_{x_t}^*} - x_t^\dagger \theta_a)\big]$$

Since $\theta_{u,a} \in [0,1]$ for all u, a and $\|x\|_1 \leq M_1$ for all $x \in \mathcal{X}$, we have that for $a \in B_x$, $\sum_u x_u[\theta_{u,a_x^*} - \theta_{u,a}] = \big|\sum_u x_u[\theta_{u,a_x^*} - \theta_{u,a}]\big| \leq M_1$ where the positivity of the l.h.s. follows by the sub-optimality of a. We thus have

$$R(T) \leq M_1 \sum_{a \in A} \mathbb{E}[z_a(T)] \tag{2.9}$$

where $z_a(T) = \sum_{i=1}^T \mathbb{1}(a \in B_{x_t}, a_t = a)$ is the number of times the arm a was played and was suboptimal. We thus focus on bounding $\mathbb{E}[z_a(T)]$. Fix an arm $a \in A$, and denote by $\Xi(t)$ the event $\{a \in B_{x_t}, a(t) = a\}$ and by $E(t)$ the event

$$E(t) := \{ \sum_{v \in G} x_{v,t}[\hat{\theta}_{v,a_{x_t}^*}(t) + c(n_v(t), n_{v,a_{x_t}^*}(t))] \quad \leq \sum_v x_{v,t}[\hat{\theta}_{v,a}(t) + c(n_v(t), n_{v,a}(t))]\}$$

where $c(t,s) = \sqrt{\ln t/s}$, and $\hat{\theta}_{v,a}(t)$ is the estimation of $\theta_{v,a}$ up to time t, more accurately, up to $n_{v,a}(t)$ the number of times $r_{v,a}$ obtained. Then $\Xi(t) \subseteq E(t)$, as $a_t = a$ implies that the arm a had a higher UCB p_a than the optimal one. Hence $z_a(T) = \sum_{t=1}^T \mathbb{1}\left(\Xi(t) \cap E(t)\right)$. Observe that $E(t)$ implies that at least one of the following three events must be true:

$$\begin{aligned}
E_1(t) &:= \sum_v x_{v,t}[\hat{\theta}_{v,a}(t) - c(n_v(t), n_{v,a}(t))] \geq \sum_v x_{v,t}\theta_{v,a} \\
E_2(t) &:= \sum_v x_{v,t}[\hat{\theta}_{v,a_{x_t}^*}(t) + c(n_v(t), n_{v,a_{x_t}^*}(t))] \leq \sum_v x_v(t)\theta_{u,a_{x_t}^*} \\
E_3(t) &:= \sum_v x_{v,t}(\theta_{v,a_{x_t}^*} - \theta_{v,a}) < 2\sum_v x_{v,t} c(n_v(t), n_{v,a}(t))
\end{aligned}$$

Therefore

$$\mathbb{1}(\Xi \cap E) \leq \big[\mathbb{1}(E_1 \cap \Xi) + \mathbb{1}(E_2 \cap \Xi) + \mathbb{1}(E_3 \cap \Xi)\big] \tag{2.10}$$

Let's consider $E_3(t)$ first. Define $l = \frac{8M_1^2}{(\Delta^a_{min})^2} \ln T$. Then $\mathbb{1}(E_3(t) \cap \Xi(t))$ is $\mathbb{1}\big(E_3(t) \cap \Xi(t) \cap (\exists w \in S_t : n_{w,a}(t) \leq l)\big) + \mathbb{1}\big(E_3(t) \cap \Xi(t) \cap (\forall w \in S_t n_{w,a}(t) > l)\big)$. Note that $\sum_{t=1}^T \mathbb{1}\big(\Xi(t) \cap (\exists w \in S_t : n_{w,a}(t) \leq l)\big) \leq dl$. To see this, we observe that $L(t) = \sum_{w:n_{w,a}(t)\leq l} l - n_{w,a}(t)$ decreases by at least one at t for which $a(t) = a$ and $\exists w \in S_t$ s.t. $n_{w,a}(t) \leq l$. Hence, it becomes zero after at most dl such events. On the other hand, $\mathbb{1}(E_3(t) \cap (\forall w \in S_t\ n_{w,a}(t) > l)) = 0$. This is because

$$
\begin{aligned}
&\sum_v x_v(t) \left(\theta_{v,a^*_{x_t}} - \theta_{v,a}\right) - 2\sum_v x_{v,t} c(n_v(t), n_{v,a}(t)) \\
&\geq \Delta^a_{min} - 2\sum_v x_{v,t} c(n_v(t), n_{v,a}(t)) \quad \text{because } a \in B_{x_t} \\
&> \Delta^a_{min} - 2\sum_v x_{v,t} \sqrt{\frac{2\ln n_v(t)}{l}} \quad \text{because } n_{v,a}(t) > l \\
&= \Delta^a_{min} - 2\sum_v x_{v,t} \sqrt{\frac{2(\Delta^a_{min})^2 \ln n_v(t)}{8M_1^2 \ln T}} \geq 0
\end{aligned}
$$

Hence, $\sum_{t=1}^T \mathbb{1}(E_3(t) \cap \Xi(t)) \leq dl$. Moreover, $\mathbb{1}(E_1(t))$ is bounded above by

$$
\begin{aligned}
&\sum_{v:x_{v,t}>0} \mathbb{1}\Big(x_{v,t}[\hat{\theta}_{v,a}(t) - c(n_v(t), n_{v,a}(t))] \geq x_v(t)\theta_{u,a}\Big) \\
&\leq \sum_{v:x_{v,t}>0} \sum_{s=1}^{n_v(t)} \mathbb{1}\big(\hat{\theta}_{v,a}(s) - c(n_v(t), s) \geq \theta_{u,a}\big)
\end{aligned}
$$

By the Chernoff-Hoeffding bound,

$$
\mathbb{P}(\hat{\theta}_{v,a}(s) - c(n_v(t), s) \geq \theta_{u,a}) \leq e^{-4\ln n_v(t)} \leq n_v^{-4}(t)
$$

Thus, $\sum_{t=1}^T \mathbb{P}(E1(t)) \leq \sum_{t=1}^T \sum_{v:x_{v,t}>0} \sum_{s=1}^{n_v(t)} n_v^{-4}(t) \leq 2d$. One can similarly bound $\sum_{t=1}^T \mathbb{P}(E_2(t))$. Thus, (2.9), (2.10) imply

$$
R(T) \leq M_1 \sum_{a\in A} (dl + 4d) = \sum_a \frac{8M_1^3 d}{(\Delta^a_{\min})^2} \ln T + 4KdM_1.
$$

which concludes the proof.

2.4.2 Proof for Theorem 3

The general structure of the proof of our main result follows that of [5]. The main technical innovation is the realization that, in the setting when the contexts are drawn i.i.d. from some distribution, a standard matrix concentration bound allows us to treat $\lambda_n I + n^{-1}(X_{\mathcal{T}}^\dagger X_{\mathcal{T}})$ in Algorithm 2 as a deterministic positive-definite symmetric matrix, even as $\lambda_n \to 0$.

Let $\mathcal{E}_T$ denote the time instances for $t > p$ and until time T in which the algorithm took an exploitation decision. Recall that, by Cauchy-Schwarz inequality, $x_t^\dagger(\theta_{a^*_{x_t}} - \theta_a) \leq \|x_t\|_1 \|(\theta_{a^*_{x_t}} - \theta_a)\|_\infty \leq \sqrt{d}\|x_t\|_2\|(\theta_{a^*_{x_t}} - \theta_a)\|_\infty \leq \sqrt{d}\Delta_{\max}$. In addition, recall that $\sum_{t=2}^T 1/t \leq \log T$. For $R(T)$ the cumulative regret until time T, we can write

$$
\begin{aligned}
R(T) = \mathbb{E}[\sum_{t=1}^T x_t^\dagger(\theta_{a^*_{x_t}} - \theta_a)] &\leq p\Delta_{\max}\sqrt{d} + \Delta_{\max}\sqrt{d}\mathbb{E}[\sum_{t=p+1}^T \mathbb{1}\{x_t^\dagger\theta_a < x_t^\dagger\theta_{a^*_{x_t}}\}] \\
&\leq p\Delta_{\max}\sqrt{d} + \Delta_{\max}\sqrt{d}\mathbb{E}[|\mathcal{E}_T|] + \Delta_{\max}\sqrt{d}\mathbb{E}[\sum_{t\in\mathcal{E}_T} \mathbb{1}\{x_t^\dagger\theta_a < x_t^\dagger\theta_{a^*_{x_t}}\}] \\
&\leq p\Delta_{\max}\sqrt{d} + p\Delta_{\max}\sqrt{d}\log T + \Delta_{\max}\sqrt{d}\mathbb{E}[\sum_{t\in\mathcal{E}_T} \mathbb{1}\{x_t^\dagger\theta_a < x_t^\dagger\theta_{a^*_{x_t}}\}] \\
&\leq p\Delta_{\max}\sqrt{d} + p\Delta_{\max}\sqrt{d}\log T + \Delta_{\max}\sqrt{d}\mathbb{E}[\sum_{t\in\mathcal{E}_T}\sum_{a\in\mathcal{A}} \mathbb{1}\{x_t^\dagger\hat{\theta}_a > x_t^\dagger\hat{\theta}_{a^*_{x_t}}\}].
\end{aligned}
$$

In the last line we used the fact that when exploiting, if we do not exploit the optimal arm $a^*_{x_t}$, then it must be the case that the estimated reward for some arm a, $x_t^\dagger\hat{\theta}_a$, must exceed that of the optimal arm, $x_t^\dagger\hat{\theta}_{a^*_{x_t}}$, for the current context x_t.

We can continue the chain of inequalities and write,

$$
R(T) \leq p\Delta_{\max}\sqrt{d} + p\Delta_{\max}\sqrt{d}\log T + \Delta_{\max}\sqrt{d}K\sum_{t=1}^T \mathbb{P}\{x_t^\dagger\hat{\theta}_a > x_t^\dagger\hat{\theta}_{a^*_{x_t}}\}.
$$

The above expression depends on the value of the estimators for time instances that might or might not be exploitation times. For each arm, these are computed just like in Algorithm 2, using the most recent history available. The above probability depends on the randomness of x_t and on the randomness of recorded history for each arm.

Since $x_t^\dagger(\theta_{a^*_{x_t}} - \theta_a) \geq \Delta_{\min}$ we can write

$$\mathbb{P}\{x_t^\dagger \hat{\theta}_a > x_t^\dagger \hat{\theta}_{a^*_{x_t}}\} \leq \mathbb{P}\Big\{x_t^\dagger \hat{\theta}_a \geq x_t^\dagger \theta_a + \frac{\Delta_{\min}}{2}\Big\} + \mathbb{P}\Big\{x_t^\dagger \hat{\theta}_{a^*_{x_t}} \leq x_t^\dagger \theta_{a^*_{x_t}} - \frac{\Delta_{\min}}{2}\Big\}.$$

We now bound each of these probabilities separately. Since their bound is the same, we focus only on the first probability.

Substituting the definition of $r_a(t) = x_t^\dagger \theta_a + \epsilon_{a,t}$ into the expression for $\hat{\theta}_a$ one readily obtains,

$$(\hat{\theta}_a - \theta_a) = \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-1} \left(\frac{1}{n} \sum_{\tau \in \mathcal{T}} x_\tau \epsilon_{a,\tau} - \lambda_n \theta_a\right).$$

We are using again the notation $\mathcal{T} = \mathcal{T}_{a,t-1}$ and $n = |\mathcal{T}|$. From this expression, an application of Cauchy-Schwarz's inequality and the triangular inequality leads to,

$$\begin{aligned} |x_t^\dagger(\hat{\theta}_a - \theta_a)| &= \left|x_t^\dagger \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-1} \left(\frac{1}{n} \sum_{\tau \in \mathcal{T}} x_\tau \epsilon_{a,\tau} - \lambda_n \theta_a\right)\right| \\ &\leq \sqrt{x_t^\dagger \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-2} x_t} \left(\left|\frac{1}{n} \sum_{\tau \in \mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\right| + \lambda_n |x_t^\dagger \theta_a|\right). \end{aligned}$$

We introduce the following notation

$$c_{a,t} \equiv \sqrt{x_t^\dagger \left(\lambda_n I + \frac{1}{n} X_{\mathcal{T}}^\dagger X_{\mathcal{T}}\right)^{-2} x_t}. \tag{2.11}$$

Note that, given a and t both n and $\mathcal{T}$ are well specified.

We can now write,

$$\begin{aligned} &\mathbb{P}\Big\{x_t^\dagger \hat{\theta}_a \geq x_t^\dagger \theta_a + \frac{\Delta_{\min}}{2}\Big\} \leq \mathbb{P}\Big\{\Big|\frac{1}{n} \sum_{\tau \in \mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\Big| \geq \frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n |x_t^\dagger \theta_a|\Big\} \\ &\leq \mathbb{P}\Big\{\Big|\frac{1}{n} \sum_{\tau \in \mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\Big| \geq \frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n Q\Big\}. \end{aligned}$$

Since $\epsilon_{a,\tau}$ are sub-gaussian random variables with sub-gaussian constant upper bounded by L and since $|x_t^\dagger x_\tau| \leq 1$, conditioned on x_t, $\mathcal{T}$ and $\{x_\tau\}_{\tau \in \mathcal{T}}$, each $x_t^\dagger x_\tau \epsilon_{a,\tau}$ is a sub-gaussian random variable and together they form a set of i.i.d. sub-gaussian random

variables. One can thus apply standard concentration inequality and obtain,

$$\mathbb{P}\Big\{\Big|\frac{1}{n}\sum_{\tau\in\mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\Big| \geq \frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n Q\Big\} \leq \mathbb{E}\Big[2e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}}-\lambda_n Q\right)^{+2}}\Big]. \tag{2.12}$$

where both n and $c_{a,t}$ are random quantities and $z^+ = z$ if $z \geq 0$ and zero otherwise.

We now upper bound $c_{a,t}$ using the following fact about the eigenvalues of any two real-symmetric matrices M_1 and M_2: $\lambda_{\max}(M_1^{-1}) = 1/\lambda_{\min}(M_1)$ and $\lambda_{\min}(M_1+M_2) \geq \lambda_{\min}(M_1) - \lambda_{\max}(M_2) = \lambda_{\min}(M_1) - \|M_2\|$.

$$c_{a,t} \leq \left(\lambda_n + \lambda_{\min}^+(\mathbb{E}\{x_1^\dagger x_1\}) - \Big\|\frac{1}{n}X_{\mathcal{T}}^\dagger X_{\mathcal{T}} - \mathbb{E}[x_1^\dagger x_1]\Big\|^+\right)^{-1}.$$

Both the eigenvalue and the norm above only need to be computed over the subspace spanned by the vectors x_t that occur with non-zero probability. We use the symbol $^+$ to denote the restriction to this subspace. Now notice that $\|.\|^+ \leq \|.\|$ and, since we defined $\Sigma_{\min} \equiv \min_{i:\lambda_i>0}\lambda_i(\mathbb{E}[X_1X_1^\dagger])$, we have that $\lambda_{\min}^+(\mathbb{E}[X_1X_1^\dagger]) \geq \Sigma_{\min}$. Using the following definition, $\Delta\Sigma_n \equiv n^{-1}X_{\mathcal{T}}^\dagger X_{\mathcal{T}}^\dagger - \mathbb{E}[X_1X_1^\dagger]$, this leads to, $c_{a,t} \leq (\lambda_n + \Sigma_{\min} - \|\Delta\Sigma_n\|)^{-1} \leq (\Sigma_{\min} - \|\Delta\Sigma_n\|)^{-1}$.

We now need the following Lemma.

Lemma 2. *Let $\{X_i\}_{i=1}^n$ be a sequence of i.i.d. random vectors of 2-norm bounded by 1. Define $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n X_iX_i^\dagger$ and $\Sigma = \mathbb{E}[X_1X_1^\dagger]$. If $\epsilon \in (0,1)$ then,*

$$\mathbb{P}(\|\hat{\Sigma} - \Sigma\| > \epsilon\|\Sigma\|) \leq 2e^{-C\epsilon^2 n},$$

where $C < 1$ is an absolute constant.

For a proof see [81] (Corollary 50).

We want to apply this lemma to produce a useful bound on the r.h.s. of (2.12). First notice that, conditioning on n, the expression inside the expectation in (2.12) depends through $c_{a,t}$ on n i.i.d. contexts that are distributed according to the original

distribution. Because of this, we can write,

$$\mathbb{P}\Big\{\Big|\frac{1}{n}\sum_{\tau\in\mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\Big| \geq \frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n Q\Big\} \leq \mathbb{E}\Big[2e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}}-\lambda_n Q\right)^{+2}}\Big]$$
$$\leq \sum_{n=1}^{t}\Big(\mathbb{P}\{|\mathcal{T}_{a,t-1}|=n\}\times\mathbb{E}\Big[2e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}}-\lambda_n Q\right)^{+2}}\Big||\mathcal{T}_{a,t-1}|=n\Big]\Big).$$

Using the following algebraic relation: if $z, w > 0$ then $(z-w)^{+2} \geq z^2 - 2zw$, we can now write,

$$\mathbb{E}\Big[e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}}-\lambda_n Q\right)^{+2}}\Big||\mathcal{T}_{a,t-1}|=n\Big]$$
$$\leq \mathbb{P}\{|\Delta\Sigma_n| > \Sigma_{\min}/2|\ |\mathcal{T}_{a,t-1}|=n\} + e^{-\frac{n}{2L^2}\left(\frac{\Sigma_{\min}\Delta_{\min}}{4}-\lambda_n Q\right)^{+2}}$$
$$\leq \mathbb{P}\{|\Delta\Sigma_n| > \Sigma_{\min}/2|\ |\mathcal{T}_{a,t-1}|=n\} + e^{\frac{Q\Delta_{\min}\Sigma_{\min}}{4L^2}}e^{-\frac{n(\Delta_{\min})^2(\Sigma_{\min})^2}{32L^2}}$$

Using Lemma 2 we can continue the chain of inequalities,

$$\mathbb{E}\Big[e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}}-\lambda_n Q\right)^{+2}}\Big||\mathcal{T}_{a,t-1}|=n\Big] \leq 2e^{-C(\Sigma_{\min})^2 n/4} + e^{\frac{Q\Delta_{\min}\Sigma_{\min}}{4L^2}}e^{-\frac{n(\Delta_{\min})^2(\Sigma_{\min})^2}{32L^2}}.$$

Note that $||\Sigma|| \leq 1$ follows from our non-restrictive assumption that $\|x_t\|_2 \leq 1$ for all x_t. Before we proceed we need the following lemma:

Lemma 3. *If* $n_c = \frac{p}{2k}\log t$ *, then* $\mathbb{P}\{|\mathcal{T}_{a,t-1}| < n_c\} \leq t^{-\frac{p}{16K}}$.

Proof. First notice that $|\mathcal{T}_{a,t-1}| = \sum_{i=1}^{t-1} z_i$ where $\{z_i\}_{i=1}^{t-1}$ are independent Bernoulli random variables with parameter $p/(Ki)$. Remember that we can assume that $i > p$ since in the beginning of Algorithm 2 we play each arm p/K times.

Note that $\mathbb{P}(X > c) \leq \mathbb{P}(X + q > c)$ is always true for any r.v. X, c and $q > 0$. Now

write,

$$\begin{aligned}\mathbb{P}(|\mathcal{T}_{a,t-1}| < n_c) &= \mathbb{P}\left(\sum_{i=1}^{t-1} z_i < n_c\right) = \mathbb{P}\left(\sum_{i=1}^{t-1}(z_i - p/(Ki)) < n_c - (p/K)\sum_{i=1}^{t-1} 1/i\right) \\ &\leq \mathbb{P}\left(\sum_{i=1}^{t-1}(-z_i + p/i) > -n_c + (p/K)\sum_{i=1}^{t-1} 1/i\right) \\ &\leq \mathbb{P}\left(\sum_{i=1}^{t-1}(-z_i + p/i) > (p/K)\log t - n_c\right). \end{aligned} \tag{2.13}$$

Since $\sum_{i=1}^{t-1}\mathbb{E}[(z_i - p/(Ki))^2] = \sum_{i=p+1}^{t-1}(1-p/(Ki))(p/(Ki)) \leq \frac{p}{K}\log t$, we have that $\{-z_i + p/i\}_{i=1}^{t-1}$ are i.i.d. random variables with zero mean and sum of variances upper bounded by $(p/K)\log t$. Replacing $n_c = (p/2K)\log t$ in (2.13) and applying Bernstein inequality we get, $\mathbb{P}(|\mathcal{T}_{a,t-1}| < n_c) \leq e^{-\frac{\frac{1}{2}(p/(2K))^2\log^2 t}{\frac{p}{K}\log t + \frac{1}{3}(p/(2K))\log t}} \leq t^{-\frac{p}{16K}}$. □

We can now write, by splitting the sum in $n < n_c$ and $n \geq n_c$

$$\begin{aligned}&\mathbb{P}\Big\{\Big|\frac{1}{n}\sum_{\tau\in\mathcal{T}} x_t^\dagger x_\tau \epsilon_{a,\tau}\Big| \geq \frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n Q\Big\} \\ &\leq \sum_{n=1}^{t}\mathbb{P}\{|\mathcal{T}_{a,t-1}| = n\}\mathbb{E}\Big\{2e^{-\frac{n}{2L^2}\left(\frac{\Delta_{\min}}{2c_{a,t}} - \lambda_n Q\right)^{+2}} \Big| |\mathcal{T}_{a,t-1}| = n\Big\} \\ &\leq \mathbb{P}\{|\mathcal{T}_{a,t-1}| < n_c\} + 4e^{-C(\Sigma_{\min})^2 n_c/4} + 2e^{\frac{Q\Delta_{\min}\Sigma_{\min}}{4L^2}} e^{-\frac{n_c(\Delta_{\min})^2(\Sigma_{\min})^2}{32L^2}} \\ &\leq t^{-\frac{p}{16K}} + 4t^{-\frac{Cp(\Sigma_{\min})^2}{8K}} + 2e^{\frac{Q\Delta_{\min}\Sigma_{\min}}{4L^2}} t^{-\frac{p(\Delta_{\min})^2(\Sigma_{\min})^2}{64KL^2}}.\end{aligned}$$

We want this quantity to be summable over t. Hence we require that,

$$p \geq \frac{128KL^2}{(\Delta_{\min})^2(\Sigma_{\min})^2}, p \geq \frac{16K}{C(\Sigma_{\min})^2}, p \geq 32K. \tag{2.14}$$

It is immediate to see that our proof also follows if $\Delta_{\min}$, $\Sigma_{\min}$ and L are replaced by $\Delta'_{\min} = \min\{1, \Delta_{\min}\}$, $\Sigma'_{\min} = \min\{1, \Sigma_{\min}\}$ and $L' = \max\{1, L\}$ respectively. If this is done, it is easy to see that conditions (2.14) are all satisfied by the p stated in Theorem

3. Since $\sum_{t=1}^{\infty} 1/t^2 \leq 2$, gathering all terms together we have,

$$R(T) \leq p\Delta_{\max}\sqrt{d} + p\Delta_{\max}\sqrt{d}\log T + \Delta_{\max}\sqrt{d}K\left(4e^{\frac{Q\Delta'_{\min}\Sigma'_{\min}}{4L'^2}} + 10\right)$$

$$\leq p\Delta_{\max}\sqrt{d} + 14\Delta_{\max}\sqrt{d}Ke^{Q/4} + p\Delta_{\max}\sqrt{d}\log T. \qquad \square$$

2.5 Numerical Results

In Theorem 1 and Theorem 3, we showed that, Algorithm 1 and Algorithm 2 have an expected regret of $O(\log T)$. We now illustrate this point by numerical simulations.

2.5.1 Group-UCB

We evaluated Group-UCB on two datasets, *MovieLens* [32] and *MoviePilot* [58]. ***MovieLens Dataset.*** The*MovieLens* dataset consists of 1 000 209 ratings, given by 6 040 users to 3 883 movies. Ratings of a movie range from 1 to 5. Movies in the dataset are labeled by genres, such as "Animation", "Children's", *etc.* When we build the user-movie matrix, where each row is a user, each column represents a movie, and the cell is the rating from the user to the movie, there are around 96% cells for which ratings are missing. We construct a low-rank approximation of the dataset, and subsequently use it to predict those missing ratings—*i.e.,* perform matrix completion.

MoviePilot Dataset. The dataset comprises $4,536,891$ ratings given by $171,670$ users over $23,974$ movies. In addition, the dataset includes household information about a subset of the users. In particular, this subset is organized in 290 households with sizes 2, 3 and 4. Specifically, 272 households are formed by 2 users and 14 by 3 users, while only 4 households have 4 users. As the *MovieLens* dataset, we again preprocess the dataset to complete the user-movie rating matrix.

Simulation Setup and Evaluation. In the simulation, arms correspond to movie genres, and rewards to ratings. The process is as follows. Given a group G, at each session t, the subgroup S_t is selected uniformly at random from non-empty subset of G. If the Group-UCB policy selects the genre a, a movie selected u.a.r. among movies in the genre a is displayed to the present group. The reaction of a user is then the rating

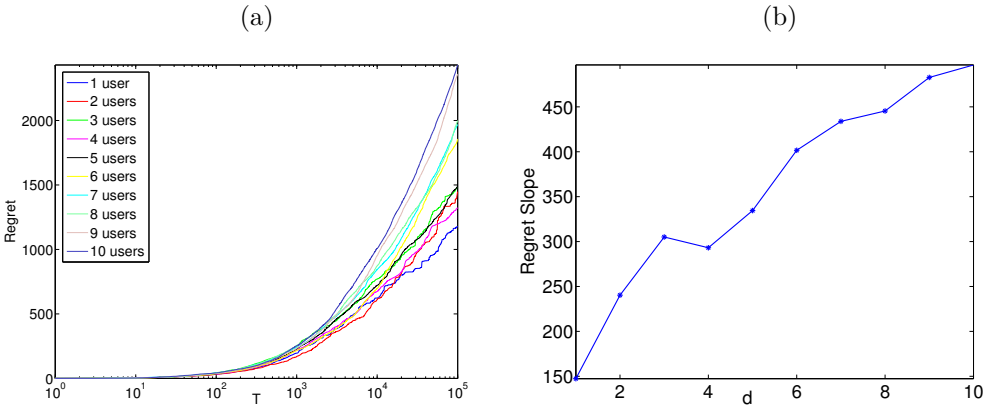


Figure 2.1. (a) Random groups up to 10 users. (b) Average slope of the regret with regard to group size.

that she provided. If the rating is missing, the predicted rating is used.

We consider two types of groups: a) a group of randomly picked users; b) a group of users who share location. For a), we randomly pick 10 users from the dataset, then build 10 groups by adding one more user to a group each time. For b), we select 6 users with the same zip code, similarly build 6 groups by adding one more person to form a new group each time. Profiles of members in this group are shown in Table 2.1. We evaluate the performance by cumulative regret over time. Figure 2.1(a) shows the regret for group a) and Figure 2.2(a) for group b). Both plots are in semi-log scale, and show that regrets are indeed grow logarithmically. We compute the regret slope as as the slope of the final portion of the curve. We take the average regret slope from multiple samples of groups in varied sizes. Figure 2.1(b) shows the correlation between the slope and the group size, and it is positive.

User ID	Gender	Age	Occupation
1	Male	18-24	college/grad student
2	Male	18-24	programmer
3	Male	18-24	college/grad student
4	Female	25-34	sales/marketing
5	Female	18-24	other
6	Male	35-44	academic/educator

Table 2.1. Member information in the group

In above simulation, members in the group have equal weight on the reward. We now investigate the scenario in which users have different influence in the group reward.

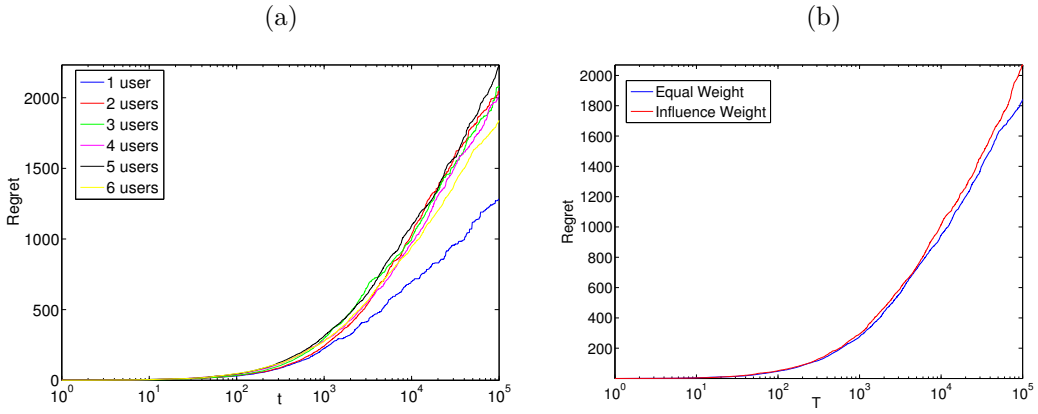


Figure 2.2. (a)Regrets for groups of people with same zip code. (b)Regrets for equal weight vector and influence weight vector.

As it is not easy to determine the influence power of each member, we use an arbitrary heuristic for experiment: 1) female gain 1 more weight unit; 2) children and elder people add 1 more weight unit. Taking the 6-user group in Table 2.1 as an example, the weight vector w is (1,1,1,3,2,3) before normalization. We then apply Group-UCB on the group with this weight vector, with other settings not changed. Figure 2.2(b) displays the regrets for *Equal Weight* and our heuristic *Influence Weight*, on semi-log scale. We can see that regrets are logarithmic in time t for both groups, indicating that our Group-UCB policy can take varying weight vector x_t.

In contrast to *MovieLens*, *MoviePilot* contains no genre information of movies. For this reason, we generated genres artificially by taking into account how well known different movies are. Note that a movie being well-known does not necessarily mean it has been rated highly. Our notion of genre basically tries to distinguish mainstream movies from movies only known to a more selective audience. The different genres were defined as follows. First we computed an histogram of the frequency of viewings for movies. Then we partitioned this histogram into different sections. E.g. its tail, its kink, etc... The different sections corresponding a movie being watched by a number of users in one of the following intervals: $[500, 1\,000)$, $[1\,000, 2\,793)$, $[2\,793, 5\,793)$, $[5\,793, 11\,793)$ and $[11\,793, \infty)$. Finally, we associated different genres to each of these different intervals. Our interest in using the *MoviePilot* data set is to identify whether interesting phenomena occur as the results of using real group information. This could not be done

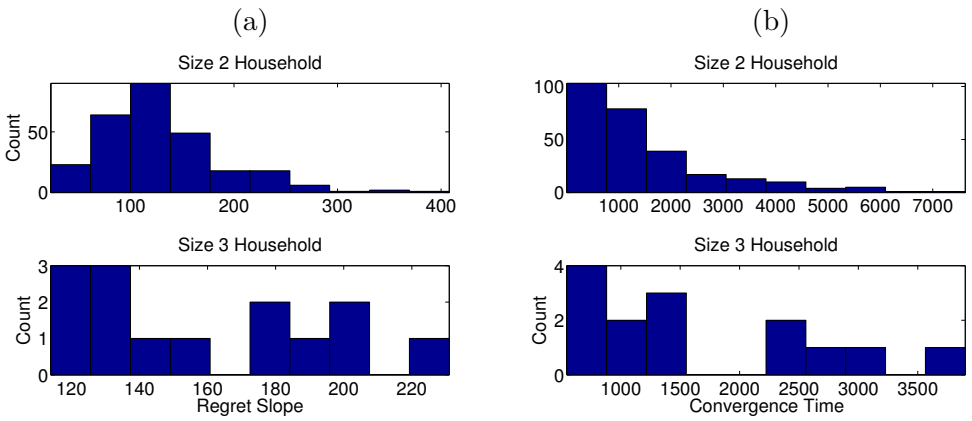


Figure 2.3. (a) Distributions of scaling coefficients for households of size 2 and 3. (b) Distributions of convergence times for households of size 2 and 3.

with the *MovieLens* data set because no group information is given. Hence the reason why we had to artificially form groups by grouping users with similar characteristics, like zip code.

We computed the distribution of the regret slope for different household sizes and also the distribution of the convergence times. The results are shown in Figure 2.3. The distributions for size 3 households are not very informative because there are not that many households of size 3 in the data set. However, looking at houses of size 2 one finds that both the values of the scaling coefficient and convergence time oscillate quite a bit. In particular, the distribution of the convergence times seems to have a long tail. This points to the fact that the behavior of Group-UCB is significantly dependent on the actual expected rewards of each individual of the group for each genre not only on the size of the group. From Figure 2.4 one also sees, not surprisingly, that poorer performance (higher regret slope) seems to be positively correlated with slower convergence times.

2.5.2 Contextual ϵ-Greedy

We here show the experimental results for Algorithm2, most importantly, exemplify how violating the stochastic assumption might degrade its performance. Figure 2.5 (a) shows the average cumulative regret (in semi-log scale) over 10 independent runs of Algorithm 2 for $T = 10^5$ and for the following setup. The context variables $x \in \mathbb{R}^3$ and at each

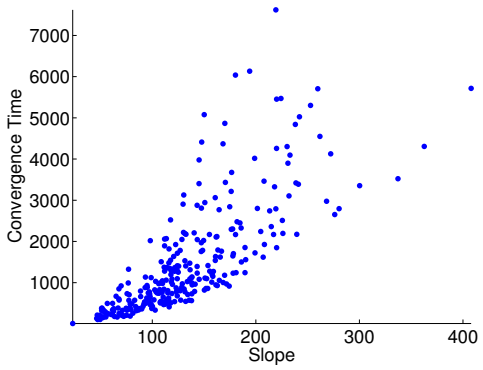


Figure 2.4. Scatter plot showing correlation between scaling coefficients and convergence times.

time step $\{x_t\}_{t\geq 1}$ are drawn i.i.d. in the following way: (a) set each entry of x to 1 or 0 independently with probability 1/2; (b) normalize x. We consider $K = 6$ arms with corresponding parameters θ_a generated independently from a standard multivariate gaussian distribution. Given a context x and an arm a, rewards were random and independently generated from a uniform distribution $U([0, 2x^\dagger\theta_a])$. As expected, the regret is logarithmic. Figure 2.5 (a) shows a straight line at the end.

To understand the effect of the stochasticity of x on the regret, we consider the following scenario: with every other parameter unchanged, let $\mathcal{X} = \{x, x'\}$. At every time step $x = [1, 1, 1]$ appears with probability $1/z$, and $x' = [1, 0, 1]$ appears with probability $1 - (1/z)$. Figure 2.5 (b) shows the dependency of the expected regret on the context distribution for $z = 5$, 10 and 100. One can see that an increase of z causes a proportional increase in the regret.

2.6 A Group Recommender Demo System

In this section, we present our group recommender demo system, using dining out with friends as the application. For classic recommender system for individuals, the post-recommendation behavior is simple: users give ratings or zero-one feedback to recommended items. It gets complicated for groups. Members may have different feedback towards the recommendation. Existing group recommender systems just present results to each member in the group [21, 60], and assume members can discuss the

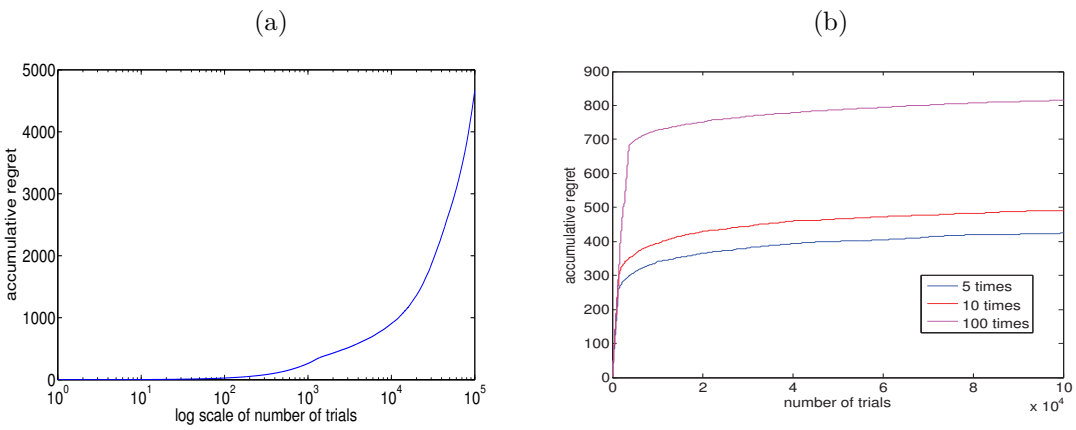


Figure 2.5. (a) Regret over T when x_t is from i.i.d. (b) Regret over T when x_t is not from i.i.d.

results offline [54].

In the design of our demo system, we focus on facilitating group consensus to the recommendation. It allows users organizing group events, provides recommendation and helps the group to reach consensus. The group decision making process is shown in Figure 2.6. Members can vote and chat for each options, or select new options. Members' votes are stored to learn their preferences, which are used in the recommendation for future events.

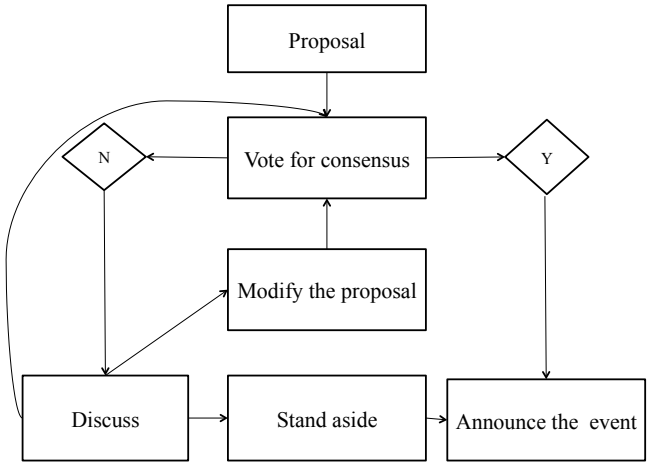


Figure 2.6. Group decision making framework

2.6.1 System Overview

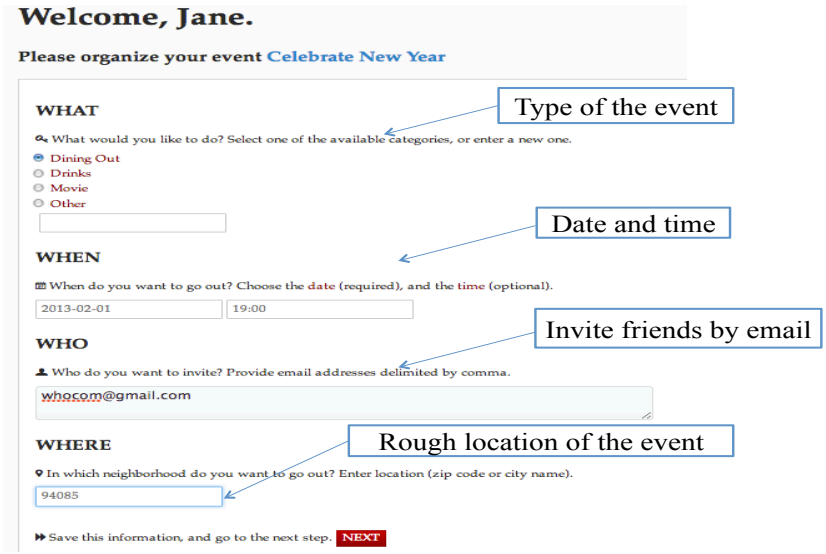
The workflow of the system is as follows. A new user needs to sign up with a valid email address. The system will send a validation email and let the user complete the registration. The user then can login the system to see his/her profile page, where the user can create a new event, view current events, and review past events.

The user who creates the event serves as the admin of the event. As in Figure 2.8(a), the admin will provide the type of event, date and time, friends' email addresses and the location of the event. The system will automatically suggest 5 friends, based on the co-event frequency with the admin. The group is then constituted by the admin and friends he/she invited. Each member can also invite more people to the event in later stage.

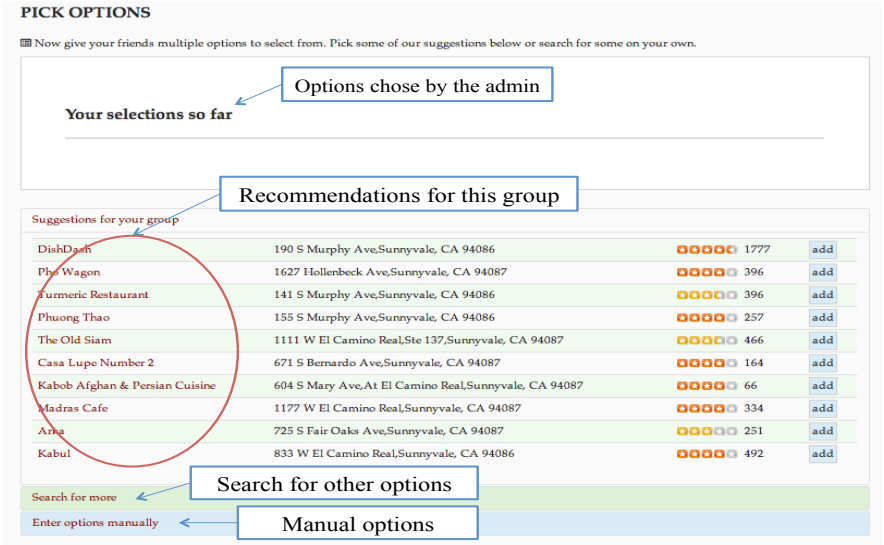
Based on the context provided by the admin, the system generates a recommendation list with 10 items tailored to this group as in Figure 2.8(b). The admin can select one or more options from the recommended list, or through the search interface, or manually type places. When choices are picked, the admin's task of organizing event is now finished. The system will send out an invitation email to members in the group, who are attendees. The email contains the link for the event page as in Figure 2.8(c).

Attendees can vote options either positively or negatively on the event page as in Figure 2.8(d). Each member's feedback is revealed to the whole group. Attendees can change their opinions based on others' votes. Attendees can leave comments on the event page, which are visible to the whole group. When the group reaches consensus on the option, the admin can close the event. The system will send out emails with the event summary including date, time, members, and the final place picked by the group.

There are two places the system pushes the recommended results to the group. One is when the admin organizes the event as in Figure 2.8(b). The other scenario is when attendees see the event page and decide to add more options.



(a) Event Detail

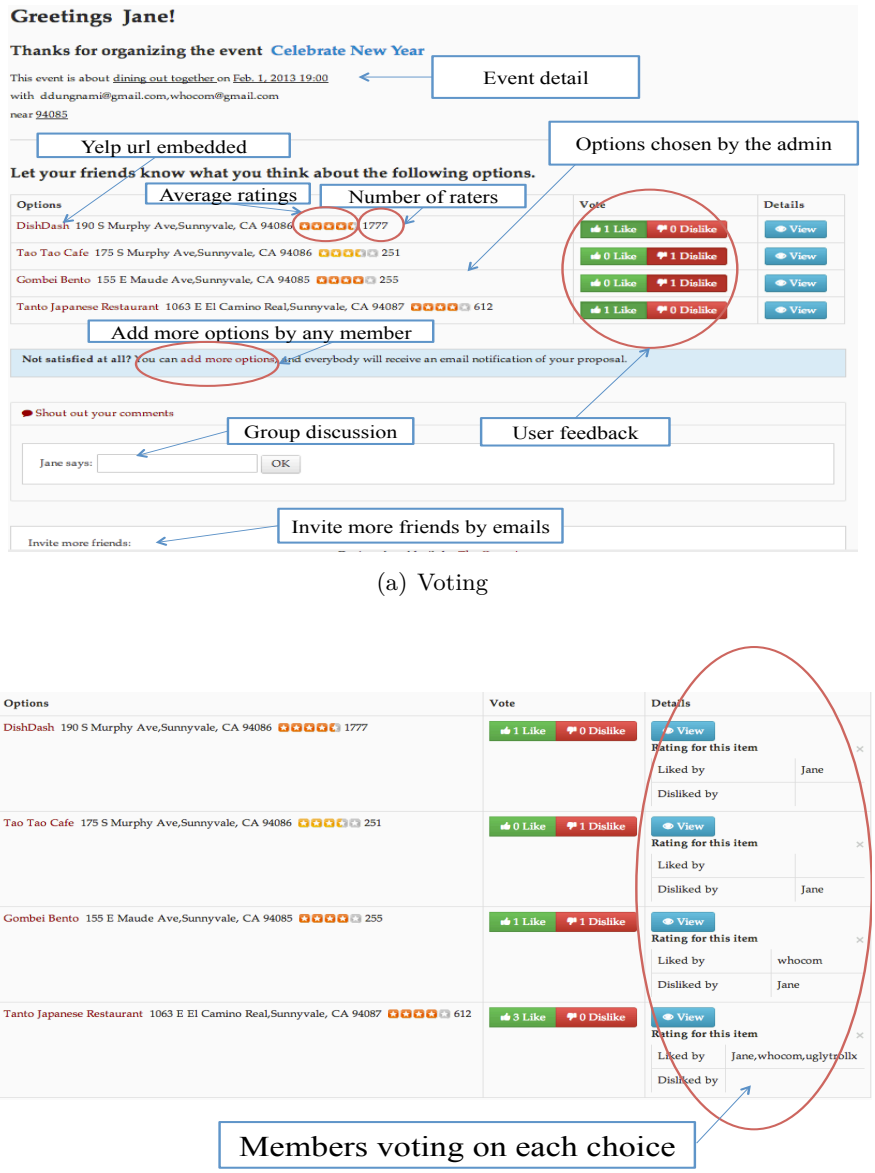


(b) Recommendation

Figure 2.7. Create event and recommendation for the group

2.6.2 Recommendation Model

In the system, each user is identified through their valid email address. The system has a database which keeps track of user, item (which are restaurants in our application), event, user-item vote in each event and derived friendship based on the co-presence of attendees in events. Figure 2.9 shows the information the system stored.



(a) Voting

(b) View Detail

Figure 2.8. Voting and other members' decisions

Learn Member's Preference

Formally, we denote by all users the set $\mathcal{U}$ where $u \in \mathcal{U}$, items the set $\mathcal{I}$ where $i \in \mathcal{I}$, events the set $\mathcal{E}$ where $e \in \mathcal{E}$. The vote for each item in an event by a user is binary in $y_{u,i} = \{0, 1\}$. Each item has features extracted from yelp page and is represented in the feature vector x_i. Each member's preference on these features is denoted by θ_u. We assume the vote of a user to an item is a function of user preference and item features:

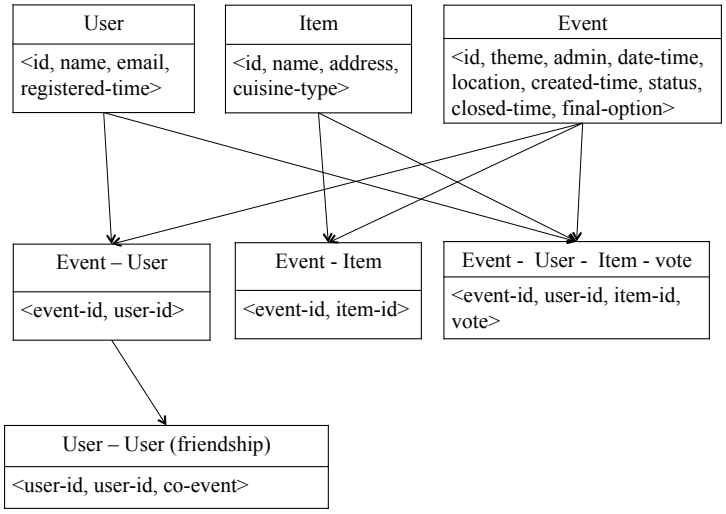


Figure 2.9. Information stored by the system

$y_{u,i} = \Phi(x_i^\dagger \theta_u)$. For users participated in enough historical events in the system, we can apply broadly used techniques such as logistic regression to learn the parameter θ_u. For new users or who lack of sufficient information, we can apply MAB algorithm discussed in previous sections to quickly estimate θ_u.

Learn Members' Influence

The member's preference is assumed to be independent to other members in the group. However, in the group consensus making process, each member is aware of others' votes which can impact their own decisions. We propose an influence cascade model which assumes the decision made by each member is affected by the individual's preference and other members' influence.

Consider the scenario that a group G with $|G| = d$ finite number of members tries to reach consensus on the item i. Every individual provides his/her own feedback to the item, which is represented by a binary value $y_{u,i} \in \{0, 1\}$. Note that $y_{u,i}$ can be multinomial variable such as "like", "dislike" or "neutral", and it can also be a real number scaling from 1 to 5. We here deal with the binary case only. Let $e = (G, i, \{y_{u,i}\}_{u \in G})$ be an event by the group G with the item i. The vote of each item for the group depends on the number of positive votes. In other words, the average

strategy is used to determine the group preference of an item as $y_{G,i} = 1/|G| \sum_{u \in G} y_{u,i}$.

We use $p(u|i)$ to denote the individual inherent preference on the item, $p(v|u)$ be the influence of the decision by the member v to the member u. We now model how each individual votes given he/she is aware of existing decisions of others. The order in which people vote can be arbitrary and depends on when they access the decision process. Moreover, their vote can be influenced by the votes of other people voted before them. Let $\mathcal{U}_e^+(u)$ be users who give positive feedback in event e before user u and $\mathcal{U}_e^-(u)$ be users who give negative feedback. The probability that the member u will give positive feedback to the item depends on his own preference on the item and other members' feedback as in (2.15). Putting differently, a user u votes positively by tossing $|\mathcal{U}_e^+(u)| + 1$ independent $0-1$ coins and observing whether any of them return 1. The first coin is 1 with probability $p(u|i)$, and others with probability $p(v|u)$.

$$\Pr(y_{u,i} = 1) = 1 - (1 - p(u|i)) * \prod_{v \in \mathcal{U}_e^+(u)} (1 - p(v|u)) \tag{2.15}$$

Given a set of events E, we can estimate the pairwise influence $p(v|u)$ and independent preference $p(u|i)$ by maximizing the likelihood of all events which the user u has participated.

$$L(E, u) = \prod_{e: y_{u,i}=1} \Pr(y_{u,i} = 1) \prod_{e: y_{u,i}=0} \Pr(y_{u,i} = 0) \tag{2.16}$$

We change the variables and let $1 - p(u|i) = q_{u,i}$, $r_{u,i} = \Pr(y_{u,i} = 1)$, $b_{v,u} = 1 - p(v|u)$. The problem becomes

$$\begin{aligned}
\text{Maximize} \quad & \prod_{e: y_{u,i}=1} r_{u,i} \prod_{e: y_{u,i}=0} \prod_{v \in U_e^-(u)} b_{v,u} q_{u,i} \\
\text{subject to} \quad & 0 \leq r_{u,i} \leq 1; 0 \leq q_{u,i} \leq 1 \, \forall i \\
& 0 \leq b_{v,u} \leq 1 \, \forall v \\
& r_{u,i} + \prod_{v \in U_e^+(u)} b_{v,i} q_{u,i} \leq 1 \, \forall e
\end{aligned}$$

Note the last constraint is an inequality rather than an equality. The objective function will strictly increase when either increase $r_{u,i}$ or $b_{v,i}$ or $q_{u,i}$, so the inequality will always be a binding constraint at the solution. The objective function is a monomial thus our problem is a geometric program which can be solved efficiently(see [59]).

The individual preference $p(u|i)$ reflects the interests of the individual and does not vary to a given event or a given group. We can estimate the individual preference from external resources, or by selecting events that the individual is the first one to make decision and using logistic regression on these events. Then $q_{u,i} = 1 - p(u|i)$ is treated as known constants when solving (2.16).

Once we learn $p(u|i)$ and $p(v|u)$ through the training data, we can predict the individual's vote on future event as $y(u,i) = \beta * p(u|i) + (1-\beta)\frac{\sum_{v \in G} p(v|u) * p(v|i)}{|G|}$, where β controls the impact of individual's inherent preference and members' influence.

2.6.3 User Study and Model Evaluation

We use an offline user study to measure the performance of our influence cascade model. The main intuition is that if there is no pairwise influence on the decision, we can get good prediction on the individual feedback only using individual preference. Otherwise, our decision cascading model in (2.15) will achieve better prediction performance. We sampled 277 group events organized through our system. In total 19 individuals are involved and 79 items are chosen. Groups have size 2, 4 and 8. Every user participated in 54 events on average. A few of members did not provide any feedback. We collected 4398 valid votes at last. We split events chronologically into training(80%)and testing set (20%). The baseline algorithm uses logistic regression to predict individual preference, which is in turn used to predict individual's vote.

	baseline	influence model
true positive rate	0.52	0.77
false positive rate	0.35	0.6
accuracy	0.65	0.7
AUC	0.61	0.85

Table 2.2. Prediction performance on test set, prediction threshold = 0.5

Table 2.2 shows that the influence cascade model has high true positive rate, better accuracy and much better AUC on the test set. However, the false positive rate is also higher. This is because our model focuses on positive votes only, and ignores the impact of negative and missing votes from members. A future work is to model the influence behind both positive and negative votes.

2.7 Conclusion

Our work has initiated a group MAB problem with a group of users, which is applicable for many applications, particularly group recommendation. Our group MAB setting addressed the challenges that prior information about the individual and the group is missing, and the group is dynamic in terms of participated members at each time. We considered two scenarios regarding the group feedback process: 1) receiving every present member's feedback; 2) receiving only single group feedback. We designed corresponding policies, and presented the theoretic proof for their regret bounds. We also showed the effectiveness of our policies through numeric results.

We also designed a group recommender demo system which is consensus-focused. It used group dining out as the application and facilitates group decision making to recommendations. We conducted user study through system and collected data of 277 group events. We proposed a influence cascade model which takes in account members' influence when each member makes vote. Experiments showed that the influence cascade model can effectively predict the individual's votes to items.

Chapter 3

Cohort Modeling for Enhanced Personalized Search

Web search engines utilize behavioral signals to develop search experiences tailored to individual users. To be effective, such personalization depends on access to sufficient information about each user's interests and intentions. For new users or queries, profile information may be sparse or non-existent. To handle these cases, and perhaps also improve personalization for the others, search engines can employ signals from those similar to the current user along one or more dimensions, i.e., those in the same cohort. In this chapter we describe a characterization and evaluation of the use of such cohort modeling to enhance search personalization. We experiment with three predefined cohorts-topic, location, and top-level domain preference independently and in combination, and also evaluate methods to learn cohorts dynamically. We show via extensive experimentation with large-scale logs from a commercial search engine that by leveraging cohort behavior we can attain significant relevance improvements when combined with a production ranker that uses similar classes of personalization signal but at the individual searcher level. Additional experiments show that our gains can be extended when we dynamically learn cohorts and target classes of ambiguous or unseen queries.

3.1 Introduction

3.1.1 Background and Related Work

There are three relevant areas of related work: (1) personalization of search engines based on short- and long-term searcher interests, (2) collaborative filtering, and (3)

mining the search behavior of other users to complement and enhance search personalization.

Large-scale behavioral data from search engines has been mined extensively to improve result relevance in the aggregate across all users [3, 38]. Search preferences are personal and research on personalizing retrieval [66, 78] has shown that implicitly collected information such as browser history, query history, and desk-top information, can be used to improve the relevance of search results for a particular individual. Short-term behavior from within the current search session has been used for tasks such as result ranking [39] or predicting future search interests [86, 87]. Teevan *et al.* [78] showed that their personalization algorithm improved as more data was available about the current user. Long-term behavior has been used for personalizing search by building longitudinal models of user interests [70], including using the previous queries associated with the pursuit of similar information needs [74]. Models can use different sources, ranging from specific query-URL pairs which have high precision but low coverage [79] to more general methods that use topical representations of user search interests [70].

When there is insufficient data about the current user, the search behavior of other related users may be beneficial in modeling user interests and intentions. Teevan *et al.* [80] explored the similarity of query selection, desktop information, and explicit relevance judgments across a small group of work colleagues grouped along two dimensions: (1) the longevity of their personal relationship, and (2) how explicitly the group was formed. They found that some groupings provide insight into what members considered relevant to queries related to the group focus, but that it can be challenging to identify valuable groups implicitly. White *et al.* [88] address this issue by implicitly modeling the search task of the user, finding others who have attempted a similar task, and using their on-task behavior to enhance relevance. Although they used cohorts (location, topic expertise, and search engine entry point) as part of their ranking experiments, they observed limited gain in their experimental setting and how they chose to model and integrate cohorts.

Collaborative filtering (CF) [30] can also be used to find people with similar interests and leverage their activities and preferences to help the current user. The lack of

sufficient personal information (sometimes referred to as the "cold start" problem) has been studied in research on CF and on recommender systems [64]. This research has shown that a number of sources can be used to generate recommendations from others in a given community, including agreement in item ratings [61] and social network memberships [42].

There are three predefined cohorts that we focus on in our study: topical interests, domain preferences, and geographic location. We now describe relevant related work in each area, beginning with topical interest, some of which leverages CF to find similar users.

Topic information can be used directly to improve search engine ranking [14]. Sugiyama *et al.* [71] addressed sparseness in user term weight profiles by applying CF techniques to attain term weights based on those of users with similar profiles. Similar approaches have used click-through data to personalize result rankings and backed-off to the clicks of others [4, 72]. Almeida and Almeida [4] used Bayesian algorithms to cluster users of an online bookstore into communities based on links clicked within the site and found that the popularity of links within different communities could be used to customize result rankings. Lee [46] proposed a system that uses data mining to uncover patterns in users' queries and browsing to generate recommendations for users with similar queries. These techniques perform matching with other users based on individual queries or URLs, severely limiting coverage. Freyne and Smyth [19] addressed this concern by connecting different communities based on the degree to which their queries and result clicks overlap.

Alternative methods have been proposed that are query independent. Smyth [69] suggested that click-through data from users in the same "search community" (e.g., a group of people who use a special interest Web portal or work together) could enhance search. He provided evidence for the existence of search communities by showing that a group of co-workers had a higher query similarity threshold than general Web users. Leong *et al.* [36] showed that searchers exhibited domain preferences, where they favored particular sources when selecting results. White et al. [89] found that domain experts preferred different top-level domains than novices, with more focus on

educational (.edu) and governmental (.gov) sites, whereas novices preferred commercial (.com) sites.

Turning to location, Mei and Church [56] found that geographic location might serve as a reasonable proxy for community, since they observed that grouping users based on the IP address similarity could improve relevance. Cheng and Cantu-Paz [19] developed models for personalized click prediction in online advertising that leveraged demographic and location features to improve prediction accuracy. Bennett *et al.* [13] showed the effectiveness of location-based personalization, whereby models of searcher interests for particular locations can be learned and used in concert with the searcher's current location to improve relevance. White and Buscher [85] automatically identified users with local expertise (knowledge of a specific city or town) from search log data, and showed that the interests of these local users was both different and that there were differences in the quality of the entities they visited (restaurants reserved in this case), with locals selecting higher rated venues. Weber and Castillo [84] estimated searcher demographics by joining search location with census data and demonstrated variations in search behavior for different demographic groups.

3.1.2 Contributions

Our research extends previous work and makes the following research contributions:

- We generate pre-defined cohorts using information readily available to search engines, specifically topic, location, and top-level domain preference.
- We show that modeling user interests within these cohorts can enhance state-of-the-art search personalization methods, leading to significant gains in search relevance.
- We show that there are particular sets of easily-identifiable queries (such as ambiguous or new queries for a given user), for which our cohort modeling approach can be particularly effective.
- We propose methods to dynamically learn cohorts rather than use pre-defined sets, and demonstrates strong relevance gains.

3.2 Cohort Modeling

We now describe the construction of our cohort model, beginning with the nature of the data, but also including features computed.

Upon submission of a query to a search engine, a list of search results is retrieved and ranked for the user. The user examines the list and decides the next action on the results: click or not click. Search engine logs capture much of this interaction. An entry in the search log comprises a tuple $\langle u, q, d, c, t \rangle$, where u is a user from the universe of users U, q is a query in the universe set of queries Q, and d is from the universe of documents (search results) D, c is a binary value that 1 is a click and 0 otherwise, and t is the timestamp. Such tuples can be created for each of the top-ranked search results returned by the search engine.

The click-through rate (CTR) of a query-document pair $\langle q, d \rangle$ is the ratio of the number of clicks on the document to the number of impressions in which that result is shown for the search query. CTR is commonly used to measure the probability of a click given a query-document pair, i.e., $P(c = 1|d, q)$. Given this, plus the simplicity and general applicability of CTR, it seems appropriate to focus on it for this first study of cohort modeling. In our approach we focus on a subset of clicks suggesting that searchers are satisfied with the particular search results that they have selected. We refer to these as satisfied (SAT) clicks.

Definition 1 (SAT Clicks)**.** *As defined in [30], SAT clicks have an associated dwell time of $\geq$ 30 seconds between search engine actions, or it is the last action in a search session (assumed to be SAT).*

Using SAT clicks rather than all clicks can provide a more accurate CTR signal since accidental or misinformed clicks are excluded. Therefore, rather than simply counting the number of clicks divided by the number of impressions, we can compute CTR as:

$$ctr(d, q) = \frac{SATClicks(d, q)}{Impressions(d, q)} \tag{3.1}$$

Users may search for different information under the same query. This can also be

reflected in a CTR tailored to each searcher. We use an individual's click-through rate $CTR(d, q, u)$ to estimate the degree of satisfaction of the user with a document given a query:

$$ctr(d, q, u) = \frac{SATClicks(d, q, u)}{Impressions(d, q, u)} \tag{3.2}$$

As mentioned earlier, we represent users by contextual features corresponding to domain preference, location, and topic interests. We now describe each of the representations in turn.

Top-Level Domain: The domain name of a URL represents its networking context, the administrative autonomy, and authority. A recent study showed that searchers exhibit a preference for particular domains irrespective of relevance [36]. The number of unique domain names across the broad range of information needs in our dataset is intractable. We therefore elect to study top-level domain (TLD) first. TLD includes generic domain extensions such as .com, .net, .org; sponsored extensions such as .mil, .asia, .edu; and country code such as .us, .uk, .fr. Related work on domain expertise in search revealed that there were differences in the TLDs selected depending on user domain expertise level (experts preferred .edu and .gov, novices preferred .com) [89]. The TLD may therefore offer some insight into the subject matter expertise of the searcher, which can be useful in performing richer personalization.

We limit our study on search logs collected in the United States geographic locale, but we observe many clicks on URLs with other country code domain extensions. This information could be used to estimate the native language of users or simply countries with which they have an interest. There are many TLDs, and because many are fairly new or visited infrequently (at least from search results), we do not observe many clicks related to them.

We include all general and sponsored TLDs (23 in total), and also select 11 country code TLDs which are registered in the first and second year of availability, since we observe the number of Web pages and clicks of a TLD is related to its time in existence. We then randomly sample 3% search logs during a two-month period and examine the number of SAT clicks on selected TLDs. TLDs with less than 1000 SAT clicks, e.g.,

.arpa, .post, .tel, are excluded. We retained the remaining 31 popular TLDs and use "othe" for all other cases. This set of TLDs is used to construct our cohorts.

Location: The location of a user may also reveal their search interests and intentions [13]. We estimated the location of the user at query time using reverse IP geocoding. Since a user may not be confined to a particular city, but will generally remain within a state, we compute location preference for each user at the state level. There are 51 U.S. state features. When we failed to identify the location of a user, we categorize their location as "other".

Topic: We utilize the Open Directory Project (ODP, dmoz.org), a human-generated hierarchical taxonomy of Websites, as our topical ontology. This has been used extensively in previous work on personalization to model search interests at a level beyond queries and documents [15, 70]. Topics are assigned to URLs using the content-based classifier described and evaluated in [14]. The user's degree of interest in a topic is then inferred from the number of clicks of URL results under that topic in her past click history. ODP contains 15 top-level categories such as "Arts", "Sports", etc. To control the size of the feature space, we focus on top-level categories only.

Given features explained above, we define a cohort as a group of users sharing a contextual feature. The total number of selected features is 99. In other words, we define 99 cohorts of users based on shared contextual features. A user can be a member of multiple cohorts. We model cohort membership to indicate how likely a user is to be a member. It is also used to measure how strongly to weight the user contribution to the cohort when aggregating co-hort clicks.

We denote C_j as the $j-$th cohort of a particular type, C^T as cohorts of top-level domains, C^L as cohorts of locations, and C^O of ODP categories (topic). Since the following calculations for each of the three cohort types are the same, we ignore the superscript in the cohort notation for simplicity.

Definition 2 (Cohort Membership)**.** *The cohort membership vector for user u is defined as a m-tuple* $W(u) = [w(u,1), w(u,2), \ldots, w(u,m)]$*, in which m is the number of cohorts, and w(u,i) represents the degree of membership for the user in i-th cohort (say, "California"). W(u) is normalized such that* $\sum_i w(u, C_i) = 1$*. The cohort membership*

is drawn from a multinomial distribution of SAT clicks, and calculated as follows:

$$w(u, C_j) = \frac{SATClicks(u, C_j) + 1}{\sum_i SATClicks(u, C_i) + K} \tag{3.3}$$

Example 1: Suppose there are only three cohorts: California, Washington, and Oregon, we observe three SAT clicks when the user is at California, one SAT click at Washington, then the cohort membership is [0.57, 0.29, 0.14].

Definition 3 (Cohort CTR)**.** *Given a cohort type (e.g., Topic), the cohort CTR for a query and URL document* $\langle q, d \rangle$ *is a m-tuple* $c\text{-}ctr(d, q) = [ctr(d, q, C_1), ctr(d, q, C_2), \ldots, ctr(d, q, C_m)]$*, in which* m *is the number of cohorts, and* $ctr(d, q, C_i)$ *is the probability that users in* $i-th$ *cohort will click document* d *for the query* q*. It is a weighted aggregation of individual CTR as follows:*

$$ctr(d, q, C_j) = \frac{\sum_u SatClicks(d, q, u) * w(u, C_j)}{\sum_u Impressions(d, q, u) * w(u, C_j)} \tag{3.4}$$

Cohort CTR is used to measure the cohort preference on the document d given the query q. It weights a user's clicks by their cohort membership. Users who exhibit strong preference to the cohort will contribute more in the cohort CTR, e.g., for a California state cohort, a user residing in that state for a long duration will have a higher influence factor than a user who only visits occasionally.

Example 2: Suppose there are only three cohorts: California, Washington and Oregon, and two users a and b. The cohort membership vector for a is $W(a)$= [0.57, 0.29, 0.14], for b is $W(b)$= [0.1, 0.1, 0.8]. Given the query [osu], considering two search results d_1= "osu.ppy.sh", and d_2 = "oregonstate.edu", the number of SAT clicks by user a is $S(a)$= [5, 1] for d_1 and d_2 respectively, the number of SAT clicks by user b is $S(b)$ = [1, 5]. For simplicity, we assume the impression on each document for each user is 100. By equation (4), we can compute the cohort CTR for the result d_1 as $c\text{-}ctr(d_1, q)$ = [0.044, 0.039, 0.016], and $c\text{-}ctr(d_2, q)$ = [0.0159, 0.02, 0.044]. It demonstrates that California cohort prefers the result d_1, and Oregon cohort prefers d_2, given the query [osu]. Note that the global CTRs for both results are the same: $ctr(d_1, q) = ctr(d_2, q)$

$= (5 + 1)/(100 + 100)$.

There are two intuitions behind our model. First, users in a cohort with shared contextual features are likely to be coherent in search intents and click preference (an assertion supported by some of our preliminary investigations is not reported here for space reasons). Second, a common approach for handling the problem of insufficient individual historical data is to leverage global CTR. However, global CTR treats clicks from all users equally, and therefore has limited potential to help in personalization. Our approach identifies and separates cohort clicks from global clicks. When estimating an individual's click preference, we can learn more from clicks by cohorts of similar users, who have higher impact on the estimation, and are better aligned with the target user. We show in our later experiments that cohort modeling can outperform global CTR.

CTR is one of the most informative metrics to measure search result quality. However, CTR estimates are sometimes noisy when observations are scarce. For example, if we only observe one impression for a pair $\langle d, q \rangle$, and a single SAT click on the document d, we will obtain $ctr(d, q) = 1$. This is an inaccurate estimate of the true click probability caused by data sparseness. These instances are quite common in logs, especially for tail queries that occur rarely. To handle this situation, we apply smoothing methods to estimate CTR. We add a pseudo count that counts SAT clicks αN times during N impressions. The smoothed CTR is computed as follows.

$$\hat{ctr}(d, q) = \frac{SATClicks(d, q) + \alpha N}{Impressions(d, q) + N} \tag{3.5}$$

After smoothing, extreme cases should have lower CTR than URLs with sufficient SAT clicks and impressions, but higher than those with no SAT clicks. Based on this expectation, we sample hundreds of instances and manually validate the output to tune α and N. Following several experiments we set $\alpha = 0.001$, and $N = 1000$. When calculating cohort CTR for a given cohort C_j, we then smooth cohort CTR with smoothed global CTR as follows:

$$\hat{ctr}(d, q, C_j) = \frac{N * \hat{ctr}(d, q) + \sum_u SATClicks(d, q, u) * w(u, C_j)}{N + \sum_u Impressions(d, q) * w(u, C_j)} \tag{3.6}$$

We set $N = 10$ through similar manual validation. For un-observed or scarcely observed $\langle d, q, C_j \rangle$, the cohort CTR is aligned with a smoothed global CTR of $\langle d, q \rangle$.

For a user, the click probability on a URL document can be estimated from the click history of similar people. Given the cohort model, we now derive cohort features, which infer individual click probability under her cohort membership.

Definition 4 (Cohort Features)**.** *Consider a user u with cohort membership $W(u)$, and the cohort CTR for a query document pair c-ctr(d, q), we derive cohort features $Z(d, q, u)$ as an m-tuple: $[z(d, q, u, C_1), z(d, q, u, C_2), \ldots, z(d, q, u, C_m)]$, where m is the number of cohorts, and $z(d, q, u, C_j)$ is the click probability in the j-th cohort. The probability is computed as follows:*

$$z(d, q, u, C_j) = w(u, C_j) * \hat{ctr}(d, q, C_j) \tag{3.7}$$

Example 3: Following the setting in Example 2, that $c\text{-}ctr(d_1, q) = [0.044, 0.039, 0.016]$, given a new user with $W(c) = [0.56, 0.22, 0.22]$, the cohort features for document d_1, query q which is $[osu]$ and the user c, is $z(d_1, q, c) = [0.02464, 0.00858, 0.00352]$.

When a user submits a query q, we estimate their click preference on a document d depending on their cohort membership $w(u, C_j)$ and the cohort click probability $\hat{ctr}(d, q, C_j)$. The weight of cohort membership controls how much we can infer about this user's click behavior based on a cohort's click behavior. If a user belongs to the California cohort with a weight of 0.9 and the Washington cohort with a weight of 0.1, the estimation of their click probability therefore relies mainly on the cohort California, and only slightly on the cohort Washington. We create cohort features for each $\langle d, q, u \rangle$ tuple, and let the ranking algorithm decide the ranking of URL candidates based on these cohort signals.

3.3 Empirical Datasets

To evaluate the effectiveness of our cohort model for enhancing personalization, we apply it to extend the personalization model of a commercial search engine. The existing personalization approach already uses a number of short- and long-term topical, location, and domain preference features, similar to those proposed in the academic literature, e.g., [13, 15, 70, 79, 87] so serves as a strong baseline for our experiments. Note that the baseline approach is built upon a standard search engine that retrieves the most relevant documents via querying. We evaluate our methods retrospectively using logs containing search behavior and the original result ranking. The dataset used in our study is sourced from the log of a popular Web search engine. We mined over two months of logs and extracted events comprising tuples of: query, an ordered list of the top-10 search results returned by the engine, and searcher clicks on those results. The order of the URLs for a query was produced by the personalized production ranker from the search engine. We re-rank results using this enhanced model. This methodology allows us to estimate the effectiveness of our cohort modeling approach.

3.3.1 All Queries

Cohort features, which are based on click history, are good indicators of document relevance for given queries. However, the volume of URL documents is huge, and a large amount of documents are not clicked or shown to many users. Although we incorporated smoothing technique to overcome such sparsity we found in practice, constructing features at a higher level further alleviates the challenge. For instance, we can replace URL documents in our cohort models by URL domains and re-rank using the same personalization approach. Specifically, the symbol d is used to represent a URL domain rather than a URL document. A domain is part of a URL, e.g., URL=http://www.cnn.com/politics, domain=cnn.com.

As stated above, we use the production ranker from the search engine as the baseline for comparison. We then train a new model, with cohort features added. Logs in a two-month period (March 31 2013 to May 28 2013) are used to build cohort membership

vectors and cohort CTRs. We name this time segment as the profiling period. Cohort features are then built for $\langle u, q, d \rangle$ tuples in logs from the following week (May 29 2013 to June 4 2013), which is then divided into training, validation and testing periods. The first three days were used for training, the next two days were used for validation, and the last two days were used for testing. The performance is evaluated by re-ranking top results returned from the baseline ranker. This method has been used successfully in prior personalization studies [13, 70].

Table 3.1 presents the statistics on the datasets used, including the number of search queries (impressions), the number of distinct queries, the number of distinct URL domains, and the numbers of users in our dataset. Besides the comparison on all queries, we also classified queries into various segments to facilitate a more detailed analysis of the performance of our cohort modeling methods.

Table 3.1. Data sets for cohort modeling experiments, All dates from 2013.

Data	Cohort Profiling	Training and Validation	Testing
Date range	03/31 - 05/28	05/29 - 06/02	06/03 - 06/04
#impressions	1,016,333,942	11,615,957	5,352,460
#distinct queries	248,419,356	4,096,337	2,192,327
#distinct domains	25,704,086	3,116,209	2,087,303
#users	23,378,476	1,144,715	739,281

3.3.2 New Queries in User Search History

Previous studies have shown that although searchers frequently submit repeated queries for re-finding purposes, there are also a large fraction of user queries that are new [65, 77]. A new query from a particular user means by definition that user has not submitted it previously (at least not in an observable period such as the two months used for profile building). Given their frequency, new queries are a particular subset where search engines could offer significant benefit, but since there is no user history it is not clear what support they can offer on an individual level. This means that they must resort to global models of all users' on-query behavior. These are queries where cohort modeling may offer particular assistance.

Definition 5 (New Query)**.** *In our experiment, for each user, queries that are shown*

in the testing period but not in the profiling, training and validation periods are defined as new queries. In contrast, queries that appear in all periods are defined as old queries.

In our analysis, we identify new queries for each user and separate them from old queries to evaluate the re-ranking performance of cohort models. To simplify the determination of new queries, we focus on exact match of queries on training and testing periods. The derivation and application of more sophisticated matching methods (e.g., semantically-equivalent queries), is a separate research problem and is reserved for future work. Some pre-processing is applied, including converting queries to lowercase, removing additional whitespace, and deleting punctuation while preserving n-grams for terms joined by punctuation (e.g., asp.net).

3.3.3 Popularity of Queries

Our cohort model leverages group click preferences to estimate individual click preferences. For a user who submitted a query, we identified a cohort of other users who are similar. However, if only a small number of users in the group submitted the same query, the prediction of cohort preference for the query will be biased and not representative of the full cohort. As part of our re-ranking experiments, we wanted to better understand the impact of query popularity on re-ranking performance when cohorts were utilized.

Definition 6 (New Query)**.** *The popularity of a query is determined by the number of distinct users who submitted it during the profiling period, which is denoted by* N_u*. Search has a long tail effect that many tail queries are submitted only one or two times, by a small number of users. Cumulatively, there are a large number of such queries. We divided queries into two datasets: (1) popular:* $N_u \geq 10$*, and (2) unpopular:* $N_u < 10$*, Approximately 30% of distinct queries are popular by our definition.*

3.3.4 Query Entropy

As mentioned earlier, some queries have almost uniform click preference among all users, for example, [facebook] or [amazon] have high CTR their associated sites. For

these cases, individual, cohort, and global preferences are consistent. Thus the cohort model has limited potential to improve retrieval performance. We measure the diversity of clicks among users for each query by computing the query entropy as follows:

$$H(q) = -\sum_{d} \frac{\hat{ctr}(d,q)}{\sum_d \hat{ctr}(d,q)} \log \frac{\hat{ctr}(d,q)}{\sum_d \hat{ctr}(d,q)} \tag{3.8}$$

where $\hat{ctr}(d,q)$ is defined at Equation (5).

We focus on top five URL domains returned as search results ordered by global CTR. Note that we take the natural logarithm. As a result, the maximum entropy value is $\log(5) = 1.6$, and the minimum is zero. If clicks of all users led to the same destination, the value of the entropy will be zero, indicating the query has the smallest variation in click behavior. A high value of entropy indicates the query has large variations.

Click entropy has been used in many studies to evaluate the complexity of queries, e.g., [42]. However, by assuming same search results are shown to all users who submitted the same query, its implementation in those studies only considers the number of clicks and ignores the impression counts. Our data comprises logs of a search engine equipped with personalization. Consequently, URL documents have unequal chance of being shown. Therefore we take advantage of CTR and consider both clicks and impressions.

To examine how the performance of our cohort model relates to the level of query entropy, we separate queries into three subsets: low entropy, medium entropy and high entropy. The corresponding entropy ranges are $[0, 0.2)$, $[0.2, 1.2)$, and $[1.2, 1.6)$. The motivation is that for queries with small entropy on global CTR, it is less likely that cohort click preference differs from global click preference. For queries with large entropy, global clicks are diverse, thus we expect that cohorts can differentiate clicks, and therefore offer better personalized search results.

3.3.5 Acronym Queries

Many acronyms are ambiguous and associate with more than one meanings. For example the intent behind [msg] may differ depending on the user location, e.g., users in

New York City may be more likely to mean Madison Square Garden, whereas the likely intent elsewhere in the United States could be monosodium glutamate. As such, search engine performance can be improved on acronym queries via personalization that considers the location of the searcher as part of the ranking process [70]. To understand the effect of acronym queries on the performance of our models, we used a set of acronyms defined in previous work [75]. From these data, we selected 432,564 acronyms which had a length of 2, 3 and 4 characters. The average number of meanings per acronym was 2.91. We then intersected these with the two-day logs used for testing, resulting in around 11,000 distinct query matches.

3.4 Experiment Results and Findings

We now describe our experimental results. As mentioned earlier, our baseline is the current production ranker in the commercial search engine, which leverages the global relevance of query and URL document, and personalized features including those involving topical, location, and resource preferences. Our cohort model integrates cohort features that are highly similar to those in the baseline. Comparing the models lets us estimate changes in personalization effectiveness attributable to the cohort modeling.

3.4.1 Ranking Models and Evaluation Metrics

Using the described dataset, we train a ranking model Lambda-MART learning algorithm [90] for re-ranking the top ten results of the query. LambdaMART is an extension of LambdaRank [17] which is based on boosted decision trees. It has been shown to be one of the best algorithms for learning to rank. Indeed, an ensemble model in which LambdaMART rankers were the key component won Track 1 of the 2010 Yahoo! Learning to Rank Challenge [18]. Our cohort features are insensitive to ranking algorithm, thus any reasonable learning-to-rank algorithm should also observe relevance gains as we do. We trained four ranking models using three types of predefined cohort features introduced earlier, specifically:

- A model with ODP cohorts only (ODP);

- A model with top-level domain cohorts only (TLD);
- A model with location cohorts only (Location), and;
- A model with all three cohorts, concatenated together (ALL).

We also build cohort models dynamically by clustering users based on contextual features. In next section, we describe the cohort clustering methods and experiment with varying the number of clusters (cohorts), denoted as k.

As described earlier, we collected two months of search logs to construct user profiles, and the next one week of logs for training, validation, and test. Evaluating personalization at scale is challenging; since users can have different intentions for the same query, therefore using third-party relevance labels may be insufficient. To address this concern, we exploit user clicks to obtain personalized relevance judgments for each query-document pair returned for a query. Clicks can be classified into various types by the dwelling time on landing page. If the dwell time is too short, the searcher may be dissatisfied with the search result. In this study, we label URLs with SAT Click (defined earlier) by the user positively, and other URLs negatively. This method for generate click-based judgments has been used in prior personalization studies [15, 70, 88].

We measure the quality of re-ranking using mean reciprocal rank (MRR) and mean average precision (MAP). In both cases, the mean is the average across all impressions, including those where the ranking does not change as a result of the treatment. MAP considers cases where there are multiple SAT clicks (better for informational queries); MRR is focusing only on the rank of the first SAT click. MAP is the mean of the average precision scores for each query,

$$MAP = \frac{1}{N} \frac{\sum_{i=1}^{n} Precision(i) Rel(i)}{\sum_{i=1}^{n} Rel(i)} \tag{3.9}$$

where n is the number of URLs in the impression, ranging from 1 to 10. $Rel(i)$ is an indicator function returning 1 if the URL in the rank i is relevant, otherwise 0. $Precision(i)$ is the precision at cut-off i in the ranked list.

MRR targets the rank of the first relevant document in the result list. It is the average of the reciprocal ranks over all queries,

$$MRR = \frac{1}{N} \sum_{q} \frac{1}{rank(q)} \tag{3.10}$$

where rank(q) is the rank position of the first URL document that received satisfied click for the query q.

Due to proprietary concerns, we do not report absolute values but only relative changes of the cohort model against the baseline: $\Delta MRR = 100 * (MRR(cohort) - MRR(base))$ and $\Delta MAP = 100 * (MAP(cohort) - MAP(base))$.

3.4.2 Research Questions and Findings

To understand the effect of cohorts in personalized search, we answer the following questions in the remainder of the chapter:

- Can our method enhance the baseline generally, for all queries?
- Can we identify particular classes of queries that benefit from our cohort modeling, and to what extent
- Can we improve the relevance further by learning cohorts (rather than the predefined cohorts defined earlier)? (Section 3.5)

We now present the results of our analysis on all queries and on each of the query subsets described in the previous section.

All Queries

We begin with the first question: in general, can cohort models improve the retrieval performance when used in addition to existing personalization method(s)? This helps us understand the overall impact of the cohort modeling on search engine performance. Table 3.2 reports the MAP/MRR gains of our model versus the baseline (the production ranker) along with the standard error of the mean (SEM). The findings presented in the table show our cohort model significantly outperforms the baseline (with paired t-tests).

Results that received SAT clicks by users are promoted by the cohort by the ranking (as can be seen with the low re-ranked@1 percentage in Table 3.2). All types of cohorts are informative. In particular, TLD yields the largest gain, perhaps because it captures differences in expertise of interests (e.g., people selecting en.wikipedia.org rather than a commercial domain). Location cohorts may have achieved the lowest gain because firstly, the baseline already covered the individual location preference; secondly we use state to represent location and it can mask important intra-state movement. A finer grained representation of location may be required, but we also need to consider how best to do that in a scalable manner while ensuring that there are sufficient numbers of users in each cohort. Note that although these changes may appear small, they are averaged across all queries, including many that are unchanged.

Table 3.2. Gains in MAP and MRR over baseline(± SEM).

Cohort	ReRank@1	ΔMAP ± SEM	Δ MRR ± SEM
ODP	0.91%	0.0181 ± 0.0013	0.0187 ± 0.00142
TLD	0.96%	0.0224±0.00140	0.0229±0.00144
Location	0.90%	0.0111±0.00138	0.0113±0.00141
ALL	0.98%	0.0193±0.00140	0.0211±0.00145

A trend that we see in Table 3.2 that is mirrored in all of our experimental findings is that ALL performs as well or less well than the other models. This model re-ranks slightly more results (Table 3.2), meaning that its application is less focused. Also, the presence of multiple cohorts may make the ALL cohort signal noisier. Given the promising gains observed across all queries, we now turn our attention to the various query subsets that were introduced earlier in the paper. In the remainder of this section we present results on each of those subsets defined in Section 3.3. Since the performance of both of the MRR and MAP metrics is similar, we focused on a single metric (MRR) for the remainder of this section.

New Queries

In our dataset, the average ratio of distinct new queries of all queries is about 70% per user, consistent with previous work [71]. It indicates that users submit a large portion of new queries that are not recorded by the search engine previously (at least not in

the past two months, which may be all the engine has access to for a user at query time given profile size limitations at scale). Thus personalization based solely on an individual history is insufficient.

Our cohort model utilizes search and click history of similar users to alleviate the challenge of insufficient data. We split the testing data into two subsets composed of old queries and new queries for each user respectively. Figure 3.1 shows the performance difference over the baseline for each type of cohorts. In this figure and others in this section, the value of zero denotes the original performance of the baseline. The figure shows that indeed our model works well on new queries that haven't been seen previously from a given user. We observe statistically significant gains across all predefined cohorts (all $p < 0.001$). When queries are repeated, the baseline with individual search history may work well, and adding cohort features had little or even slightly negative effect by introducing noise (as is evidenced by the blue bars and negative MRR changes). It is also interesting to observe that the ODP (topic) cohort performed best for new queries. One possible explanation for this finding is that queries without an exact match that appear in the users' history are most likely to be informational, and therefore benefit most from users with similar topical interests.

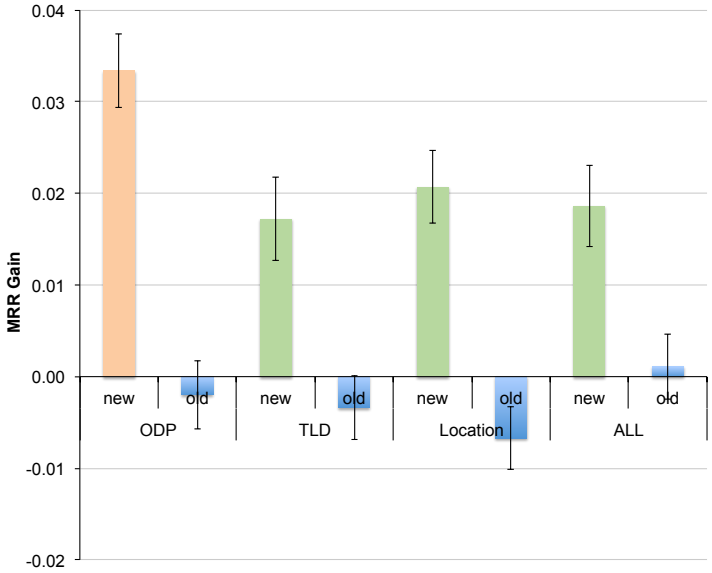


Figure 3.1. Gains in MRR over baseline for each cohort type for new and old queries from each user (±SEM)

Query Popularity

We are also interested in the effect of query popularity on the performance of the cohort modeling, in order to understand how sensitive our model to the size of cohorts. Figure 3.2 shows MRR gains on the popular and unpopular query sets, as described earlier. The performance gain on popular set is much larger than that in unpopular set. Again all differences are significant given the extent of the gains and large sample sizes ($p < 0.001$). The results match our expectation. When a query is searched by many users, we can distinguish cohort preference accurately. However, if a query is searched by only few people, the estimation is less accurate.

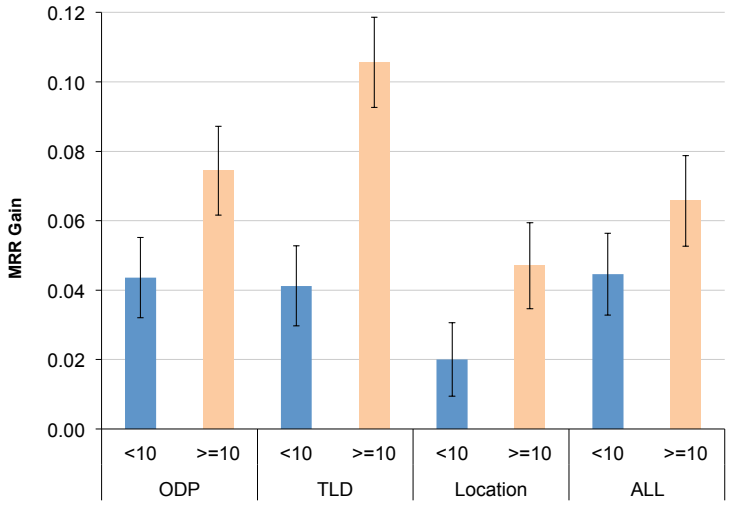


Figure 3.2. Gains in MRR over baseline for each cohort type for difference in the popularity of the query(±SEM)

Query Entropy

We conjectured that since a large entropy implies diverse clicks on URLs, separating and assigning weights on clicks by cohorts can help identify an individual's preference more accurately. Therefore we expect queries with large entropy will obtain large benefit from the cohort model. Figure 3.3 presents the MRR gain over the baseline for the three query entropy bins: low, medium, and high.

The results shown in the figure confirm our intuition regarding where personalization

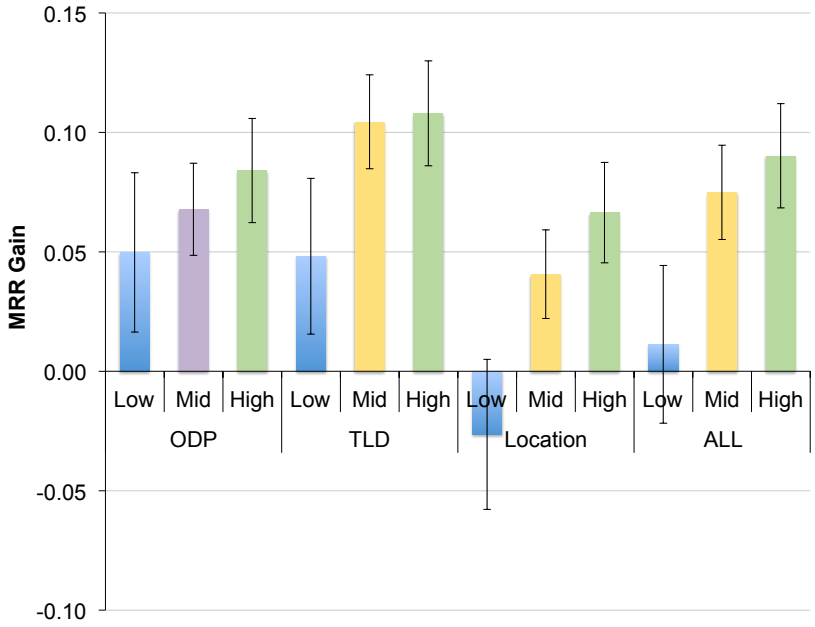


Figure 3.3. Gains in MRR over baseline for each cohort type for different query entropy bins(±SEM)

enhancements might help. On all types of cohorts, the query set with low entropy received smallest gain over the baseline. Queries with medium and high entropy obtained larger performance increases (all statistically significant, $p < 0.001$). This suggests that one strategy to realize strong gains from the cohorts may be to bypass low entropy queries and only apply cohort models on queries with medium or higher entropy. As observed in other analyses in this section, we also observe that the Location cohorts achieved the smallest gain, and even resulted in a loss for low entropy queries. This may be related to the use of state level cohort features, which may be too coarse grained to capture individual click preferences. More work is required to determine how best to represent and apply location information for cohort modeling.

Acronym Queries

As mentioned earlier, acronym queries such as [acl], [atm], etc. are a specific set of ambiguous queries where personalization may help [70]. We examine the effectiveness of our cohort modeling methods on the subset of acronym queries described earlier. Table 3.3 shows the MAP and MRR gains over the baseline for this query set (all

significant at $p < 0.001$). The results clearly demonstrate extremely strong gains in performance for the subset of acronym queries for each of the cohort types studied. Although this may only be a relatively small query set, it is encouraging to see the significant gains in acronym queries. It is clear from the findings presented in the section so far that there are a broad range of different query classes for which the cohort modeling performs well. However, the performance of the ALL model was generally slightly lower than the other models. There may be a better way to combining the cohorts and the next section we describe an approach to learn cohorts dynamically.

Table 3.3. Gains in MAP and MRR over baseline for acronym queries(± SEM).

Cohort	Δ MAP ± SEM	Δ MRR ± SEM
ODP	0.1566 ±0.0562	0.1622 ±0.0568
TLD	0.1585 ±0.0568	0.1519 ±0.0578
Location	0.1450 ±0.0535	0.1552 ±0.0544
ALL	0.1212 ±0.0544	0.1265 ±0.0553

3.5 Learned Cohort Models

The results in the previous section show that our cohort modeling techniques using pre-defined features can more accurately estimate users' individual click preferences (as represented via increased SAT clicks) than our competitive baseline. A challenge of this approach is the tradeoff between the number of cohorts and the prediction power of cohorts on individuals. One can define more fine-grained cohorts, for instance, including second or even lower levels of ODP, and changing locations from state to city or ZIP-code level. However, more cohorts result in fewer users in one cohort and less reliable CTR estimation. To overcome this tradeoff, we propose an alternative that generates cohorts automatically via clustering. The objective is to construct homogeneous clusters (cohorts) given a large number of features.

3.5.1 Clustering Method

In this section, we discuss how we learn cohorts automatically using k-means clustering. Each user is represented by a vector of contextual features $x_u \in R^d$, which is concatenated from the three sets of pre-defined cohort features on topic, location and top level domain. The dimension of the feature vector is 99 in our setting. The objective is to assign users into cohorts. A map-reduce implementation of k-means algorithm is applied to cluster users into k clusters (cohorts). We then define two implementations of customized cohort membership vector.

Definition 7 (Learned Membership, Hard)**.** *Given learned k-cohorts, a user's cohort membership vector is defined as a k-tuple* $W(u) = [w(u,1), w(u,2), \ldots, w(u,k)]$. *The membership to the* $i-th$ *cohort depends on whether the user is assigned to the* $i-th$ *cluster, that is* $w(u,i) = 1$ *if the user is in the* $i-th$ *cluster, otherwise* $w(u,i) = 0$.

Definition 8 (Learned Membership, Soft)**.** *Given learned k cohorts, a user's cohort membership vector is defined as a k-tuple* $W(u) = [w(u,1), w(u,2), \ldots, w(u,k)]$. *The membership to* $i-th$ *cohort is determined by the minimum Euclidean distance between the user and the centroid. Let the centroids learned by k-means be* $\mu_1, \mu_2, \ldots, \mu_K$. *Ideally the Gaussian Mixture Model could achieve the goal with additional computational overhead. In this large-scale study, we leverage the k-means results and assign cluster membership as follows:*

$$w(u, C_j) = p(C_j|x_u) = \frac{exp(\frac{-d(x_u,\mu_j)^2}{\alpha^2})}{\sum_{i=1}^{K} exp(\frac{-d(x_u,\mu_i)^2}{\alpha^2})} \tag{3.11}$$

where $d(x_u, \mu_j)$ *is Euclidean distance between the user vector* x_u *and the centroid* μ_j, *and* α *is estimated from average distance between centroids. This is a simplified implementation of the Gaussian Mixture Model having identity covariance.*

As to the hard membership assignment, each user has only one non-zero cohort membership, which may be preferable on many clusters with large k. For users with diverse preferences, it is natural to allow multiple cluster membership for them. Therefore soft membership may produce higher performance gain.

3.5.2 Evaluating Clustered Cohorts

We compare the performance of re-ranking by clustered cohorts against the model with predefined ALL cohorts. We evaluate MRR change on a selected subset of queries. The subset of queries is new queries with entropy larger or equal to 0.2. This is a query subset on which we observed a large performance gain with predefined cohort model. We experiment on learned cohorts using select probes of $k \in 5, 10, 30, 50, 70$, with soft assignment of cohort membership, which is expected to be better. Figure 4 displays the experimental results. Note that the baseline ranker (where MRR gain = 0) already contained global CTR as a feature, which is equivalent to $k = 1$. We therefore do not report performance at $k = 1$ in Figure 3.4.

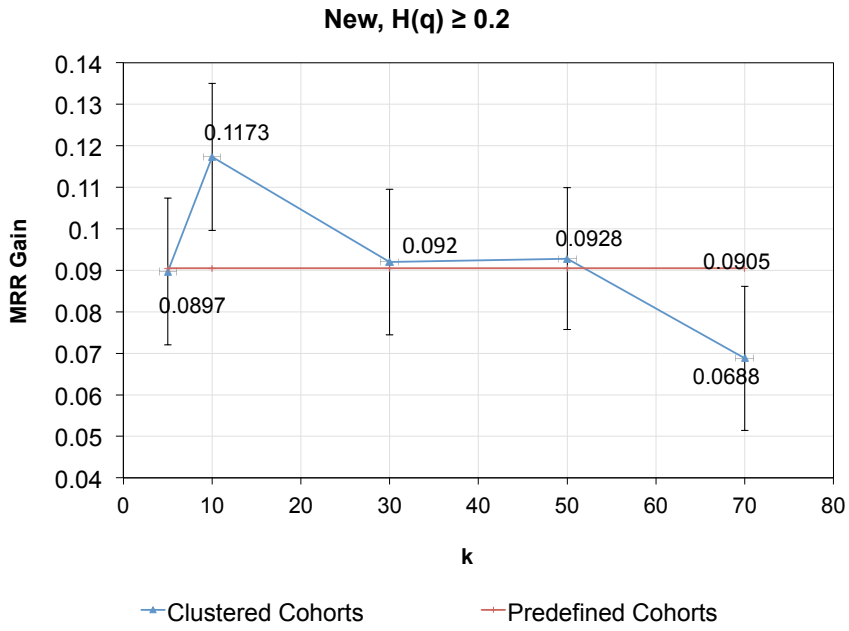


Figure 3.4. Gains in MRR over the baseline for clustered cohorts versus predefined cohorts for different k for a selected query set(±SEM).

The figure shows that we can observe the largest MRR gain by the clustered cohorts model when k=10. The MRR gains are slightly larger than the model with predefined cohorts in $k = 30, 50$, and slightly smaller in $k = 5$. The MRR gain decreases sharply when k becomes too large, e.g., 70. We did not perform a full sweep of k given computational and time constraints, but the findings are still informative and the gains over

ALL at k=10 are significant.

We also evaluated the model on other subsets of queries and observed similar results: largest gain is obtained in small k and largest k has smaller gain than the predefined cohorts model. One possible explanation for this is that the user features are sparse and that as k increases, the reliability of the cohort signal in each cluster degrades. The fact that we can obtain strong performance by reducing the dimensionality of the features from 99 to 10 is promising for large-scale deployment (as it means compact user profiles). It also reveals the opportunity of profiling users with more subtle and sparse features than projecting to a few principal dimensions, as was done with the pre-defined cohorts.

As mentioned previously, we can employ hard or soft clustering, depending on whether we want users to be in a single cohort only (hard) or appear in multiple cohorts potentially with different weights (soft). The implications of this include the nature of the profile that a search engine should store (e.g., a single cohort identifier per user versus a set of cohort identifiers per user). In the analysis above, we employed soft clustering. One concern we had regarding hard clustering is that it may lead to an inaccurate CTR estimation for users who are not close to the cluster centroid. To better understand the impact of this decision, we compare the performance of models with cohorts by hard-clustering, soft-clustering and predefined features w.r.t. the baseline as in other experiments thus far. Table 3.4 presents the findings of this analysis.

Table 3.4. Gains in MAP and MRR over baseline for acronym queries(± SEM).

Metric	Hard Membership	Soft Membership	Predefined Cohorts
Δ MAP	0.0731±0.0158	0.1143±0.0170	0.0932±0.0172
Δ MRR	0.0737±0.0165	0.1173±0.0177	0.0905±0.0180

We can see from the table that cohorts created using hard membership achieved the smallest performance gain, and are worse than those by predefined cohorts. Soft performs significantly better than hard; other differences are not significant. The results suggest that finding weights to assign to each cohort is important for estimating individual preferences. Users also have variations inside a cohort, and their preferences cannot simply be generalized by one cohort.

3.5.3 Preference Analysis

Given that we have these different ways to identify cohorts, we were interested in understanding the relationship between existing search engine results and global/cohort preference. To show that our performance improvement on personalization is not simply caused by gathering more features for the ranking algorithm, we conduct analysis on search logs and investigate whether cohorts manifest unique preference, which is directed by users in the cohort. For a query with many candidate results, global CTR can offer a ranking of URL candidates. We refer to this here as global choice. In each identified cohort, cohort CTR can yield a ranking as well, and we refer to this as cohort choice. We focus on the difference between global choice and cohort choice of the top-ranked result.

Definition 9 (DiffTop)**.** *The DiffTop for a given query q is an m-tuple vector D(q)* $= [diff(q, C_1), diff(q, C_2), \ldots, diff(q, C_m)]$*. Each value is a binary value to indicate whether a cohort has unique preference. We denote the top ranked URL domain d by the cohort choice of j-th cohort as* d_{C_j}*, and by global choice as* d_G*. If* $d_(C_j) \neq d_G$*, we set* $diff(q, C_j) = 1$*, otherwise 0.*

Among selected logs in profiling period, we choose queries with at least two distinct URL domains clicked, and count how many are inconsistent in the cohort choice and global choice. Our result shows 2% of distinct queries demonstrated unique preference by at least one cohort. The ratio appears small, but considering the total number of queries is extremely large, and the fact that we focus on clicks on domain level in the top position only, it is still a strong signal on the potential of cohorts.

There are cases that the values of cohort CTR for the top and the second top URL domains are very close, e.g., equally small. The top and the second top URL domains have similar cohort preference. To address such subtle scenarios, we defined a weighted DiffTop measure as follows.

Definition 10 (Weighted DiffTop)**.** *: The weighted DiffTop for query q is a m-tuple* $w\text{-}D(q) = [w\text{-}diff(q, C_1), w\text{-}diff(q, C_2), \ldots, w\text{-}diff(q, C_m)]$*. Each value measures the degree of unique preference by the cohort. We define the decrease delta (*Δ*) to measure*

the difference in the click probability between the top and the second top URL domain.

$$\Delta(d_1, d_2, q, C_j) = \frac{\hat{ctr}(d_1, q, C_j) - \hat{ctr}(d_2, q, C_j)}{\hat{ctr}(d_1, q, C_j)} \tag{3.12}$$

where d_1 is the url domain in top position, and d_2 the one is the second position. The DiffTop is then weighted by Δ as follows:

$$w\text{-}diff(q, C_j) = diff(q, C_j) * \Delta(d_1, d_2, q, C_j) \tag{3.13}$$

If Δ equals zero, the top candidate is less dominant in the cohort, thus makes the DiffTop a weaker signal about unique preference for this cohort. To compare across cohorts, we average weighted DiffTop across queries for each cohort as $D(C_j) = \sum_q w\text{-}diff(q, C_j)/diff(q, C_j)$. Such weight is then aggregated for a particular cohort type. Taking ODP cohorts for example, $D(C^O) = \sum_j D(C_j)/m$, where m is the number of cohorts of type ODP. If the average value is large, it implies that members of the cohort behave differently than non-members. We compare cohorts by ODP, TLD, Location features, also include clustered cohorts with $k = 10$. Figure 3.5 shows the aggregated weighted DiffTop value. At least two insights can be made. The first is that all cohorts have high average DiffTop weights in general. This shows that our selected features are useful in distinguishing cohort choice and global choice. The second is that ODP and Clustered cohorts are more informative than TLD and Location, perhaps because they are denser.

3.6 Discussion and Conclusion

We have proposed an approach for using cohorts of searchers similar along one or more dimensions to enhance Web search personalization. To understand the value of these cohorts we performed an extensive set of experiments with predefined cohorts as well as cohorts dynamically learned from behavioral data, and for different query sets, including acronyms and queries previously unseen from a given user. These are scenarios where we would like to be able to employ personalization but often it does not

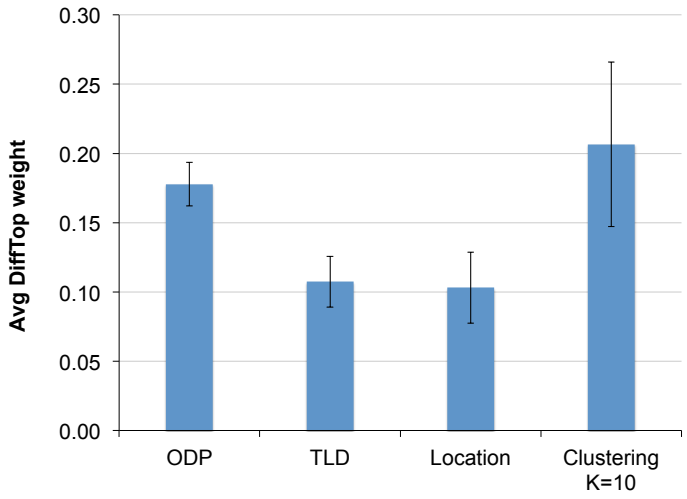


Figure 3.5. Average DiffTop weight for each cohort (±SEM).

succeed given insufficient data about the interests of individual users. The results of our experiments have clearly demonstrated the value of cohorts, especially for ambiguous and new queries from users, where our observed gains over a production ranker appear to be most significant.

In our experiments, we used a competitive baseline a ranking algorithm that already had personalization signals based on a number of personal and contextual features for individual searchers. Despite such attention to representing the individual user's interests, the cohort-based models presented in this paper were still able to enhance the strong personalization baseline and achieve significant gains. This is promising as it suggests that we can learn how to integrate the cohort signals and make decisions about when to use them in combination with individual signals when both are present, or in isolation when only cohort signals are available. That said, further experiments are necessary with other personalization models to assess the generalizability of our findings to other settings.

The pre-defined cohorts have the disadvantage that they require system designers to select important features manually in advance. Using unsupervised clustering we circumvented this problem and learned cohorts dynamically. We are pleased that using cluster-generated cohorts that outperformed the pre-defined cohorts. However, the

success of any clustering method is dependent on the features that are used. In this paper we used a set of features associated with topical preference, location, and top-level domain preference, but there are other viable alternatives (e.g., demographics, social network cliques) and we need to explore their effectiveness in detail.

We have shown that that best performance from cohorts learned via k-means clustering is attained when we set k=10. In a production search engine handling millions of users and billions of queries, the amount of space that can be devoted to each user is minimal. We have shown that for each user we would only have to store a small amount of additional information about their cohorts in each user's profile, e.g., a single membership bit for each of the 10 cohorts.

Overall, it is clear that there is significant potential value from modeling cohorts in search personalization. In future work we could develop more sophisticated cohort representations which leverage other sources of data from beyond search (e.g., browsing signals) and social network information depicting relationships between cohort members (and cliques, group dynamics, etc.), and experiment with other clustering methods, such as hierarchical clustering, which may yield richer and more homogenous clusters.

Chapter 4

Comparable Groups

The assessment of a researcher often involves examining researchers in her reference group. There are many scenarios that require a group of comparable people for reference, including applying for funding, evaluation for tenure and searching for review peers. This task is usually done manually. In this chapter, we study the problem of finding a group of comparable researchers for any given researcher automatically.

4.1 Introduction

To compare a pair of researchers, we need metrics to assess their research impact. There are limited number of existing metrics to evaluate research impact at the individual level, for example, h-index [34] and g-index [28]. Such metrics are mainly based on raw citation counts, i.e., the number of papers citing a given paper. It has several limitations. Garfield [31] argued that citation counts are a function of many influencing factors besides scientific quality, such as area of research, number of co-authors, the language used in the paper, and so on. Citation behavior also varies across sub-fields. In many cases few if any citations are recorded, even though the paper's influence may go beyond this measure of impact [55]. Moreover, citation counts evolve over time. Papers published longer ago are more likely to cumulate higher counts than those released more recently.

A researcher's research interest may shift over time, as well as their research productivity. Citation count fails to capture such change, thus it is not a reliable metric to determine whether two researchers are similar and comparable. We seek for the comparison that takes time into account. We then build a research trajectory composed by the sequence of *venue*, *paper* and *year* for each researcher. The sequence is

ordered by year. Venues often have ranks to indicate its quality and reputation. Since a venue contains a collection of papers, and the qualify of papers directly connect to the quality of the venue, we thus use the rank of venue as the surrogate as the qualify of the publication [1].

We then propose a trajectory matching algorithm to compare two researchers. The distance between two researchers is calculated by allowing some mismatches, and counting the number of deletion and insertion operations necessary to harmonize the two sequences. We first consider the sequence of venue ranks, then add the research topics which are learned from paper's abstract and title. One interesting research question is that whether we can find senior researchers who have similar trajectory in the early stage as the junior researcher in a query. With simple modification of our matching algorithm, we can find more senior researchers for any given junior researcher. This results can be especially useful when trying to predict the trajectory of a researcher for years to come.

In this work, we focus on the computer science domain and use data from online services which index research work, specifically, DBLP[2] and arXiv.org [3]. Since researchers in computer science often prefer conference publications, and the data available on the web is also skewed to conferences, our work focus on conference papers only. Other disciplines may favor journals or other types of publications. Our approach apply equally to these settings. As to the source of venue ranking, there are several services provide it, e.g. Google Scholar Metrics[4], Microsoft Academic [5]and CORE[6]. Their ranking mechanisms vary therefore the results are different. Sometimes many ranking results appear surprising. How to obtain objective topic dependent venue ranks remains an interesting and open research problem. To simplify the process and focus on the comparison algorithms, we take advantage of CORE, which is an existing subject-dependent ranking

[1] alternate methods for assigning a quality to a paper can also be used.

[2] http://www.informatik.uni-trier.de/~ley/db/

[3] http://arxiv.org/

[4] http://scholar.google.com/intl/en/scholar/metrics.html

[5] http://academic.research.microsoft.com/

[6] http://core.edu.au/index.php/categories/conference%20rankings

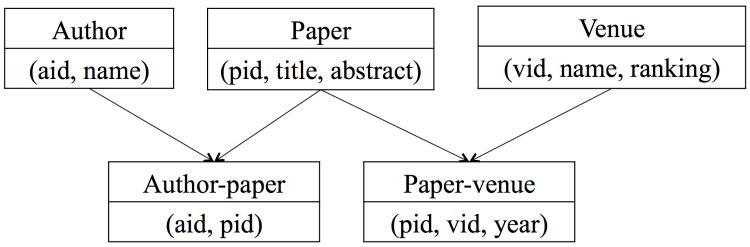


Figure 4.1. Tables and data schemas

that covers broadly known conferences.

In the rest of the chapter, we first show results of exploratory analysis over the large-scale scholarly data that contain millions of researchers and publications. We then describe our algorithm and demonstrate how to compare researchers and detect comparable relations automatically for the first time. Moreover, we provide the map-reduce version of our algorithm in order to handle the still-increasing amount of data. Finally we discuss there are many interesting open problems for future work.

4.2 Problem Setting

Let $\mathcal{A}$ be the set of authors[7], $\mathcal{P}$ be the set of papers, $\mathcal{V}$ be the set of publishing venues. Each paper $p \in \mathcal{P}$ is associated with a set of authors $a \in \mathcal{A}$. The paper is published in a venue $v \in \mathcal{V}$ at the time t. We assume that for each venue v we have a score which corresponds to the rating of the venue, where higher score implies higher rank and quality. In database terms, our system contains three entity tables: author, paper, venue, and two relation tables: author-paper, paper-venue (Figure 4.1). The problem we are interested in is, given the database of researchers and their publications, for any pair of researchers (a_i, a_j), to measure the extent to which they are comparable, under various notions of similiarity.

[7]We use author and researcher interchangeably

Table 4.1. Dataset statistics

id	dataset name	#papers	#authors
1	DBLP	2,764,012	1,018,698
2	ArnetMiner	1,572,278	309,978
3	Our Corpus	1,558,500	291, 312

4.3 Data and Analysis

Our empirical analysis is based on two datasets available on the web: bibliographical information about computer science journals and proceedings from DBLP, and a citation network dataset from arnetMiner [76]. Both datasets collect scholarly data up to January 2011. The venue name in the arnetMiner data is noisy, since the name of a conference can appear in multiple forms, for example, full phrases of conference name, abbreviations, and abbreviations plus volume numbers and so on. We found abbreviations of conference names are used consistently in DBLP. Thus we extract the abbreviation and combine other information to identify the venue for each paper. However, the data from arnetMiner contains rich information including title, abstract and most importantly, year. We match data from both datasets by the paper name and author names, then create a corpus with the joint data. Table 4.1 lists the statistics of datasets, including the one we derived. We first analyze the data characteristics before using it to answer our questions.

4.3.1 Exploratory Data Analysis

There are various phases to the career of a researcher. A graduate student may enter industry and stop their research activity. A faculty member may spend months or years away from their home topic during a sabbatical. A researcher may retire or switch to a different area. There is no convenient way to learn about these phases from available data. For simplicity, we define the time from the first publication to the last publication as the research period.

Definition 11 (Research Period)**.** *Given a researcher a, and the year sequence associated to publications $T(a) = t_{a,1}t_{a,2}\dots t_{a,n}$, the length of research period is defined as*

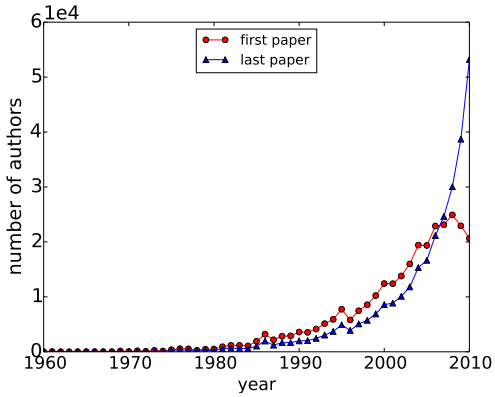


Figure 4.2. Start and end years.

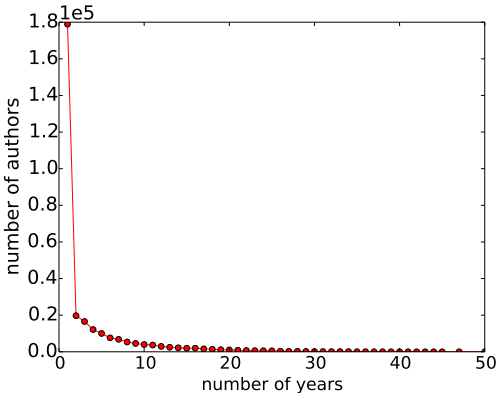


Figure 4.3. Research period length.

the time gap between the last and first publications: $w_a = t_{a,n} - t_{a,1} + 1$.

We extract first and last years of publications for each user, and compute the length of research period. Figure 4.2 plots in each year, how many researchers published their first and last paper. From 1990 to 2000, the number of authors starting their career from the year steadily exceeds that of those who ended their research work. After the year 2006, the relationship is reversed (likely due to "end effects" from using a snapshot of data). Figure 4.3 shows the relation between number of authors and the number of years in their research period. 61.4% of researchers published papers only in one year (typical examples are students who published one paper and then graduated, and researchers from other areas who published one paper in a CS venue). The number of researchers whose research period is at least 10 years is $29,671$. These authors account for 10% of all researchers but are connected to 52.6% of papers in our corpus.

We use two metrics to understand the career trajectory of each researcher: burst speed and half year speed.

Definition 12 (Burst Speed)**.** *Given an author a, the burst speed is defined as the number of years to reach her/his first bursty year. The bursty year is the year that with largest productive score, which is calculated as*

$$v_a(t) = \frac{|P_a(t)|}{|P_a|} - \frac{1}{w_a}$$

where w_a is the length of research period for the author a, and $\mathcal{P}_a(t)$ is the set of papers

by the author a at the year t.

Definition 13 (Half-Speed)**.** *Given an author a, the half-speed is defined as the least number of years she/he took to reach half of her/his total publications, which is calculated as:*

$$h_a = \min_n\{n : \frac{\sum_{s=1}^{n}|P_a(t_{a,s})|}{|P_a|} \geq \frac{1}{2}\} - 1$$

Using these concepts, we can ask the following questions:

- Given a year t, how many authors are in their bursty year?
- Given an author a, which year is his/her bursty year?
- What are the average half-speed and burst-speed?

We select researchers whose research period length is larger or equal to **10** years. Among these researchers, the average number of publications is **13.8**, average research period length is **15.83**, average productive score is **0.212**, the average burst speed is **6.32**, and the average half speed is **7.74**. The result shows that authors reached half of their publications quickly after their bursty year. Figure 4.4 shows the distribution of burst speed and half speed. We observed that authors with zero half speed and burst speed account for 9.5%, and 35% of all selected authors respectively. In addition, many authors published most papers in their first year (recall, these are authors who are active for over a ten year period).

We now show the correlation between the number of publications and the above definitions: research period length, burst speed and half speed. For each author with a research period of over ten years, we count the number of publications associated with her/him, find the research period length, and compute the burst and half speed. Then we take the average of research period length, burst and half speed, given a value x of number of publications. Figure 4.5 tells that for authors who have x publications, what are their average research period length, burst speed and half speed. We set a threshold of 120 publications in the plot because there are too few points above the threshold. The figure shows logarithmic-like behavior.

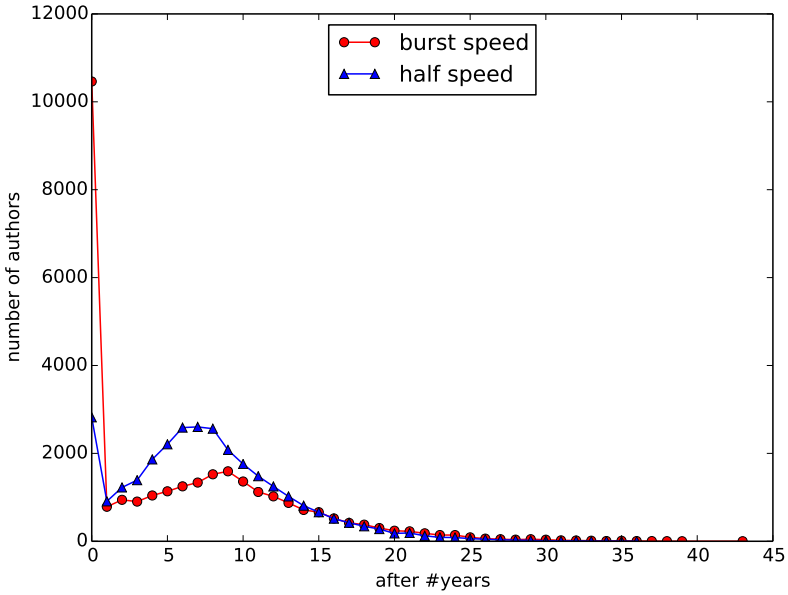


Figure 4.4. Burst speed and half speed

Table 4.2. Sequence example of a researcher

Time	T(a)	$t_{a,1}$	$t_{a,2}$	...	$t_{a,n}$
Papers	P(a)	$p_{a,1}$	$p_{a,2}$	...	$p_{a,n}$
Venues	V(a)	$v_{a,1}$	$v_{a,2}$	...	$v_{a,n}$

4.4 Algorithms to Comparable Researchers

We now describe how we define measures to evaluate and compare researchers. As mentioned above, we use venue ranking as a basis by which to evaluate a researcher. During the research period, the author publishes papers year by year, thus forming a sequence of publications, which are associated with various attributes (venue, title, abstract etc.). Table 4.2 shows an example of sequences, where $T(a)$ is the time sequence, $P(a)$ are publications and $V(a)$ gives venues of publication. The unit of time sequence is year, and publications and venues are ordered according to the time sequence.

We use an existing ranking[8] of broadly known conferences across sub-fields in computer science. This CORE ranking covers 1006 conferences, while DBLP lists 4000 unique conference names. The fraction of papers published in the ranked conferences

[8] `http://www.cs.wm.edu/~srgian/tier-conf-final2007.html`

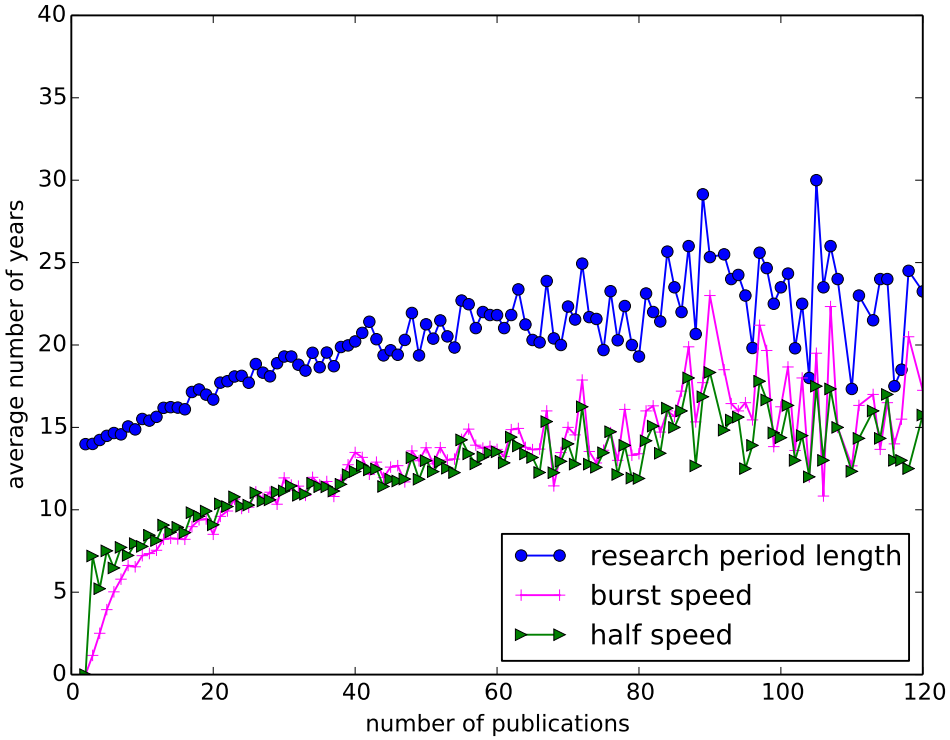


Figure 4.5. Correlations with #publications

is 44%. So while there is missing data, the coverage is still satisfying. Future work is to obtain a ranking that covers more venues.

The CORE ranking breaks venues into five categories: {A+, A, B, C, L}, where A+ is the best. We map these five categories to integer scores: $\{5, 4, 3, 2, 1\}$, in the way that 'A+' matches '5' to 'L' matches '1'. We consider two approaches to using venue scores:

Definition 14 (Venue Score: Categorical)**.** *This approach treats the score as a categorical variable, which takes values from the set* $\{5, 4, 3, 2, 1\}$*. There is no order between two scores. The relationship between two venue scores are equal and non-equal.*

Definition 15 (Venue Score: Ordinal)**.** *In this case, score is a ordinal variable, taking values from the set* $\{5, 4, 3, 2, 1\}$*. The order of the venue score is determined by the integer value of the score.*

4.4.1 On Venue Score

A key question is whether the venue score is a suitable measure on which to rank authors. We compare against two broadly accepted existing metrics: h-index and g-index, and collect values of these two metrics for authors in our corpus. We define an evaluation metric for researchers solely based on the venue score as follows.

Definition 16 (v-index)**.** *Given a researcher a, the v-index is the sum of the (ordinal) venue scores of all publications by her/him.*

$$v\text{-}index = \sum_{i=1}^{n} v_{a,i}.$$

in which n is the number of publications by the author.

Among all users in our database, the minimum of v-index is 2, the maximum is 1229, the mean is 53.12 and the median is 32. Large values on v-index only occur few times in our corpus. We compute the distribution of v-index as a function of h-index and g-index. Figure 4.6 shows the results. For clear visualization, we plot instances with v-index less than or equal to 800, which covers most researchers. The x-axis lists each value of v-index, and the y-axis shows the average value of h-index or g-index given the x. We found for most cases, v-index has positive linear correlation with h-index and g-index. Outliers appear at very large values in each index. We conclude that venue score is an acceptable metric by which to evaluate a researcher's research output.

Given a sequence of venue scores for each author $V(a)$, we compute the distance between two authors by matching the two sequences. This is computed with the well-known Wagner-Fischer dynamic programing algorithm [82]. We discuss how to apply the algorithm in our setting.

Given two sequences $S = s_1 s_2 \dots s_n$ and $R = r_1 r_2 \dots r_m$ over the alphabet Σ, the

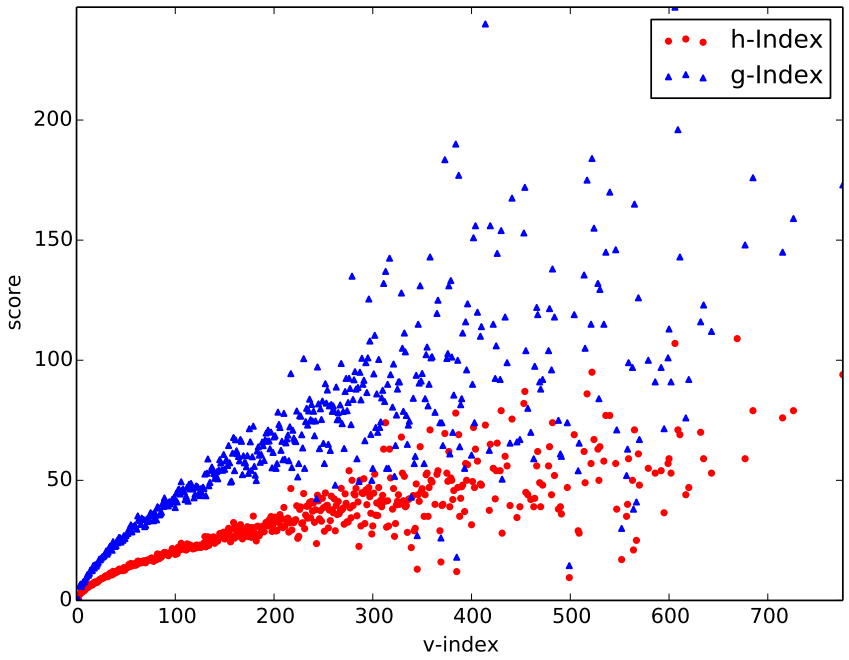


Figure 4.6. Correlations with h-index and g-index

matching score between a pair (s_i, r_j), where $s_i, r_j \in \Sigma \cup \{-\}$, is as follows:

$$d(s_i, r_j) = \begin{cases} 0, & \text{if } s_i = r_j \\ 1, & \text{if } s_i \neq r_j \\ 1, & \text{if } s_i = - \text{ or } r_j = - \end{cases}$$

The symbol "$-$" means a gap for insertion or deletion in the alignment. Dynamic programming is used to compute the optimal alignment of two sequences.

$$D(i,j) = \min \begin{cases} D(i-1, j-1) + d(s_i, r_j) & \text{match or mismatch} \\ D(i-1, j) + d(s_i, -), & \text{insertion} \\ D(i, j-1) + d(-, r_j), & \text{deletion} \end{cases}$$

where $D(i, 0) = i$ and $D(0, j) = j$. In the end, $D(n, m)$ returns the minimum number operations needed to match these two sequences. A direct application of the algorithm

Table 4.3. An optimal sequence matching

S = $V(a_i)$	5	4	4	3	4
R = $V(a_j)$	5	4	5	-	4
d	0	0	1	1	0

is to treat venue score as categorical variable. Table 4.3 shows an example of two researchers' sequences and their distance, where $D(a_i, a_j) = 2$.

It is likely that an author publishes more than one paper a year. Within a year, we can either randomly order the papers, or apply an ordering of the venue scores to form the sequence. A more sophisticated approach is considering sequences of sets rather than points. Then the distance of two positions in two sequences can be computed by the jaccard distance of sets. That is, $d(s_i, r_j) = 1 - |s_i \cap r_j|/|s_i \cup r_j|$, where s_i is the set of publications in $i-$th year in the sequence S, and r_j is the set in $j-$th year in the sequence R. For insertion and deletion operations, the empty set $\emptyset$ is used for the gap. The resulting distance is used to define the comparable relation between researchers:

Definition 17 (Comparable Relation)**.** *Given a researcher a, and the distance between a and other authors, we sort authors by the distance in ascending order. We say that the top k authors are* comparable *to the given researcher a.*

We experiment this approach on our corpus with authors whose research period is larger than 10 years. We sort venue scores in descending order within a year, compose a sequence of venue scores for each author, and compute the distance between each pair following by our algorithm. With distances to every other researcher computed, we determine comparable authors for any given researcher by above definition. Here we set the threshold k to be 20. For brevity, we show results of two examples and their comparable people in the Table 4.4. The first example is for the researcher "Judea Pearl", who mainly focuses on research in machine learning and artificial intelligence. It is perhaps surprising that our approach returns many researchers in the same or related research areas. On the other hand, for the researcher "Dimitris N. Metaxas", who works on compute vision, the results we returned contain researchers in various topics, for example, "Kunle Olukotun" is a pioneer of multi-core processors. Among all researchers, the average distance to their comparable authors is 16.51 ± 11.38, the

minimum is 11.12 and maximum is 268.13.

Table 4.4. Case study of edit distance

Researcher	Top 20 Comparable Researchers	Average Distance
Judea Pearl	Craig Boutilier, Surajit Chaudhuri, Manfred K. Warmuth, Satinder P. Singh, Yoram Singer, Michael J. Kearns, Eyal Kushilevitz, Geoffrey E. Hinton, Silvio Micali, Avrim Blum, Shafi Goldwasser, Robert E. Schapire, Piotr Indyk, Daniel S. Weld, Andrew W. Moore, Stuart J. Russell, Jon M. Kleinberg, Jeffrey D. Ullman, Eric Horvitz, Nick Koudas	16.2
Dimitris N. Metaxas	Kunle Olukotun, Ken Kennedy, James R. Larus, Orna Grumberg, Ji-Rong Wen, A. Prasad Sistla, William T. Freeman, Richard Szeliski, Xiaolin Wu, Uzi Vishkin , Yiming Yang,Thomas G. Dietterich, Stefano Soatto, Dean M. Tullsen, Hwee Tou Ng, Christopher D. Manning, Vijay V. Vazirani, John Riedl, Robert Morris, E. Allen Emerson	21.25

4.4.2 On Topic Similarity

Although our corpus is focused on computer scientists, the computer science discipline spans a range of topics from theoretical studies of algorithms to computing systems in hardware and software. For real world applications, it is more common to compare researchers who work on the same or similar areas. When the pool of candidates is filtered before the evaluation and comparison, our method can be directly applied. If such prior information is not available, we propose to learn topic interests of researchers then compare them automatically. In other words, we can detect both *similar* and *comparable* people simultaneously. Our main intuition is that the matching distance of two points in two sequences can depend on both venue score and topic similarity.

We design a new distance metric to integrate both topic similarity and venue quality. Given two sequences S and R corresponding to authors a_S and a_R, the matching between i-th point in S and j-th point in R is calculated as

$$d(s_i, r_j) = |v_i - v_j + \epsilon| \cdot w(p_i, p_j)$$

where v_i and p_i are i-th venue score and paper for author a_S; v_j and p_j are the corresponding venue score and paper for author a_R (with venue score as an ordinal variable);

and ϵ is a small constant, discussed below. The topic distance $w(p_i, p_j)$ depends on the topic similarity of two papers p_i and p_j, and is computed as $w(p_i, p_j) = 1 - \text{sim}(p_i, p_j)$. The value of topic similarity of two papers $\text{sim}(p_i, p_j)$ is in [0,1]. If two papers are on similar topics, the topic distance is small, otherwise it is big.

When venue scores v_i, v_j are the same, in the previous algorithm, the distance is zero. However the topic distance might be large. We introduce the constant ϵ to include the topic distance for points with same venue score. In our experiments, ϵ is set to be 0.1. Based on the above definition, we see if the topic distance is small and the venue score distance is small, the distance between these two points is small.

The topic similarity between two papers is computed based on the content of papers. In our corpus, we have the title for all papers, and abstracts for about a third of papers. To discover topics for papers, we implement Latent Dirichlet Allocation (LDA) [35]. We treat the concatenation of title and abstract of a paper as a document. Topics are derived from the whole corpus. We then obtain the topic distribution for each paper. The main parameter is the number of topics. We experimented with 20, 50 and 100 topics, with manual validation on frequent words in each topic, and select the number of topics which provides the best presentation of topics.

Given the topic distribution for each paper, we can compute the topic similarity via cosine similarity, or Jensen-Shannon divergence, etc. We use cosine similarity in our examples. With the new distance metric, the dynamic programming formula is modified to the following.

$$D(i,j) = \min \begin{cases} D(i-1, j-1) + d(s_i, r_j) & \text{match or mismatch} \\ D(i-1, j) + v(s_i), & \text{insertion} \\ D(i, j-1) + v(r_j), & \text{deletion} \end{cases}$$

We present some examples of researchers from different topics in our results in Table 4.5. In general, we found results for each researcher are closer in research topics. See, for example, the output for "Dimitris N. Metaxas" in Tables 4.4 and 4.5. For "Judea Pearl", both edit distance and topic edit distance return comparable authors in similar

research topics. Recall that the topic distribution of each author is learned mostly from their paper titles. We manually validated many examples, and compared the results by simply utilizing the topic similarity between authors and by our approach. We found that sequence matching combining topics from title and venue scores did a better job in finding authors in similar research area. Taking "Richard M. Karp" for example, we find that 16 of the 20 comparable researchers returned also have entries in Wikipedia, a crude indication that they are similarly notable. Future work may more systematically examine the performance of clustering similar authors by our distance metric.

There are a few notable bad examples: the comparable researchers for "Donald E. Knuth" are only loosely related. Knuth's paper titles are often short, and commonly use generic computer science terms like "Algorithm". Hence, topic inference on his papers has poor performance, and the comparable authors are mainly determined on venue score sequence matching. As our data contains only $30,000$ authors, many are missing (along with their papers), limiting the set of potential comparable authors.

4.4.3 On Prefix Matching

Each year, many junior researchers begin their career. It is useful and interesting to matching junior researchers to segments of senior researchers. With simple modification, our algorithm can be used to compare a junior researcher to senior researchers in their early career stage. This can be useful, for example, to committees considering the future prospects of job candidates, and to junior researchers finding out whose career trajectory they are following.

Formally, we are interested in the problem that, given a senior researcher and a junior researcher characterized by S and R respectively. Instead of matching full sequence of S and R, we want to match R to every prefix of S. A prefix of the sequence S with length n is denoted by $S[1:k]$ where $1 \leq k \leq n$. The final distance is then the minimal of matching distances with every prefix. If we store all intermediate steps of the dynamic programming table, we can easily compute the distance of prefix matching. Specifically, the vector $D(:, m)$ stores the minimal distance from every prefix of S to R, where m is the length of R. Consequently the minimal distance is no longer $D(n, m)$

but $\min D(:, m)$.

We sample authors with fewer than 100 papers within less than 20 years of research period to test the prefix matching. Comparing to matching full sequence, there are more senior researchers in the results by prefix matching.

4.4.4 On Parallel Computation

Our trajectory matching algorithm computes the matching distance between each pair of researchers through their full publication records. Let N be the number of authors, and L be the average length of trajectories. The computational complexity is $O(N^2L^2)$. This is quite expensive in terms of computational time particularly when N is big. Fortunately, the computation can be done in parallel by Map-Reduce(MR) framework [25].

The MR version of trajectory matching algorithm is as follows. It requires two map reduce phases to finish the job.

Algorithm 4 Map Reduce Version of Trajectory Matching

Phase 1: input document contains authors and their trajectory vector $[a_i, T_i]$
map(String key, String value): // key: document name, value: document contents
for each pair $\langle a_i, a_j \rangle$ in value: **do**
  Emit($\langle a_i, a_j \rangle, \langle T_i, T_j \rangle$);
end for
reduce(String key, Iterator values): //key: $\langle a_i, a_j \rangle$, values: $\langle T_i, T_j \rangle$
for $\langle T_i, T_j \rangle$ in values: **do**
  dist = $D(T_i, T_j)$;
  Emit($a_i, \langle a_j : dist(a_i, a_j) \rangle$);
end for
Phase 2: input is the output of reducer in previous phase.
map(String key, Text value): //Identity mapper
for v in value: **do**
  Emit(key, v);
end for
reduce(String key, Iterator values): //key: a_i, values: $[a_j : dist(a_i, a_j)]$
results = [];
for v in values: **do**
  results.append(v);
end for
sort(results) // by distance score.
Emit(key, results)

Note that in phase 1, the mapper generates N^2 pairs which needs many reducers

and costs lots of network communication time. To further improve it and reduce the number of author pairs emitted, one can put authors into groups. More specifically, we divide N authors into K groups. We emit $\langle a_i, g_k \rangle$ as key, and $\langle T_i, [a_j : T_j]_{j \in g_k} \rangle$ as value in the mapper of the first phase. In the following reducer, we compute the distance between a_i and all $a_j \in g_k$, emit a_i as key and $[a_j : dist(a_i, a_j)]$ as value. In the second phase, we take the output of the reducer of the first phase as the input and go through an identity mapper, then reduce by the key a_i and merge distance pairs from different groups. By doing such grouping, we can sharply reduce the number of reducers needed and therefore lower the cost of network communication.

4.5 Conclusions and Future Work

In this chapter, we address the novel problem of automatically finding comparable researchers through large scholarly data. Unlike existing work, which evaluates researchers mainly by citation counts, our methods consider the sequence of the quality of publishing venues, which seems more appropriate for evaluating and comparing research output. To allow automatic identification of comparable people in similar research areas, we further propose a distance metric which combines the topic similarity and venue quality. Our approach can be easily modified to match junior researches to senior researcher at their beginning of research periods. We analyze the computation time and provide a map-reduce version of the algorithm, which is further improved by grouping authors.

Our analysis and experiment was conducted on large-scale scholarly datasets available on the web. The effectiveness of our methods are demonstrated by arbitrarily picked examples. There are several problems open for future study.

- Data Collection: Lack of data may lead to less accurate results. Many challenges exist in the data collection, e.g. reducing the language gap, knowledge extraction from multiple data sources with different formats.
- Evaluation: There is currently no "ground truth" for our methods. We are developing a user interface to allow exploration of comparable people, and collect user

feedback on results.

- Comparable Network: With comparable relation established, we can define a comparable network, in which each node is a researcher, and edges connect comparable nodes. The weight on the edge is related to the distance between two nodes. It may be interesting to examine the structure of such network, and compare it with co-authorship and citation networks.

Table 4.5. Case Study of Topic Edit Distance

Researcher	Top 20 Comparable Researchers
Richard M. Karp	David R. Karger, Ravi Kumar, Jeffrey D. Ullman, Avrim Blum, Joseph Naor, Frank Thomson Leighton, Rajeev Motwani, Hari Balakrishnan, Eric Horvitz, Mostafa H. Ammar, Rina Dechter, Prabhakar Raghavan, Craig Boutilier, Rafail Ostrovsky, Raghu Ramakrishnan, Yossi Azar, James F. Kurose, Josep Torrellas, Rakesh Agrawal, Andrew Y. Ng
Judea Pearl	Craig Boutilier, Satinder P. Singh, Avrim Blum, Manfred K. Warmuth, Michael J. Kearns, Piotr Indyk, Eyal Kushilevitz, Surajit Chaudhuri, Yoram Singer, Robert E. Schapire, Jon M. Kleinberg, Shafi Goldwasser, Robert Endre Tarjan, Geoffrey E. Hinton, Eric Horvitz, Milind Tambe, Jeffrey S. Rosenschein, Silvio Micali, Daniel S. Weld, Nick Koudas
Hari Balakrishnan	Ion Stoica, James F. Kurose, Baochun Li, Gustavo Alonso, Mostafa H. Ammar, Eitan Altman, Robert Endre Tarjan, Surajit Chaudhuri, Jon M. Kleinberg, Ness B. Shroff, Yossi Azar, Eli Upfal, Peter Steenkiste, Joseph Naor, Sang Hyuk Son, Qian Zhang, Frank Thomson Leighton, Randy H. Katz, Hagit Attiya, Wang-Chien Lee
Nancy Lynch	Baruch Awerbuch, Scott Shenker, J. J. Garcia-Luna-Aceves, Sajal K. Das, Roger Wattenhofer, Moni Naor, Hossam S. Hassanein, Rachid Guerraoui, Amr El Abbadi, David E. Culler, Yishay Mansour, Christos H. Papadimitriou, Klara Nahrstedt, Danny Dolev, Christos Faloutsos, Deborah Estrin, Mostafa H. Ammar, Mario Gerla, Lionel M. Ni, Serge Abiteboul
Dimitris N. Metaxas	Andrew Blake, Trevor Darrell, Jitendra Malik, Jean Ponce, Narendra Ahuja, Shaogang Gong, Dale Schuurmans, Stefano Soatto, Alan L. Yuille, Pascal Fua, Aly A. Farag, Pedro Domingos, Shree K. Nayar, Xilin Chen, Chris H. Q. Ding, Brendan J. Frey, Pietro Perona, Santosh Vempala, Thomas G. Dietterich, Nassir Navab
Dan Boneh	Amit Sahai, Ueli M. Maurer, Jacques Stern, Ronald L. Rivest, Ran Canetti, Shafi Goldwasser, Stuart J. Russell, Abraham Silberschatz, Matthew Andrews, George Varghese, Russell Impagliazzo, Cynthia Dwork, David Heckerman, Hector J. Levesque, Eyal Kushilevitz, Michael J. Kearns, Robert E. Schapire, Joe Kilian, Anoop Gupta, Tatsuaki Okamoto
Donald E. Knuth	Mark Roberts, H. Ramesh, Fady Alajaji, Hemant Kanakia, Mary E. S. Loomis, Michael D. Grossberg, Antonio Piccolboni, Paul Milgram, Andy Boucher, Paul Hart, Kazuo Sumita, Nicholas Carriero, Shiro Ikeda, James M. Stichnoth, Michael Bedford Taylor, Kimiko Ryokai, Riccardo Melen, Jatin Chhugani, Dianne P. O'Leary, Lal George

Chapter 5

Summary and Future Directions

In this thesis, we analyze and model groups in real world problems. The first problem we look at is recommendation targeting groups rather than individuals. We address two challenges: that group information is insufficient and that the presence of members is dynamic, and propose a group multi-armed bandit framework to generate recommendations in sequence. We show for the first time that our group variations of MAB have logarithmic regret bounds, and are efficient in terms of computation time and memory space requirements. We also design a demo system that allows a group to organize events and reach consensus on which recommended item to accept. We conducted a user study through our demo system, and observe the existence of member's influence on decision making. An influence cascade model is proposed to predict individual's decision. Results show that this influence model allows to improve the prediction of individuals' decisions, which can be used to make better recommendations.

The second problem we focus on is personalized search. We address the challenge that each day users submit a large portion of new search queries that do not have sufficient associated historical information in the system to allow good personalized results. We propose a cohort modeling framework to predict individual's search interest and intent based on cohorts' behavior. We examine three pre-defined cohorts, i.e. location, domain and topic, and evaluate their effectiveness on enhancing search personalization by applying cohort models both in isolation and in combination. We also analyze the performance of our cohort models in a number of additional search scenarios (e.g., ambiguous or unseen queries), and show strong relevance gains. Finally we propose a method to learn cohorts via clustering, removing the requirement that we use predefined groups. We show that by doing this we can further enhance personalization over

using our pre-defined cohort modeling method.

The third problem is identifying which groups a given researcher better compares against. We address the challenge that researchers often change research topics and have varied productivity and impact over time, and propose a trajectory matching framework to compare any pair of researchers. Each trajectory is defined in three dimensions: publication, venue and time. Our algorithm can be easily modified to do prefix matching to, for example, compare a junior researcher and a senior researcher. We analyzed and experimented on large scholarly datasets available on the web, arXiv and DBLP, and demonstrate the effectiveness of our method through examples. We also discuss many open problems to study in the future.

This research on solving problems for groups using data has many natural extensions and open problems. Regarding group recommendation, our model assumes arms (items) are independent. One research problem to consider is how to model the scenario where items are correlated. Beyond groups as context, location and time should also be considered, due to the sharp increase in the use of mobile apps. As to cohort modeling, it is interesting to examine whether it can be generalized and applied to applications such as online ads. Regarding grouping comparable researchers, it is still a challenge to design a time efficient algorithm to process large datasets.

The online world is a rich testbed for our research, particularly web media and social networking sites. It is interesting to discover the different and evolving patterns of groups, as well as the uniqueness, trustworthiness, influence and the hierarchy of each group. Another set of questions is to consider richer types of groups and try to automatically identify the different meaningful types of group that a user might belong to. In this context, it might be useful to study sets of groups not limited to people but extend to items such as location or semantically correlated words.

References

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 2011.

[2] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, volume 3, page 3, 2008.

[3] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006.

[4] Rodrigo B Almeida and Virgilio AF Almeida. A community-aware search engine. In *Proceedings of the 13th international conference on World Wide Web*, pages 413–421. ACM, 2004.

[5] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

[7] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, March 2003.

[8] Peter Auer, Nicole Cesa-Bianchi, Paul Fischer, and Lehrstuhl Informatik. Finite-time analysis of the multi-armed bandit problem. In *Machine Learning*, 2002.

[9] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE, 1995.

[10] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The non-stochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[11] Peter Bartlett and Shai Ben-David. Hardness results for neural network approximation problems. In *Computational Learning Theory*, pages 50–62. Springer, 1999.

[12] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[13] Paul N Bennett, Filip Radlinski, Ryen W White, and Emine Yilmaz. Inferring and using location metadata to personalize web search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 135–144. ACM, 2011.

[14] Paul N Bennett, Krysta Svore, and Susan T Dumais. Classification-enhanced ranking. In *Proceedings of the 19th international conference on World wide web*, pages 111–120. ACM, 2010.

[15] 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.

[16] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E Schapire. Contextual bandit algorithms with supervised learning guarantees. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[17] Christopher JC Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *NIPS*, volume 6, pages 193–200, 2006.

[18] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24, 2011.

[19] Haibin Cheng and Erick Cantú-Paz. Personalized click prediction in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 351–360. ACM, 2010.

[20] Wei Chu, Lihong Li, Lev Reyzin, and Robert E Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[21] Mark O' Conner, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: A recommender system for groups of users. In *Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work*, 2001.

[22] Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[23] Varsha Dani, Thomas Hayes, and Sham M Kakade. The price of bandit information for online optimization. *Advances in Neural Information Processing Systems*, 20:345–352, 2008.

[24] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 355–366, 2008.

[25] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI?04: PROCEEDINGS OF THE 6TH CONFERENCE*

ON SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION. USENIX Association, 2004.

[26] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *ICML*, 2011.

[27] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *UAI*, 2011.

[28] Leo Egghe. An improvement of the h-index: The g-index. *ISSI newsletter*, 2(1), 2006.

[29] Peter C. Fishburn. *The Theory of Social Choice*. Princeton University Press, 1973.

[30] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS)*, 23(2):147–168, 2005.

[31] Eugene Garfield et al. Citation analysis as a tool in journal evaluation. American Association for the Advancement of Science, 1972.

[32] GroupLens. http://www.grouplens.org/node/73.

[33] Elad Hazan and Satyen Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[34] Jorge E Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences of the United States of America*, 102(46), 2005.

[35] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *Advances in neural information processing systems*, 2010.

[36] Samuel Ieong, Nina Mishra, Eldar Sadikov, and Li Zhang. Domain bias in web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 413–422. ACM, 2012.

[37] W. Niu J. Kay. Adapting information delivery to groups of people. In *the First International Workshop on New Technologies for Personalized Information Access at the 10th International Conference on User Modeling*, 2005.

[38] 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.

[39] David S. Johnson and Franco P Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93–107, 1978.

[40] Anagnost T. Joseph F. McCarthy. Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In *Computer-Supported Cooperative Work*, 1998.

[41] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pages 440–447. ACM, 2008.

[42] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[43] Lorcan Coyle Kevin McCarthy, Maria Salame, Lorraine McGinty, Barry Smyth, and Paddy Nixon. Cats: A synchronous approach to collaborative group recommendation. In *The Nineteenth International Florida Artificial Intelligence Research Society Conference*, 2006.

[44] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*, 20:1096–1103, 2007.

[45] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, 2007.

[46] Y-J. Lee. Vizsearch: A collaborative web searching environment. *Computers and Education*, 2005.

[47] L. Li, W. Chu, J. Langford, and R.E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[48] John Langford Lihong Li, Wei Chu and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[49] Giovanna Petrone Liliana Ardissono, Anna Goy, Marino Segnan, and Pietro Torasso. Intrigue: Personalized recommendation of tourist attractions for desktop and handset devices. In *Applied Artificial Intelligence*, pages 687–714, 2003.

[50] Francesco Ricci Linas Baltrunas, Tadas Makcinskas. Group recommendations with rank aggregation and collaborative filtering. In *Recsys*, 2010.

[51] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[52] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers, 2004.

[53] Joseph F. McCarthy. Pocket restaurant finder: A situated recommender system for groups. In *the Workshop on Mobile Ad-Hoc Communication at CHI*, 2002.

[54] Kevin McCarthy, Maria Salamo, Lorcan Coyle, Lorraine McGinty, Barry Smyth, and Paddy Nixon. Group recommender systems: a critiquing based approach. In *IUI*, 2006.

[55] Lokman I Meho. The rise and rise of citation analysis. *Physics World*, 2006.

[56] Qiaozhu Mei and Kenneth Church. Entropy of search logs: how hard is search? with personalization? with backoff? In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 45–54. ACM, 2008.

[57] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266. ACM, 2003.

[58] MoviePilot. http://moviepilot.com/.

[59] Seth A. Myers and Jure Leskovec. On the convexity of latent social network inference. In *NIPS*, 2010.

[60] George Popescu and Pearl Pu. What's the best music you have? designing music recommendation for group enjoyment in groupfun. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, 2012.

[61] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[62] Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

[63] P. Rusmevichientong and J.N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2), 2010.

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

[65] Si Shen, Botao Hu, Weizhu Chen, and Qiang Yang. Personalized click model through collaborative filtering. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 323–332. ACM, 2012.

[66] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831. ACM, 2005.

[67] Jill Freyne Shlomo Berkovsky. Group-based recipe recommendations: analysis of data aggregation strategies. In *Recsys*, 2010.

[68] Ashish Chawla Sihem AmerYahia, Senjuti Basu Roy, Gautam Das, and Cong Yu. Group recommendation: Semantics and efficiency. In *VLDB*, 2009.

[69] Barry Smyth. A community-based approach to personalizing web search. *Computer*, 40(8):42–50, 2007.

[70] 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.

[71] 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.

[72] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: a novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 382–390. ACM, 2005.

[73] Sutton and Barto. *Reinforcement learning, and introduction.* Cambdrige, MIT Press, 1998.

[74] 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.

[75] Bilyana Taneva, Tao Cheng, Kaushik Chakrabarti, and Yeye He. Mining acronym expansions and their meanings using query click log. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1261–1272. International World Wide Web Conferences Steering Committee, 2013.

[76] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[77] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158. ACM, 2007.

[78] 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.

[79] 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.

[80] Jaime Teevan, Meredith Ringel Morris, and Steve Bush. Discovering and using groups to improve personalized search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 15–24. ACM, 2009.

[81] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Chapter 5 of: Compressed Sensing, Theory and Applications*, 2012.

[82] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1), 1974.

[83] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[84] Ingmar Weber and Carlos Castillo. The demographics of web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 523–530. ACM, 2010.

[85] Ryen White and Georg Buscher. Characterizing local interests and local knowledge. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 1607–1610. ACM, 2012.

[86] Ryen W White, Peter Bailey, and Liwei Chen. Predicting user interests from contextual information. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM, 2009.

[87] Ryen W White, Paul N Bennett, and Susan T Dumais. Predicting short-term interests using activity-based search context. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1009–1018. ACM, 2010.

[88] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1411–1420. International World Wide Web Conferences Steering Committee, 2013.

[89] Ryen W White, Susan T Dumais, and Jaime Teevan. Characterizing the influence of domain expertise on web search behavior. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 132–141. ACM, 2009.

[90] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. Ranking, boosting, and model adaptation. *Tecnical Report, MSR-TR-2008-109*, 2008.