MACHINE LEARNING FROM HUMAN PREFERENCES AND CHOICES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Karthik Raman

August 2015

MACHINE LEARNING FROM HUMAN PREFERENCES AND CHOICES

Karthik Raman, Ph.D.

Cornell University 2015

This dissertation focuses on developing new machine learning models and algorithms for the task of learning from data that originates from human-system interactions *i.e.,* the **interactive learning** paradigm. A wide array of modern technologies involve significant interaction between the humans users and the system. These technologies – which range from everyday applications such as search engines and retail services, to more disruptive ones such as self-driving cars and smart homes – can greatly benefit from the world knowledge implicit in these human interactions. However, as a consequence of interactive learning data being derived from observed human behavior, standard machine learning models are a poor fit.

This thesis develops a fundamentally new approach to interactive learning. The guiding principle in this dissertation is to **jointly** design the three key components of interactive learning: the **learning algorithm**, the **user behavioral model** and the **feedback interventions**.

The learning algorithms developed in this thesis strive to learn from preference data in a robust manner. Furthermore, they come with theoretical performance guarantees and are shown to work well in practice.

For sound learning from human interaction data, we need plausible models of user behavior while interacting with these systems. The approaches discussed here explicitly account for the different factors that impact the user decisions, such as their motivations, expertise, skills, needs and decision context.

A unique advantage interactive learning systems possess is the ability to intervene and alter the content presented to users so as to maximize learning. This dissertation covers different examples that illustrate that even small changes can greatly improve learning in these systems.

The potency of this joint design methodology is illustrated using different interactive learning examples including: (a) a scholarly text search engine for `arxiv.org`, that autonomously, robustly, and cost-effectively improve its performance; (b) web search and recommender systems that can model and facilitate complex user tasks; and (c) `peergrading.org`, a peer-grading service which collates grades from all the students in a principled manner.

BIOGRAPHICAL SKETCH

Karthik Raman was born in December 1988 in Mumbai in India. He completed his undergraduate studies in May 2010 from the Indian Institute of Technology, Bombay where he earned a Bachelor in Technology degree in the field of Computer Science and Engineering. In the Fall of 2010 he began his Ph.D studies at Cornell University in the Computer Science department under the tutelage of Professor Thorsten Joachims. He received his Doctor of Philosophy degree in Computer Science in August 2015, following which he joined Google as a Research Scientist.

This document is dedicated to all Cornell graduate students.

ACKNOWLEDGEMENTS

First and foremost, I am immensely grateful to my wonderful advisor Dr. Thorsten Joachims. He has always been there for me, sharing his sage advice on matters both professional and personal. Starting from writing research papers to designing good talks, he has always found time to guide me towards becoming a better researcher. I can honestly say that I have learned an enormous amount under his brilliant tutelage. As I get ready to begin my professional career, I sincerely hope that his supremely positive personality, bubbling passion and seemingly endless energy rub off on me. Thank you Thorsten – without you this would have never been possible!

I have been extremely fortunate to have such brilliant committee members helping me through my PhD. Thank you very much Paul Bennett, Johannes Gehrke and Robert Kleinberg for being such wonderful mentors. You have all been a source of great inspiration for me. Working and collaborating with you has been a wonderful experience for which I am truly grateful.

I also owe a great deal of thanks to my wonderful collaborators who have shared ideas with me and engaged me in stimulating discussions. It has been my privilege to get the opportunity to work with all you, starting from my wonderful Cornell collaborators Pannaga and Adith, to my brilliant Microsoft and Google collaborators Krysta, Susan, Chris, Evgeniy, Jeff, Kevin, Kevyn, Rani and Wei amongst others.

I consider my decision to pick Cornell over other universities, as one of the best decisions I have ever made. Under the tutelage of brilliant professors like Claire, Lillian, Bobby, Jon, Peter and others I have gained holistic insights in the fields of computer science and statistics. I have been highly fortunate to work with the wonderful team at arXiv led by Paul G, which has helped me gain new

research insights and validate my research ideas.

I would also like to thank all my friends here at Cornell, for their wonderful company throughout my stay at Ithaca. I am especially thankful for my great friends in the Cornell Machine Learning and NLP Discussion Groups, who have kept me on my toes and given me an opportunity to discuss my nascent ideas and provided me honest feedback. I can sincerely say that I'm a wiser man after all the spirited discussions we have had over the years. I'd like to add a special shout-out to Adith, Amit, Anshumali, Chenhao, Hema, Jon, Ruben and Vikram who helped me in numerous data annotation tasks and provided me detailed feedback on paper drafts, conference presentations and research statements. I'd also like to thank Nikos and Ainur who took me under their wing and showed me the ropes around Cornell when I first got here.

I would also like to acknowledge all the institutions that helped support my work over these past five years and enabled my research. I would specifically like to thank Google, who graciously supported me via a PhD Fellowship; Yahoo!, who helped get me started with my dissertation via a Key Scientific Challenge Award; the Cornell-Technion Research Fund and NSF who supported me via their grant awards IIS-1247696, IIS-1217686, IIS-1142251, IIS-1012593, IIS-0911036, IIS-0905467, IIS-0812091 and IIS-0713483.

Finally, none of this would have been possible without the love, support and encouragement of my family, who have stood by me for the past five years through thick and thin and kept me motivated. Words cannot do justice for the gratitude I owe to my parents Raman and Bhuvaneshwari, grandparents and my sister Aditi back home in India. Thank you very very much for everything! I also am deeply in debt of my uncle Arun and aunt Subha, who have guided and supported me and helped me re-energize myself over the numerous holidays I

have spent with them at their place.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Overview and Preliminaries

CHAPTER 1

INTRODUCTION

Intelligent user-facing systems have become part and parcel of our everyday life. These technologies – which range from internet search engines and online retailers to disruptive applications such as personal robots, online education platforms and self-driving cars – live in a symbiotic relationship with their users - or at least they should. On the one hand, the human users greatly benefit from the services provided by the systems as they assist them in their everyday life. On the other hand, these systems can greatly benefit from the world knowledge that users communicate implicitly through their interactions with these systems. These rich user interactions – queries, clicks, purchases,reviews, answers, demonstrations, etc. – are one of the key sources of "Big Data" which machine learning methods can leverage to greatly impact these systems in multiple ways.

First, these interactions providing enormous potential for economically and autonomously optimizing these systems. Second, this user data carries rich information about the personal preferences of individual users, which can be utilized to improve the user experience via personalization. Third, these interactions carry unprecedented amounts of world knowledge that can be used to solve some of the hardest AI problems.

However, a key challenge in learning from this interaction data is that it does not typically fit standard machine learning models. More specifically, user interactions usually cannot directly be used as a substitute for ground truth labels, as they are simply the observed result of complex decisions made by humans. Furthermore, conventional learning techniques are unable to exploit the additional

experimentation and learning made possible dynamically via the interactive nature of these systems.

This dissertation aims to provide solutions to this problem of learning from human interaction data by introducing new machine learning models and algorithms. The fundamental insight guiding this work is a new approach to the interactive learning problem. Instead of solely designing just the learning algorithm (as done in conventional machine learning), we will jointly design the **three** key interactive learning components (the learning **triple**):

1. **Learning algorithm:** We need algorithms that can learn robustly from observed preferences.
2. **User behavioral model:** To extract the user's true preferences from their interaction behavior we need to explicitly account for the different factors that impact the user decisions – such as their motivations, expertise, skills, needs and decision context.
3. **Feedback intervention:** A previously underutilized benefit of interactive systems is the ability to dynamically intervene and make small changes in what is presented to the user. Such changes can help ensure feedback that is more conducive for learning as we illustrate in this dissertation.

This joint design approach has been utilized in this thesis to develop algorithms that come with corresponding theoretical performance guarantees on the learning quality and are practically viable. These ideas have also been implemented and fielded in human-interactive learning systems – such as the text search engine for `arxiv.org` – which autonomously, robustly, and cost-effectively improve their performance.

1.1 Revealed Preferences: Learning from User Interactions by Introducing Interventions

Traditional machine learning algorithms for information systems have relied on expert annotated data (*e.g.,* assessors are paid to rate search results on a Likert scale). However a more economical source of data is the *implicit feedback* that users provide through their interactions (*e.g.,* clicks). The advantages of using such feedback data are clear: this feedback is not only available in abundance, but also directly indicates the users' – not the experts' – preferences. Consider, for example web search, where such feedback is readily available as users scan the results page and click on different search results, providing the system with information about the goal-directed choices users make.

To learn from this weak feedback, Part II of this dissertation discusses **Coactive learning** algorithms [140, 141] that explicitly incorporate models of how boundedly rational users make decisions. Coactive learning is an online model of interaction between a learning system and human user, where the goal is to maximum user satisfaction. At each step, the system (*e.g.,* search engine) receives a context (*e.g.,* query) from the user. The system then predicts an object (*e.g.,* ranking) and presents it to the user. In response, the user's interaction with the system (*e.g.,* via clicks) results in feedback about the presented object. This feedback, however, does not reveal what would have been the optimal object to present, but only provides an incremental improvement to the presented object. For example, clicks on the search results B and D for the ranking $[A, B, C, D, ...]$, can help us infer that the user would have preferred the ranking $[B, A, D, C, ...]$, but not that it is the best possible ranking. More generally and in contrast to

standard machine learning where optimal feedback is required, coactive learning merely requires feedback that slightly improves on the presented object. Furthermore, this user feedback can contain noise and be biased by factors such as the presentation order.

To learn from this weaker form of preference feedback, which is readily available from user interactions, this dissertation presents new techniques developed using the triple-based joint design for interactive learning. Chapter 3 illustrates the importance of incorporating feedback interventions, into the learning system design. The resulting approaches, combine principled learning algorithms with plausible models of user interaction and appropriate feedback interventions. The interplay between these learning systems and boundedly rational users leads to autonomous learning on-the-fly along with strong theoretical guarantees. In particular, this thesis proves that the new algorithms converge towards the optimal solution at a rate that is proportional to the square root of the number of learning steps and independent of the dimensionality of the feature space. This result turns out to be particularly interesting given that these convergence rates for learning from noisy, biased preferences, are asymptotically equivalent to the rates achieved by the best algorithms when the optimal object is provided as feedback.

In addition to these guarantees, this thesis also provides empirical evidence that establishes that well-designed coactive learning systems perform robustly and accurately in real-world settings. In particular, results of studies with live users [140] conducted on the experimental text search engine at `arxiv.org` (a scientific repository for e-prints) are discussed. These studies demonstrate that these algorithms can learn successfully from the interactions with users, opti-

mizing retrieval performance quickly and operating completely autonomously without requiring any maintenance from the system designers.

1.2 Complex Utilities: Learning Diversified Recommendations

Another key component of the interactive learning triple is the user model. Previous approaches to these learning problems have made simplifying assumptions on the user model for the sake of learning. For instance, most search and recommendation algorithms model the relevance[1] of different items to be independent of other items in the ranking/recommendation list. However, these simplifying assumptions rarely hold in practice. Part III aims to tackle this issue by studying the use of principled learning algorithms in conjunction with sophisticated user models that capture realistic behavior observed in users performing complex informational tasks.

Diversified retrieval is one such example of a complex informational task, where the goal is to provide a comprehensive set of results that are distinct and cover the different needs of the users. Previous approaches to this problem have had to make hand-coded choices between two factors: partially satisfying all the different needs *vs.* specifically catering to the most common need. This dissertation introduces methods that learn the right balance between these two extremes by explicitly modeling the joint utility of a collection of items using submodular functions.

Intrinsic Diversity is one of the problems for which we apply these joint modeling techniques. In intrinsic diversification, the goal is to cover different aspects

[1] a commonly used proxy for user utility in search and recommendation problems

of the information need of a *single* user. For example, a user of a personalized news system would not like to see exclusively articles about the Greek Financial Crisis on any given day, even if this was the topic she was most interested in. Instead, a diversified portfolio of news articles that covers all the interests of the user would maximize the user's overall utility.

Intrinsic Diversity is particularly prominent in web search, as a large fraction of real-world search tasks are intrinsically diverse [19]. However, since current research on web search has focused solely on optimizing and evaluating single queries, these complex tasks currently require significant user effort via multiple interactions with the search engines. An ideal search engine would not only retrieve relevant results for a user's particular query, but also be able to identify when the user is engaged in a more complex task and aid the user in completing that task – whole-task relevance. Towards this goal, Chapter 4 details the first study of Intrinsic Diversity in the context of web search and provides algorithms that optimize for whole-task relevance [134, 135]. In particular it addresses three key problems for Intrinsic Diversity (ID) retrieval: identifying authentic instances of ID tasks from post-hoc analysis of behavioral signals in search logs; learning to identify queries that mark the start of an ID search task; and given an ID query, improving the search experience by predicting which content to prefetch and rank using a joint model of the document relevances and aspect relevance to the underlying task

Intrinsic Diversity can also be learned interactively on-the-fly as demonstrated in Chapter 5. This thesis provides algorithms that continuously learn both the relevance of items and the appropriate amount of diversity a user desires. These algorithms learn from set-valued preference data derived from the

implicit feedback of user interactions using coactive learning [141]. Theoretical and empirical analysis of these algorithms again reveal convergence rates equivalent to optimal feedback conditions, which has led to these algorithms being deployed for recommending scientific articles.

Another problem that benefits from the joint modeling of the utility of a set of items is *Extrinsic Diversity*. This problem arises when *different* users express different information needs via the same query, thus resulting in ambiguity about the user intent. Diversification can be used here to provide relevant results for all the distinct information needs of the users, thereby preventing the most popular user intent from drowning out all other intents. Chapter 6 introduces coactive learning algorithms that can learn to diversify from user interactions, unlike conventional algorithms that require explicit feedback. While these algorithms [136] use submodular models similar to those in the case of intrinsic diversity, here we are optimizing the ranking for a distribution of utility functions (as opposed to just a single utility function). In addition to strong theoretical bounds, these algorithms also display significantly faster convergence than existing algorithms for single-query diversification. Furthermore, the algorithms introduced here are the first known algorithms for the task of cross-query diversification from implicit feedback.

While these extrinsic diversification methods mitigate the problem of completely missing the user's intent, they are necessarily a compromise between the breadth and the depth of coverage of each user intent. To overcome the constraints of this compromise, Chapter 7 introduces a new dynamic retrieval model that is not restricted to a single ranking [139]. The proposed model replaces the single one-size-fits-all ranking with a two-level ranking, where the

second order rankings are conditioned on the user's interactions with the top-level ranking. Constructing these two-level rankings requires modeling diversity (in the top-level ranking) and relevance (in the second-level rankings), which can be accomplished using a Structural SVM-based learning algorithm.

1.3 Stated Preferences: Scaling up Student Evaluation

Modeling how humans make decisions is essential not only for understanding the preferences they reveal implicitly through their actions, but also for understanding the preferences they state explicitly. Part IV studies as an example of learning from preferences, the problem of peer grading, where students grade each other's work. Peer grading is a promising approach for tackling the problem of student evaluation at scale, since the number of graders scales with the number of students. However, students are not trained graders, which motivates grading models that are more robust than asking students to assign letter grades. To this effect, this dissertation investigates the eliciting of *ordinal* feedback (where the emphasis is on ordering different alternatives) from students as feedback [137]. Under this feedback model, student graders make ordinal statements (*e.g., project X is better than project Y*) as opposed to cardinal statements (*e.g., project X is a B-*). The use of such ordinal feedback is preferable in such a scenario as it is easier to provide and more reliable than cardinal feedback. This dissertation covers different algorithms to aggregate this ordinal feedback from individual graders to infer an overall grade for each assignment. The proposed algorithms not only model the quality of the assignments, but also the reliability of the different graders, since graders may have differing skills and grading expertise. To demonstrate the applicability of these methods, results of

a user study conducted in a real university class are provided and discussed. In particular, these results demonstrate the proposed techniques to be a viable alternate to traditional evaluation techniques (instructor/TA grading). The study also surveyed students to find that the overall peer-grading process was found to be a helpful and valuable experience indicating the grading process to have educational value as well.

To further increase adoption of these techniques, Bayesian tools can be used to extend the aforementioned approaches to this (ordinal) peer grading problem [138]. By computing the Bayesian posterior of the ranking distribution, course instructors receive more information about the uncertainty of each assignment's aggregated grade. This information can be used for better interpreting the resulting grades or for assigning additional graders to assignments with high posterior entropy. The resulting techniques, which were deployed for use at `peergrading.org`, have found use in multiple courses as well as in conference peer reviewing.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we will first understand the interactive learning problem. We will then understand other closely related learning problems. We will also delve into work done on understanding user behavior in these interactive learning systems. Lastly we will conclude with a summary of work on the most popular well-studied learning applications: search and recommendation.

2.1 Interactive Learning

An interactive learning system is one which is constant *constantly* interacting and learning from its' user(s). Unlike traditional machine learning, the outputs of an interactive learning system are typically objects that the users can interact with (typically in a non-trivial manner) to find the information they seek and complete the task they had in mind. This user interaction behavior also serves the dual purpose of providing rich feedback to the system from which it can learn, so as to improve its' overall performance and efficacy.

This interplay between the interactive learning and the user is illustrated in Figure 2.1 using the example of a search engine. Every time a search user issues a query the search engine returns a complex object (in this case a ranking of the search results), which the user can interact with. This user interaction in turn provides meaningful feedback (say via what search results were clicked) for the system to learn from and get better overall.

While prior work has mostly focused on developing improved learning algorithms for these systems, this dissertation introduces a new way of designing

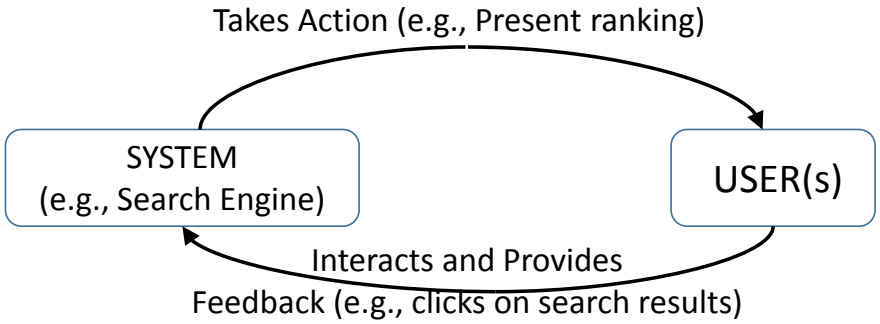

Figure 2.1: Illustration of the interplay between an interactive learning system and the users with a search engine as an example.

these systems by jointly consider the user behavioral model and the feedback interventions along with the learning algorithm. This joint design can lead to principled learning systems that capture the best attributes of system and user (while overcoming the shortcomings of the other), namely the system's strong computational capabilities and the rich world knowledge available to users.

2.2 Related Learning Paradigms

Supervised learning (or *passive* learning) is the classical learning paradigm where given a dataset of expert-labeled examples, the goal of the learning algorithm was to predict the output as accurately as possible. However these classical algorithms are limited to being able to predict simple objects such as a binary label (classification [81]), a number – either unconstrained (regression [122]) or bounded (ordinal regression [54]).

Structured learning (or *structured prediction*) [84, 159, 162] on the other hand, allows for learning systems to be able to predict more complex structures such as lists, trees and arrays. These methods [83, 158] enabled the use of machine

learning for a new set of problems, particularly in the Information Retrieval and Natural Language Processing domains. However these approaches still rely on the availability of a large amount of detailed expert-labeled data.

To reduce this data dependence, *Active Learning* approaches were introduced [49, 55, 90, 151]. Unlike *passive* supervised learning methods, active learning techniques identify which data points to label and then proceed to learn using them. The goal of active learning techniques is to not only learn a good model, but also use as few labeled examples as possible. These approaches primarily work by identifying data points with the most label uncertainty. While these approaches are able to reduce the data dependence, they still are limited by the requirement of *gold-standard* labeled data.

A more closely related set of learning problems are the *bandit learning problems*. The *Multi-Armed Bandit* (MAB) problem in particular is a well-studied problem, that has become immensely popular in sequential decision-making applications. In this problem at each instance, the learner has to make a choice between one of k arms (*i.e.*, options). Based on this choice, the learner receives some reward which is disclosed to them. To tackle this problem – and its' numerous variants – many different approaches have been employed. One set of approaches build on the seminal work in *Optimal Learning* by Gittins and colleagues [67], where the MAB problem was decomposed into an array of One-Armed Bandit problems. An alternate to some of these computationally expensive approaches, are techniques such as the Upper Confidence Bound [12]; Knowledge-Gradient based methods [62] which essentially perform a one-step lookahead to determine the best arm; as well as other simpler heuristics such as Boltzmann-exploration [37]. More recently Bayesian approaches to this prob-

lem have been employed, resulting in techniques such as Bayes-UCB [91] and Thompson-Sampling based methods [5] (which build upon the seminal work of William Thompson [160]). While there has been some recent study into the issue of structured output spaces [44], all these different learning approaches (which can be clubbed under the *online learning* paradigm [38, 108]) fall short when it comes to predicting complex objects such as lists and trees where is there a lot of inherent structure in the outputs. Furthermore they are not suited for learning from the kind of feedback observed in these interactive learning systems (*i.e.*, observed human decisions).

The *dueling bandits* problem is an exception as is intended for learning from preferences *i.e.*, when the result of a comparison of two alternatives is made available to the learner [174, 177]. However unlike the interactive learning setting where the user's preferred alternative (based on their feedback) may be any alternative in the space of possibilities, the dueling bandit approaches requires comparisons between two pre-selected alternatives.

More generally, while problems like partial monitoring [21, 22], reinforcement learning [89, 156] and inverse reinforcement learning (or apprenticeship learning) [2, 123] can learn from data that originates from human behavior, they all impose strict restrictions on the kind of feedback that can be successfully learned from. Furthermore some of these problems cannot be used for (potentially infinite) structured output spaces, whereas the interactive learning algorithms introduced in this dissertation place no such restrictions on the output space or the user feedback.

Correctable Learning is an alternate paradigm for incorporating humans in the learning loop [142]. Here, the system receives feedback about examples it has

incorrectly learned so that it can look to rectify these *mistakes*. The system uses this feedback to rectify these errors and in the process (hopefully) learn a better model. While such mistakes can be spotted by experts, in interactive learning we want the system to learn from the natural interactions of regular users.

2.3 User Behavior on Interactive Learning Systems

When interacting with these complex learning systems, the users themselves display equally complex behavior. Take for instance a search engine, where users can perform any of the following actions: issue a query, view the search results, click or skip search results, use a query suggestion, rephrase a query and reissue it and many other such actions. Each of these actions in turn are affected by biases the users experience and display.

For example, eye-tracking studies [13, 85, 111] have found that the position of objects can significantly impact (and bias) the feedback signals observed. Joachims and colleagues [85] laid the ground-work for position-discounted models in information retrieval, using eye-tracking studies that demonstrated the *top-to-bottom* viewing patterns of users. In particular, they found that regardless of the relevance (or an equivalent measure of utility) of an item, the higher it was placed in the list/ranking, the more attention (*i.e.*, clicks) it received. This bias, which is termed as the *position bias*, is one such bias that needs to be accounted for when considering user behavior as feedback on these complex objects. Another kind of bias is the *context bias* (or *batch effect*) [18, 56, 82]. This bias causes an item to be viewed more favorably (by users) the worse quality the items around it (in the output object) are. These biases are only further com-

pounded by the advent of newer technologies such as touch screens (which now account for a significant volume of traffic on these systems) [70]. Furthermore, user's inherent beliefs and biases also affect their behavior on these learning systems in addition to biases due to the presentation of the output object [167].

These challenges introduced due to these numerous biases has led to a large body of work developing user models and understanding user behavior on these systems [17, 125]. Econometric models [14, 15] have been also been utilized to help understand and characterize user behavior. These resulting models user behavior, can in turn be used by these systems to improve themselves (say via the joint design approach proposed in this dissertation).

Among interactive learning applications, search and recommendation systems have been among the most well-studied due to their prominence across different domains – such as entertainment, music, retail, dining to name a few. Understanding complex, multi-stage user behavior on these systems, as they issue multiple searches to obtain the necessary information, has become a very active area of research [97, 168]. One such problem of interest is *task-based* retrieval, where tasks are the unit of interest, as opposed to queries or sessions [73, 74, 106]. *Trail-finding* is another related problem that also considers the influence of complex factors (such as relevance, topic coverage, diversity and expertise) on user search behavior in certain contexts [153, 173]. More detailed search interfaces and functionalities, such as those in faceted search [95, 96] and exploratory search [118, 130], add new levels of complexity to user behavior in these systems.

As we will devote significant attention to the search and recommendation problems in this dissertation, the next section will continue discussing them in

more detail.

2.4 Ranking, Recommendation, Retrieval and Search

2.4.1 Learning to Rank

Most classical algorithms for ranking do not involve any machine learning. The Probability Ranking Principle [146] popularized by Stephen Robertson, states that documents should be ranked in order of the probability of relevance or usefulness. This formed the basis of one of the first Information Retrieval (IR) systems, the Okapi BM25 system [145, 147] as well as other subsequent ranking systems [101]. Another popular principle, the Vector Space Model [148], which uses vector representations for the queries and the documents, was the basis for another pioneering IR system: SMART [31].

However, researchers realized that machine learning techniques can be employed to improve over these non-learning baselines (while still utilizing insights from these prior models to design features). The resulting problem of **Learning To Rank** (LTR) is focused on producing a ranking of results given a query, by training models using gold-standard labeled datasets.

LTR approaches fall in one of three categories: point-wise, pair-wise or list-wise. Point-wise approaches model the relevance of individual documents per query. These typically involve the use of traditional machine learning techniques such as regression-based approaches [54, 65, 105].

Pair-wise approaches on the other hand, model the preference relation be-

tween pairs of documents and target ranking relevant documents above irrelevant ones. The techniques that fall under this class are typically adaptations of conventional classification algorithms such as SVMs [76, 82] or Boosting [64, 169], with suitable pair-wise loss functions chosen (based on the specific performance measure being optimized for).

Unlike point-wise and pair-wise approaches, list-wise approaches do not decompose the ranked list and instead use the entire ranking as is, within the learning formulation. These methods tend to be more complex adaptations of classification algorithms, with non-trivial optimization issues needing to be tackled to optimize these highly non-convex loss functions [34, 170].

These learning methods have been shown to perform extremely well on competitions such as the Yahoo Learning to Rank Challenge [40]. In particular, the LambaMART and LambdaRank techniques [32, 33, 169] (along with their adaptations) have found use in many practical systems. Furthermore, these learning techniques have allowed search engines to utilize large datasets and train sophisticated models. For a more exhaustive survey of the field we would refer the interested reader to Tie-Yan Liu's excellent survey report [110]. Learning to rank has also found use in more specific ranking problems, such as the approaches discussed next.

2.4.2 Diversity

Classical research on search and retrieval has focused on optimizing and evaluating single queries, as discussed in the previous section. However, many complex tasks such as vacation planning, comparative shopping, literature sur-

veys, *etc.* require multiple queries to complete the task [19, 87]. Consequently, an increasing fraction of user queries are part of more complex tasks which span multiple queries across one or more search sessions [97, 109].

One of the classical examples of complex user behavior is that of diversified retrieval/search (which is the focus of Part III of this dissertation). Thus, presenting a diverse set of results is an important goal in both web-search ranking as well as recommender systems research. Diversity in search can be of two kinds: extrinsic and intrinsic [132].

The more well-known problem is that of *extrinsic* diversity. This is the case when the intent of the query issued by the user is unclear. For instance the query `jaguar`, where it is unclear if the user is referring to the car manufacturer or the animal (or some other meaning of the phrase). Presenting a diverse set of search results can help alleviate this problem, as it allows the search engine to address different possible intents of the query and thus satisfy most users.

In contrast to extrinsically-oriented approaches, which diversify search results due to ambiguity in user intent, *intrinsic* diversification requires that results are both relevant to a single topical intent as well as diverse across aspects, rather than simply covering additional topical interpretations. In other words, the diversity is required by the user themselves, say for getting a more holistic idea of the topic.

Most research in this field has been focused on extrinsic diversity. Among the methods developed for diversity, most are not based on learning. This includes popular approaches such as MMR [35], Less is More [43], Essential Pages [157] amongst others [48, 178]. More recently supervised learning methods have

been developed for diversity [99, 149, 176] and found to work well. One of the first such methods is the SVM-Div approach proposed by Yue and Joachims [176]. As common among some diversification approaches [133, 157], SVM-Div works by casting the problem as a specific kind of set coverage instance, where the goal is to maximize the (weighted) number of intents covered in the ranking. In particular, it relates diversity in word occurrences to diversity in search intents. This relationship between words and intents is learned using a structural SVM method, where the discriminant function is formulated as a coverage problem with intent coverage serving as the loss function.

Unfortunately, supervised learning methods rely on manually judged training data with multi-topic annotations, which are highly expensive and difficult to obtain. To avoid this problem, Radlinski et al [133] proposed a multi-armed bandit based algorithm to tackle the diversification problem. However this approach learns very slowly in practice and does not couple the arms together either. Recent work [154] has generalized this by coupling the arms together using a metric space. However this approach is still limited by a hard-coded notion of diversity. Furthermore it does not generalize across queries either. Yue and Guestrin [175] proposed online learning algorithms (for the problem of intrinsic diversity). Their method works by maximizing submodular utility functions, and can generalize across queries. However, their model relies on observing cardinal utilities which are far less reliable than the preference feedback that can be easily obtained in interactive learning systems, as shown in user studies [80]. El-Arini and Guestrin [58] also propose submodularity-based techniques to optimize for both diversity and relevance, in the context of scientific literature discovery. However, their model assumes noise-free feedback, which is unrealistic for real users in interactive learning settings.

Dynamic rankings have also been proposed as a means to tackle this problem of diversification. Here, users are presented with rankings that adapt on-the-fly based on the user's interactions [30]. Adding a level of on-the-fly user interaction to ranking has also been found to helpful for structured and faceted search problems [128, 179], such as product search.

2.4.3 Rank Aggregation

While learning to rank involves identifying patterns across multiple rankings of different queries, a related class of problems, broadly termed *Rank Aggregation* [110], involve combining information contained in rankings from multiple sources for a single query. There are different reasons that motivate the use of such rank aggregation including:

- Aggregating rankings from multiple weaker sources to help come up with an overall better ranking. The principles behind this are similar to those behind ensemble machine learning methods, most notably boosting [150].
- Aggregating partial rankings from different sources to come up with a complete ranking. This can be thought of as a *divide-and-conquer* like approach to the ranking problem.
- A risk-minimizing ranking technique when the different sources have distinct areas of expertise.

Rank aggregation has been a topic of research for nearly a century which has led to a number of classical models and techniques such as the seminal work by Thurstone [161], Mallows [116], Bradley & Terry [29], Kemeny [92], Luce [114]

and Plackett [127]. Many rank aggregation methods used today [45, 69, 112] build on these classical techniques.

Search Result Aggregation (also known as **Rank Fusion** or **Metasearch**) is a specific rank aggregation problem where the goal is to merge search result rankings from different sources[1] to produce a single output ranking. Such aggregation has been widely used in both supervised and unsupervised settings, so as to improve over the ranking performance of any single method [10, 24, 52, 68, 124, 129, 164].

Another popular variant of rank aggregation is in **Social Choice** and **Voting Systems**, where preferences from a set of individuals stated over competing items/interests/candidates need to be aggregated. The goal is to identify the most preferred alternatives given conflicting preferences [9]. Commonly used aggregation techniques are the *Borda count* and other Condorcet voting schemes [10, 57, 113].

In addition to these fundamental applications, rank aggregation has also seen recent use in other application domains. These range from problems such as multilabel/multiclass classification (by combining different classifiers) [102] to learning player skills in a gaming environment [77]. Part IV of this dissertation discusses how classical rank aggregation approaches can be extended for the problem of educational assessment at scale via peer grading.

[1] Typically these sources are different search ranking algorithms or systems.

Part II

Using Feedback Interventions To Improve Learning

In this part of the dissertation, we shall illustrate the importance of the joint design approach to interactive learning problem. In particular, we shall see the difference appropriate feedback interventions can make when introduced into the learning system. Using a scholarly text search engine as a case study for interactive learning, we will demonstrate that designing learning algorithms in conjunction with suitable models of user behavior and well-thought feedback interventions, results in stable learning systems that learn constantly from user interactions despite the noisy, bias implicit in them.

The next chapter discusses a learning paradigm called *coactive learning*. Coactive Learning is a model of interaction between a learning system (*e.g.,* search engine) and its human users, wherein the system learns from (typically implicit) user feedback during operational use. As is common in interactive learning systems, user feedback takes the form of preferences. While learning algorithms have been introduced to learn from this weak feedback, these algorithms can be unstable and ineffective in real-world settings where biases and noise in the feedback are significant. The coactive learning algorithms introduced in the next chapter are the first that can learn robustly despite bias and noise. They utilize suitable feedback interventions, where the output objects (*e.g.,* rankings) are slightly perturbed before presenting to the user, so as to stabilize the learning process. In addition to theoretical and empirical results, the efficacy of these algorithms is also demonstrated via a user study on a live search engine.

CHAPTER 3

LEARNING FROM USER PREFERENCES: COACTIVE LEARNING

A growing number of interactive systems use machine learning to adapt their models to different environments, different users, or different user populations. Examples of such systems range from search engines and recommender systems, to personal assistants and autonomous robots. Conventional learning algorithms for these systems have relied on expert annotated data. Instead a more timely and cost-effective source of data is *implicit feedback* from users, which is available in abundance. We would ideally like these system to learn directly from their users in a manner that is unobtrusive, robust, and efficient.

Coactive Learning [152] is a model of learning from such feedback. It works by combining a boundedly rational model of user behavior with an online learning model that formalizes the goal of learning. In particular, Coactive Learning models the interaction between the user and a learner using weaker assumptions about the user feedback than in standard supervised learning. At each step, the learner (*e.g.,* search engine) receives a context (*e.g.,* query) for which it predicts an object (*e.g.,* a ranking, say $[d_1, d_2, d_3, d_4, ...]$). This object is then presented to the user. The system then observes the user's interactions with this object. If this object is suboptimal, the user's interaction may provide the system with a slightly improved object. Note however, that this need not necessarily be the optimal object, as typically assumed in supervised learning. This means the user merely provides a preference, which can typically be inferred from implicit feedback (*e.g.,* clicks on d_2 and d_4 imply that the user would have preferred the ranking $[d_2, d_4, d_1, d_3, ...]$). The learning goal here is to minimize the sub-optimality of the predictions over the life of the learning system.

Perceptron like algorithms have been proposed for the Coactive Learning model [152]. They have been theoretically shown to converge towards optimality in a noise-free and realizable setting. Unfortunately, as we shall demonstrate later in this chapter (via a live user study), even a small amount of noise can make these existing algorithms fail catastrophically.

To overcome this problem, we shall employ the joint design principle to interactive learning. In particular, we introduce feedback interventions, in the form of small perturbations to the predicted output. These interventions, coupled with a linear user utility model and a new learning algorithm – called the *Perturbed Preference Perceptron* – are shown to greatly improve performance, even in an agnostic setting, and increase robustness to noise. The overall approach leads to greatly improved generalization performance both in simulation experiments, as well as live user study on the search engine. Furthermore, this approach also allows us to theoretically characterize the performance of the algorithm, in terms of provable regret bounds, and thus provide explicit guidance for its application in practice, especially for ranking problems in search and recommendation.

Note that while this chapter uses ranking as the primary interactive learning example, the coactive learning model and the algorithms presented here are far more general with applications in machine translation, robotics, etc.

3.1 Related learning models

Here we will cover existing learning models which bear similarities with the coactive learning model proposed in [152] (which we discuss in the next sec-

tion). Feedback in coactive learning lies between the Multi-Armed Bandit problem [11, 12] (payoff only for selected action) and the Expert-Advice problem [38, 180] (payoff for all actions). However, the coactive learner never observes absolute payoffs, but merely a preference between two actions. This aspect of preference feedback is similar to the dueling bandits problem [174, 175]. However, in the dueling bandits model the algorithm chooses both actions, while the user and algorithm chose one action each in the coactive learning model.

Coactive learning also differs from other preference learning problems. For example, in ordinal regression [54] a training example (x, y) provides an absolute rank y. Ranking with pairwise preferences [47, 64, 75] is another popular problem. However, existing approaches to this problem require independent, identically drawn (IID) samples in a batch setting, while coactive learning works with no-IID data in an online setting. List-wise approaches to ranking (see [110]) differ from coactive learning as they require the *optimal* ranking for a query, not just a preference between typically suboptimal rankings. Partial monitoring games [21] also differ from coactive learning, as they require that loss and feedback matrices are revealed to the learning algorithm. Furthermore, partial monitoring games have no explicit notion of context that is available at the beginning of each round. Additional details about some of these models as well as other related learning models is provided in Section 2.2.

One of the key ideas in this chapter is based on perturbing the output of a predictor for improved feedback, to serve as a **feedback intervention**. In information retrieval, this idea has been proposed for at least two purposes. First, search results from two retrieval functions are interleaved [41] to elicit unbiased user preferences. Second, the "FairPairs" perturbation strategy [131] was pro-

posed for debiasing click data in search. We use the FairPairs idea, and provide the first learning algorithm that utilizes this debiasing strategy.

3.2 Coactive Learning model

We detail the coactive learning model in this section, as recently proposed by Shivaswamy and Joachims [152]. Fundamentally coactive learning is an interactive learning model that models the interplay between the (learning) system (*e.g.,* search engine) and its' user(s). At each iteration t of the user-system interactions, the user states a context $\mathbf{x}_t$ (*e.g.,* query). In response to this the learning algorithm makes a prediction $\mathbf{y}_t \in \mathcal{Y}$ (*e.g.,* ranking). The user draws some utility $U(\mathbf{x}, \mathbf{y})$ from this prediction. In the process the user also interacts with this predicted object and thus (potentially implicitly) conveys an improved prediction $\bar{\mathbf{y}}_t \in \mathcal{Y}$ as feedback to the system. Fundamentally, coactive learning utilizes these (weak) preferences over objects as the feedback for learning. It essentially requires that the feedback object $\bar{\mathbf{y}}_t$ is (mostly) an improvement (in terms of user utility) over the object presented to the user $\mathbf{y}_t$:

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) \geq_\alpha U(\mathbf{x}_t, \mathbf{y}_t).$$

The $\geq_\alpha$ notations refers to the *α-informativeness* feedback characterization, which is described formally in Equation 3.2 in the next section.

The goal of a coactive learning algorithm is to minimize regret, where the (average) **regret** of a coactive algorithm after T iterations is defined as:

$$REG_T = \frac{1}{T} \sum_{t=1}^{T} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right). \tag{3.1}$$

where the optimal prediction for iteration t is denoted as $\mathbf{y}_t^* = \texttt{argmax}_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{x}_t, \mathbf{y})$.

As discussed in Chapter 1, one of the three keys of interactive learning (as proposed in this dissertation) is the user behavior model. While users of interactive learning systems can at times display complex behavior (as discussed in Section 2.3), to simplify the learning problem we will use a simple but potent user model. In particular, the rest of this chapter[1] will assume a linear model for the user utility *i.e.,* $U(\mathbf{x}, \mathbf{y}) = \mathbf{w}_*^\top \phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{w}_* \in \mathbf{R}^N$ is an unknown vector. Here, $\phi(\mathbf{x}, \mathbf{y}) \in \mathbf{R}^N$ represents the joint feature vector of context $\mathbf{x}$ and object $\mathbf{y}$. We assume that this vector is bounded, *i.e.,* $\forall \mathbf{x}, \mathbf{y}; \|\phi(\mathbf{x}, \mathbf{y})\|_{\ell_2} \leq R$. Note that true utility U and weight vector $\mathbf{w}_*$ are never revealed to the learning algorithm. We simply assume that users behave as per this utility function (*i.e.,* preferring higher utility objects), and only use it in our evaluation.

3.2.1 Alpha Informativeness: A Feedback characterization

To be able to state any meaningful theoretical results regarding the performance of a coactive learning algorithm, we need to be able to characterize what kind of improvement the feedback object $\bar{\mathbf{y}}_t$ provides over the presented object $\mathbf{y}_t$. Towards this, we shall try to capture the behavior of a *boundedly rational user* using the α-informativeness characterization:

Definition 1 *User feedback is said to be α-informative in expectation if:*

$$\mathbf{E}_{\bar{\mathbf{y}}_t}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t)] \geq U(\mathbf{x}_t, \mathbf{y}_t) + \alpha(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)) - \xi_t. \tag{3.2}$$

In the above definition, the expectation is under $\mathbf{P}_{\mathbf{x}_t}[\bar{\mathbf{y}}_t|\mathbf{y}_t]$ (*i.e.,* uncertainty in user behavior). The definition characterizes by how much the feedback pro-

[1] Part III tackles the issue of more complex user behavior.

Algorithm 1: Preference Perceptron.

Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$

for $t = 1$ **to** T **do**

Observe $\mathbf{x}_t$

Present $\mathbf{y}_t \leftarrow \texttt{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$

Obtain feedback $\bar{\mathbf{y}}_t$

Update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$

vided, in expectation, is an α-factor improvement over the presented object relative to the maximum possible improvement $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$, while allowing for some slack ξ_t. Characterizing the feedback from boundedly rational users through Eq. (3.2) is sensible: a boundedly rational user[2] may be satisfied and not search the full space $\mathcal{Y}$ for the optimal $\mathbf{y}^*$ (captured by α), while also making imperfect assessments of utility (captured by ξ_t). We should note here that α-informativeness is **not an assumption** but simply a characterization. In fact any user behavior can be characterized by appropriate setting of α and ξ_t values.

3.3 The Preference Perceptron

The *Preference Perceptron* [152] is a simple algorithm for coactive learning that is adapted from the traditional perceptron algorithm for supervised learning [27]. It (Algorithm 1) works by maintaining a weight vector $\mathbf{w}_t$, that represents the algorithm's current estimate at iteration t of $\mathbf{w}_*$. To start with, the weight vector is typically initialized to $\mathbf{0}$. At each time step t, the algorithm observes

[2] A boundedly rational user is one who makes rational decisions (*i.e.,* trying to improve their utility) under the information bounds/constraints placed on her/him (say by the interactive learning system's interface or available functionalities).

the context $\mathbf{x}_t$ and presents an object $\mathbf{y}_t$ that maximizes $\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$ over $\mathbf{y} \in \mathcal{Y}$ (*i.e.,* the maximizer of the algorithm's current estimate of the utility function). The algorithm then observes the user feedback $\bar{\mathbf{y}}_t$ and updates the weight vector $\mathbf{w}_t$ in the direction $\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$ (as opposed to the conventional perceptron algorithm which updates using the optimal object $\mathbf{y}_t^*$). **Note** that the α parameter does not appear in these algorithms; it is simply used for the theoretical analysis.

Despite its' apparent simplicity, the preference perceptron has been shown to have tight regret bounds when the user feedback has no noise.

Theorem 2 *(Originally presented in [152]) The expected average regret of the preference perceptron can be upper bounded, for any $\alpha \in (0, 1]$ and any $\mathbf{w}_*$ as*

$$\mathbf{E}[REG_T] \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R\|\mathbf{w}_*\|}{\alpha \sqrt{T}}. \tag{3.3}$$

The above bound is tight in the noise-free case and does not make any assumptions, as any user behavior can be characterized via α informativeness. However, we will show in the next section that this seemingly perfect algorithm can fail catastrophically in noisy environments.

3.4 Case study: A live text search engine

To test how these (coactive) learning algorithms would do in practice, we conducted a user study on a live scholarly text based search engine at `arxiv.org`. *ArXiv* is a repository for e-prints of scientific articles from different domains including physics, astrophysics, statistics and computer science amongst others. With over a million technical articles, effective search engines are critical to help users find the documents of interest to them. Learning in this real-world environment is a challenging task for multiple reasons. First, users typically only

review the first page of search results (at most 10 documents per query) and second, the noise experienced in such a productive system exhibits less regularities. Thus learning a good ranking function from user interactions is as a perfect acid test for an interactive learning algorithm.

We thus implemented the Preference Perceptron algorithm on the full-text search engine of `arxiv.org` with the goal of learning a good document ranking function. Details of the implementation follow. We used a query-document feature vector $\phi(\mathbf{x}, d)$ of length 1000, which included various query-dependent features (*e.g.*, query-title match) and query-independent features (*i.e.*, the age of a document). We constructed feature vectors for rankings $\mathbf{y} \in \mathcal{Y}$ as a weighted sum $\phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \gamma_i \phi(\mathbf{x}, \mathbf{y}^{(i)})$ of feature vectors of documents in the ranking $\phi(\mathbf{x}, d)$, where $\mathbf{y}^{(i)}$ is the i-th document in the ranking. The γ_i are decreasing position discounts, such that sorting by document utility $U(\mathbf{x}, d) = \mathbf{w}^\top \phi(\mathbf{x}, d)$ provides a ranking of maximum $U(\mathbf{x}, \mathbf{y})$ for a given $\mathbf{w}$. To construct the feedback rankings $\bar{\mathbf{y}}_t$ for the Preference Perceptron, we used the *move-to-top* feedback, where documents clicked by the user were moved to the top of the ranking.

The search engine interface was relatively standard with up to 10 search results per page (as seen in Fig 3.1). Users coming to the search engine were randomly assigned one of two groups with equal probability. For users assigned to the learning group, we used the clicked documents of their query to construct the feedback rankings as described above. For users assigned to the evaluation group, the ranking induced by the current weight vector was compared to a baseline ranking that was generated with manually tuned weights. We employed Balanced Interleaving [41, 82], which is a paired, blind test for eliciting a preference between two rankings, for this comparison We record how often

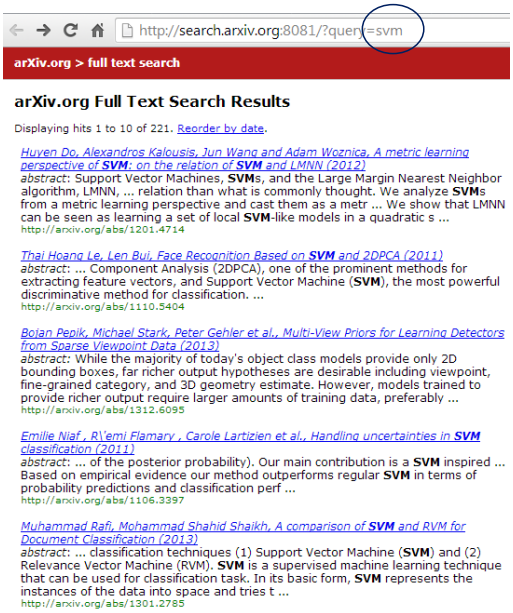

Figure 3.1: Example illustrating the arXiv full-text search engine interface for a query `svm`.

a user prefers a learned ranking over the baseline (*i.e.,* wins a pairwise comparison). Higher the user's preference for the learned ranking (as measured by the *win ratio*) the better the algorithm relative to the baseline. The Perceptron algorithms was initialized to start with the weights of the baseline ranker.

3.5 Instability of the Preference Perceptron

We ran the Preference Perceptron algorithm on the full-text search engine of `arxiv.org` for over a month. The results of the experiment are shown in Figure 3.2, which plots the win ratio[3] against a hand-tuned baseline using Interleaving [41]. As we see from the figure, the Preference Perceptron fails to learn a good ranking function in this online experiment. In particular, the Preference Perceptron (*i.e.,* the black line labeled PrefP[top] in the Figure) only barely improves over the baseline (a value of 1 would indicate equivalence to the baseline).

[3]This measures the ratio of the number of queries for which results of one system (here the learning system) are preferred over the other (here the baseline).

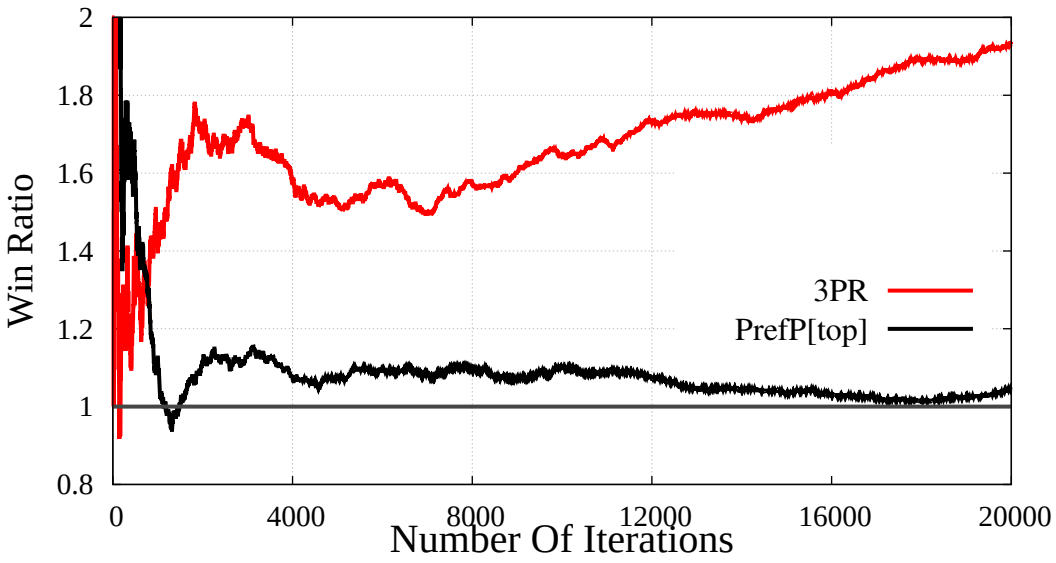

Figure 3.2: Results of the user study showing the ratio of wins versus the hand-tuned baseline for both the Preference Perceptron algorithm [152] (labeled PrefP[top]) and the 3PR algorithm proposed in this dissertation (Sec 3.7).

Figure 3.3 gives some insight into why the Preference Perceptron performs poorly. It shows that the learned rankings for the Preference Perceptron do not stabilize and that the learning process oscillates. In particular, even after thousands of updates, the top 10 documents of the same query before and after 100 update steps only overlap by 4 documents on average.

On the other hand, the algorithm shown in red in Figures 3.2 and 3.3 – the *Perturbed Preference Perceptron for Ranking* (*3PR*) which we introduce in this thesis (Sec 3.7)– achieves substantial improvements over the baseline and does not oscillate.

Before we delve into this proposed algorithm, we will first try to concretely explore why the Preference Perceptron fared so poorly in the online study.

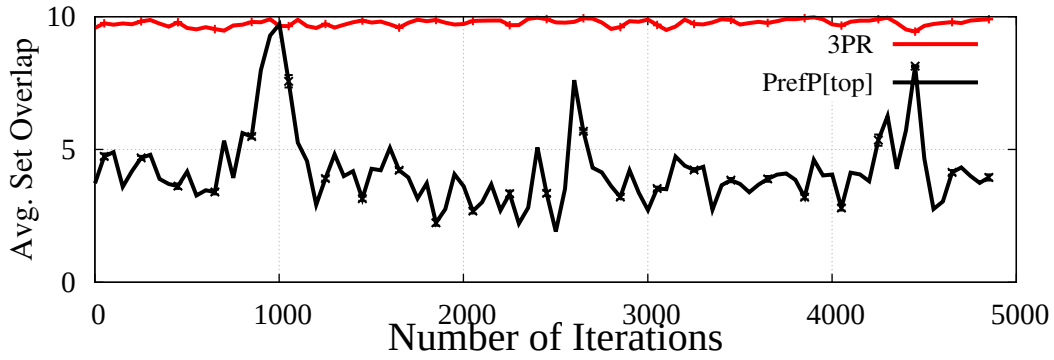

Figure 3.3: Number of common results in the top 10 for the same query using two different models that are 100 learning iterations apart(*i.e.,* $\mathbf{w}_t$, $\mathbf{w}_{t+100}$). Results are binned over intervals of size 50 and averaged over 100 random queries.

3.5.1 Instability: Illustrative example

Why did the Preference Perceptron oscillate? Consider the following toy problem, where the goal is to learn rankings (using the position weighted feature vector construction described in the previous section). In this toy example, document utility is independent of the context $\mathbf{x}$ and only document d_1 has utility $U(\mathbf{x}, d_1) = 1$ (*i.e.,* is relevant), all other documents $d_2 \ldots d_n$ have utility -1. Feature vectors $\phi(\mathbf{x}, d)$ have 2 binary features that exactly reflect utility (*i.e.,* $\phi(\mathbf{x}, d_1) = [1, 0]$ while $\forall i \in [2, n] : \phi(\mathbf{x}, d_i) = [0, 1]$). Now let us consider the following simple user model of interaction: Each iteration, users view the current $\mathbf{y}_t$ (*i.e.,* documents ranked by $\mathbf{w}_t^\top \phi(\mathbf{x}_t, d)$). They examine each document of the ranking $\mathbf{y}^{(i)}$ in order, click the first document **they deem** to have utility 1, and then stop. However, users being an imperfect judge of utility, make each $+1/-1$ utility judgment with only 80% accuracy.

As before, the feedback ranking $\bar{\mathbf{y}}_t$ is constructed from $\mathbf{y}_t$ by swapping the clicked document into rank 1. Let us analyze the behavior of the Preference Perceptron on this toy example. In fact, let us assume that the algorithm is initialized with the perfect weight vector $\mathbf{w}_1 = [1, -1]$, which correctly ranks d_1 first. If the user correctly clicks $\mathbf{y}^{(1)}$, the Preference Perceptron makes no change to $\mathbf{w}_t$. However, whenever the user selects an incorrect $\mathbf{y}^{(i)}$ below (for which there is a $\sim 20\%$ chance), the weight of the first feature decreases and while that of the second increases. Eventually, these updates cause the first component to be negative (and the second positive), which essentially *flips* the ranking with d_1 moving to the *last* position. Even if the system eventually recovers from this catastrophic failure, the same sequence of events will repeat leading to d_1 being placed at the bottom again. Thus, the system oscillates.

The gravity of the problem can be seen in the following simulation results. For $n = 10$ documents and DCG discounting for γ_i (see Section 3.7), the average rank of d_1 within the first 1000 iterations for the Preference Perceptron is 9.36 (1 is best, 10 is worst). In fact, the sole relevant document d_1 is in the worst position for most iterations of the algorithm's run, since it takes a low-probability event of 0.2^9 to correct the ranking, but a high-probability event of 0.2 almost immediately flips it back. Note that "averaging" does not fix this oscillation problem, since it is not a result of unbiased noise. In fact, an Averaged Perceptron [50] showed an average rank of 9.37 in the same simulation.

We should note here that this toy feedback model is α-informative, with the ξs being ≤ 0 for all but the optimal ranking. However, at the optimal, the ξs become large, thus hurting the algorithm.

A careful reader may still wonder why the preference perceptron failed on

the user study, since unlike in the simulation, the weight vector in the live user study was never really optimal. To understand this, consider a small modification to the above toy problem. Instead of just one relevant document, there are k relevant documents in total. However the number of relevant documents is still far smaller than the number of irrelevant documents (as is typically the case in real search engines) *i.e.*, $k << n - k$. Now instead of simply two features, let us say we have $k + 1$ feature where each relevant document $d_i(\forall i \in [2, n])$ has a zero feature vector except for the i^{th} feature whose value is 1. Similarly all irrelevant documents have all but the $(k + 1)^{th}$ feature (whose value is 1) set to 0. Using a similar user interaction model from above (with users clicking on the first k documents they consider relevant) results in the same instability, though here the weight vector may never reach optimal. We again observe oscillations back and forth in the weights of the relevant features.

More generally, the problem is that the feedback used by the preference perceptron is biased and does not fully account for noisy user behavior. This effect becomes particularly prominent for feature components whose weight is nearing the optimal for that feature. The next sections will study how we can remedy this issue by introducing the notion of feedback interventions.

3.6 Adding Feedback Interventions: Stabilizing learning

How can we prevent these oscillations to ensure convergence and improve regret? The key problem as illustrated in the previous section, is that the feedback received by the preference perceptron can be biased and noisy, with the algorithm consequently incurring large slacks ξ_t in Eqn (3.2) – for instance when d_1 is in the top position in the previous toy example, though it is perfectly α-

informative without slack in all other cases. In this section, we introduce the notion of feedback interventions and demonstrate how they can improve learning. We will develop the Perturbed Preference Perceptron to handle this bias in the feedback and guarantee stability.

To motivate the algorithm, consider what happens in the previous toy example if we run the Preference Perceptron, but present the user a *perturbed* ranking where, with 50% probability, we swap the top two documents. Even for the optimal weight vector $\mathbf{w}^*$, note that feedback on the perturbed ranking is now expected α-informative without slack (under the user interaction model of the example). This stabilizes the learning process, since preferences now often reinforce $\mathbf{w}^*$ – namely whenever the relevant document d_1 is at rank two and the user clicks on it. Running the simulation from Section 3.5.1 using the perturbed rankings greatly improves the average rank of d_1 from 9.36 to 2.08.

The perturbations introduced above are just one example of a more general idea, which we term **feedback interventions**. Given that interactive learning systems control what is presented to the users, the idea behind feedback interventions is to present users a slightly modified object (which does not significantly impact the user experience) for which feedback received is far more informative and conducive to good learning. For instance, the above example used pairwise perturbations to reduce the bias and noise in the feedback. These interventions can be even more effective in settings where some user interactions are *low-cost i.e.*, there is more freedom to intervene and make changes without any impact on the user experience. More generally, the next sections will illustrate that jointly designing learning algorithms in conjunction with suitable feedback interventions (and user behavior models) can greatly improve learning and lead

Algorithm 2: Perturbed Preference Perceptron.

Require: $Perturb(\cdots)$, $GetFeedback(\cdots)$

$\mathbf{w}_1 \leftarrow \mathbf{0}$ ▷ Initialize weight vector

for $t = 1$ **to** T **do**

Observe $\mathbf{x}_t$

Compute $\hat{\mathbf{y}}_t \leftarrow \texttt{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$

$\mathbf{y}_t \leftarrow Perturb(\hat{\mathbf{y}}_t)$ ▷ Perturb Object

Present $\mathbf{y}_t$

Obtain feedback $\bar{\mathbf{y}}_t \leftarrow GetFeedback(\mathbf{y}_t)$

Update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$

to sound learning systems.

3.6.1 Perturbed Preference Perceptron

Following the idea of using perturbation to combat feedback bias, Algorithm 2 defines the *Perturbed Preference Perceptron*. It builds off the conventional Preference Perceptron with two key changes. First, the algorithm accepts a subroutine $Perturb(\hat{\mathbf{y}}_t)$ for perturbing the object $\hat{\mathbf{y}}_t = \texttt{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$. Second, since a perturbed object $\mathbf{y}_t$ is presented to the user, the user's preference feedback – and the subsequent update – is relative to $\mathbf{y}_t$, *not* $\hat{\mathbf{y}}_t$.

3.6.2 Theoretical Analysis

We now characterize the regret of the Perturbed Preference Perceptron as a function of the perturbation strategy. This theoretical analysis is fairly general and applies to any perturbation strategy, both randomized and deterministic.

The following theorem bounds the expected regret of the Perturbed Preference Perceptron in terms of two quantities. First, let us re-characterize *expected α-informativeness* of the user feedback analogous to Eq. (3.2),

$$\mathbf{E}_{\bar{\mathbf{y}}_t,\mathbf{y}_t}\Big[\mathbf{w}_*^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)] - \mathbf{E}_{\mathbf{y}_t}\Big[\mathbf{w}_*^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \geq \alpha \left(\mathbf{w}_*^\top \phi(\mathbf{x}_t, \mathbf{y}_t^*) - \mathbf{E}_{\mathbf{y}_t}\Big[\mathbf{w}_*^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big]\right) - \xi_t. \quad (3.4)$$

Note that the feedback $\bar{\mathbf{y}}_t$ is relative to the perturbed $\mathbf{y}_t$, and that expectation is taken over perturbations.

Second, let *affirmativeness* w.r.t. a perturbed $\mathbf{y}_t$ be given by:

$$\mathbf{E}_{\bar{\mathbf{y}}_t,\mathbf{y}_t}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)] - \mathbf{E}_{\mathbf{y}_t}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big].$$

Affirmativeness reflects the relationship between noise in the user feedback and noise from perturbation relative to the current model $\mathbf{w}_t$. Positive affirmativeness indicates that the user feedback typically confirms the ordering based on the current $\mathbf{w}_t$, while negative affirmativeness indicates the opposite. Based on these two quantities, we state the following regret bound.

Theorem 3 *The expected average regret of Algorithm 2 for a perturbation strategy satisfying the following bound on the average affirmativeness,*

$$\frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)\Big] - \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big]\right) \leq \Delta, \quad (3.5)$$

can be upper bounded as

$$\mathbf{E}[REG_T] \leq \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t + \frac{\sqrt{4R^2 + 2\Delta}\,\|\mathbf{w}_*\|}{\alpha\sqrt{T}}. \quad (3.6)$$

The proofs are provided in Appendix A.1. Note that the average affirmativeness (LHS of Eqn (3.5)) is a quantity that can be estimated by the learning algorithm, implying a *dynamic* strategy that determines how to perturb. Note further that in the bound Δ is always zero in the absence of perturbation, which recovers

the conventional Preference Perceptron and its regret bound as a special case. The above bound can be substantially tighter than that of the conventional Preference Perceptron in the noisy feedback situation, since it allows trading-off between Δ and $\sum \xi_t$. In the toy example from above, perturbation reduced $\sum \xi_t$ to zero at a modest increase in Δ.

We also state two corollaries that give bounds on the regret w.r.t. an additive/multiplicative bound on the amount of perturbation.

Corollary 4 *Expected average regret of Alg 2 for a perturbation strategy satisfying*

$$\frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big]\right) \leq \Omega, \tag{3.7}$$

can be upper bounded as

$$\mathbf{E}[REG_T] \leq \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t + \frac{\sqrt{4R^2 + 2\Omega}\,\|\mathbf{w}_*\|}{\alpha\sqrt{T}}. \tag{3.8}$$

Corollary 5 *Expected average regret of Alg 2 for a perturbation strategy satisfying*

$$\forall t : \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \geq (1-\beta)\mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t) \tag{3.9}$$

for $0 \leq \beta \leq 1$, can be upper bounded as

$$\mathbf{E}[REG_T] \leq \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t + \frac{\beta R\|\mathbf{w}_*\|}{\alpha} + \frac{\sqrt{2(4-\beta^2)}R\|\mathbf{w}_*\|}{\alpha\sqrt{T}}.$$

Corollary 4 follows immediately from Theorem 3, and Corollary 5 follows the structure of the proofs in Sec 5.2 for coactive learning with approximate inference (*e.g.*, Theorem 9). The bounds presented above not only provide a theoretical sanity check for Algorithm 2 and the concept of perturbations (and more generally feedback interventions), but also give explicit guidelines for designing effective perturbation strategies that we will exploit in the next section.

3.7 Perturbed Preference Perceptron for Ranking: 3PR

Ranking is one of the most common learning tasks for online systems, as it is the basis for search and recommendation. These systems are ideally suited for coactive learning, since they can easily sense user interactions that provide (noisy) feedback. We now develop perturbation and feedback strategies for the Perturbed Preference Perceptron that ensure stable learning of ranking functions.

For a perturbed ranking $\mathbf{y}$, let $\bar{\mathbf{y}}$ be a feedback ranking that is derived from interactions (*e.g.,* clicks) in $\mathbf{y}$. Our goal is a perturbation and feedback strategy such that $\bar{\mathbf{y}}$ fulfills Eq. (3.4) with large α and small ξ. Let us consider some properties such a strategy should have.

First, it is desirable to perturb *uniformly* throughout the ranking, so that a user experiences the same amount of perturbation no matter how deep they explore. Second, we would like to make only *local* perturbations to minimally alter the ranking. Third, the construction of the feedback ranking $\bar{\mathbf{y}}$ should be *robust* to noisy clicks, limiting the increase in ξ in Eq. (3.4).

These desiderata naturally lead to the perturbation and feedback strategy in Algorithm 3, which follows the FairPairs method proposed in [131]. The top-scoring ranking $\hat{\mathbf{y}}$ (*e.g.,* $\hat{\mathbf{y}} = [d_1, d_2, d_3, d_4, d_5, d_6, ...]$) is split into adjacent pairs of documents (*e.g.,* $[(d_1, d_2), (d_3, d_4), (d_5, d_6), ...]$), and each pair is swapped with probability p to produce the perturbed ranking $\mathbf{y}$ (*e.g.,* , $\mathbf{y} = [(d_2, d_1), (d_3, d_4), (d_6, d_5), ...]$). Whenever the user clicks on the bottom document of a pair, the top and bottom document are swapped to produce the feedback ranking $\bar{\mathbf{y}}$ (*e.g.,* for clicks on $\{d_1, d_4, d_6\}$ in $\mathbf{y}$, we construct $\bar{\mathbf{y}} = [(d_1, d_2), (d_4, d_3), (d_6, d_5), ...]$). We call Algorithm 2 using the functions from Algorithm 3 the *Perturbed Preference Perceptron for Ranking (3PR)*.

Algorithm 3: Perturbation and feedback for the Perturbed Preference Perceptron for Ranking (3PR).

Function FORMPAIRS()

With prob 0.5: **return** $(\{1,2\},\{3,4\},\{5,6\}\cdots)$

else: **return** $(\{1\},\{2,3\},\{4,5\},\{6,7\}\cdots)$

Function PERTURB($\hat{\mathbf{y}}$, p)

$\mathbf{y} \leftarrow \hat{\mathbf{y}}$ ▷ Initialize with top-scoring ranking

$Pairs \leftarrow FORMPAIRS()$

for $i = 0 \cdots len(Pairs)$ **do**

$\{j, j+1\} \leftarrow Pairs[i]$ ▷ Get Pair

With prob p:

swap($\mathbf{y}[j]$,$\mathbf{y}[j+1]$); swap($Pairs[i][0]$,$Pairs[i][1]$)

return ($\mathbf{y}$, $Pairs$)

Function GET-FEEDBACK($\mathbf{y}$, $clicks$, $Pairs$)

$\bar{\mathbf{y}} \leftarrow \mathbf{y}$ ▷ Initialize with presented object

for $i = 0 \cdots len(Pairs)$ **do**

$\{j_{upper}, j_{lower}\} \leftarrow Pairs[i]$ ▷ Get Pair

if $\mathbf{y}[j_{lower}] \in clicks$ AND $\mathbf{y}[j_{upper}] \notin clicks$ **then**

swap($\bar{\mathbf{y}}[j_{upper}]$, $\bar{\mathbf{y}}[j_{lower}]$)

return $\bar{\mathbf{y}}$

We now establish regret bounds for the 3PR algorithm, using the joint feature map $\phi(\mathbf{x}, \mathbf{y})$ for queries $\mathbf{x}$ and rankings $\mathbf{y}$ described in Section 3.4. In particular, we use position-discounting factors $\gamma_i = \frac{1}{\log_2(i+1)}$ as in the DCG metric [117].

Proposition 6 *The 3PR with swap probability p has regret:*

$$\leq \frac{\sum_{t=1}^{T} \xi_t}{\alpha T} + \frac{p(1-\frac{\gamma_2}{\gamma_1})R\|\mathbf{w}_*\|}{\alpha} + \frac{\sqrt{2(4-p^2(1-\frac{\gamma_2}{\gamma_1})^2)}R\|\mathbf{w}_*\|}{\alpha\sqrt{T}}.$$

The proof is provided in Appendix A.1 along with proofs for the other theorems.

On the one hand, the *3PR* algorithm provides the first exploration strategy with a regret bound for FairPairs feedback. On the other hand, the regret bound implies that the swapping of pairs does not need to necessarily be "fair" (*i.e.*, $p = 0.5$). For example, consider a **dynamic** swap strategy that, at iteration t, determines its perturbation based on the cumulative affirmativeness $R_t = \sum_{i=1}^{t-1} \mathbf{w}_i^\top \phi(\mathbf{x}_i, \bar{\mathbf{y}}_i) - \mathbf{w}_i^\top \phi(\mathbf{x}_i, \mathbf{y}_i)$ and the maximum perturbation $D_t = \mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}'_t)$, where $\mathbf{y}'_t$ is the ranking obtained by swapping all pairs in $\hat{\mathbf{y}}_t$. Note that D_t is an a priori bound on the maximum affirmativeness of the user feedback at iteration t. Based on these observable quantities, we propose the following dynamic adaptation rule for the swap probability with the following regret bound.

Proposition 7 *For $\Delta \geq 0$, dynamically setting the swap prob. of 3PR to be $p_t \leq max(0, \frac{\Delta \cdot t - R_t}{D_t})$ has regret:*

$$\leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{\|\mathbf{w}_*\|}{\alpha\sqrt{T}} \sqrt{4R^2 + 2\Delta + (\gamma_1 - \gamma_2)R\sqrt{\frac{4R^2 + 2\Delta}{T}}}.$$

3.7.1 ArXiv User Study Results

To investigate the real-world effectiveness of the *3PR* algorithm compared to the conventional Preference Perceptron (*PrefP*), we repeated the user study from Section 3.4) on the full-text search engine of `arxiv.org` using 3PR instead. Results were collected in two subsequent runs, one for each method. As done

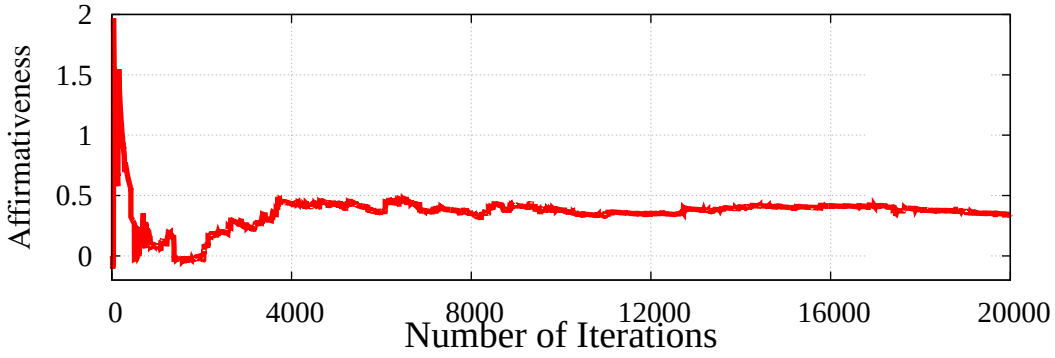

Figure 3.4: Average affirmativeness of 3*PR* in user study.

for the Preference Perceptron, the learning algorithm was initialized to start with the weights of the baseline ranker. For the learning iterations, pairs were swapped with probability 0.5. Paired feedback was constructed as described in Algorithm 3. Further details of the study are provided in Appendix A.2.

Figure 3.2 shows the results of the experiment, plotting the win ratio of each learning method over the baseline. While *PrefP* initially performs well, its win ratio eventually hovers only slightly above 1. The *3PR* method, on the other hand, converges to a win-ratio of 1.9, which is large (and highly significant according to a Binomial Sign Test) compared to the experiments in [41]. Finally, Figure 3.4 shows the average affirmativeness Δ from Theorem 3. It shows that Δ is positive and stabilizes, indicating an appropriate amount of perturbation.

3.8 Experiments on Benchmark Data

To get more detailed insights into the empirical performance of the proposed methods, we also conducted offline experiments on benchmark datasets.

First, we use the Yahoo! learning to rank dataset [40] (abbreviated *Websearch*), which consists of roughly 28k queries and 650k documents (*i.e.,* URLs). For each query-url pair in the dataset, there is a joint feature vector $\phi(\mathbf{x}, d)$ of 700 features and an integer relevance rating in the range 0-4. In each iteration, the system is given a query and presents a ranking. In total, the coactive learning system were run for 28k iterations. All results presented below are averaged over 20 different runs (by randomizing the query stream order).

Second, we simulate two news recommendation tasks, using the RCV1 [104] and the 20 Newsgroups datasets (abbreviated *News*). The RCV1 corpus contains over 800k documents that each belong to one or more of 103 topics, while the News dataset contains 20k documents that each belong to one of 20 topics. We used TF-IDF (Term Frequency-Inverse Document Frequency) features as is standard for these tasks. This leads to feature set totaling 3k size for RCV1 and 1k for News[4]. In these experiments, we simulated user interests by equating users with single topics. The user's goal is to be presented with documents corresponding to their topic. The algorithms were run for 50K iterations for RCV and 10K for News (by cycling through the data), and the results are averaged over all users (*i.e.,* topics).

We assume the following model of user interaction. The user scans the ranking from the top down to the tenth result and clicks on up to five results. To study the stability of the different algorithms, clicks are corrupted by noise. For RCV1 and News, a user goes down the ranking and clicks on relevant documents, but with η chance of incorrectly assessing the relevance of a document ($\eta = 0.2$). On the search dataset, the user's relevance assessment are corrupted by adding independent Gaussian noise ($\sigma = 1$) to the true relevance of each

[4]Feature selection, using the maximum class χ^2 metric, was performed similar to [104].

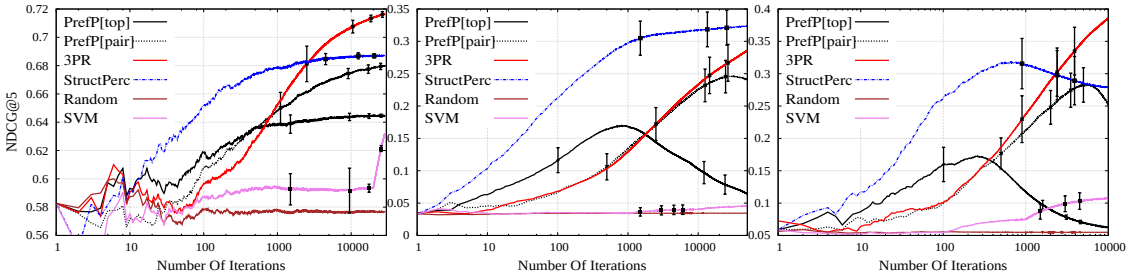

Figure 3.5: Learning curves for all algorithms on Websearch (left), RCV1 (middle), and News (right).

document; the user then clicks on the 5 documents with highest (corrupted) relevance in the top 10.

3.8.1 What is the Generalization Performance of the Perturbed Preference Perceptron?

First, let us compare the *3PR* against alternative algorithms, including the conventional Preference Perceptron where clicked documents are moved to the top of the feedback ranking (*PrefP[top]*). We also consider a variant of the conventional Preference Perceptron that uses the same paired feedback as *3PR*, but has swap probability zero (*PrefP[pair]*).

To compare with a regularized batch learner, a ranking SVM with move-to-top feedback was trained at (10,100,1k,10k,20k) iterations using a setup similar to [152]. Between training steps, the current predictor is used to present rankings. For this experiment, we retrospectively pick the best C value (per run) and report the NDCG@5 corresponding to that C (*i.e.*, biasing in favor of the SVM).

As a (rough) upper bound, we consider a *Structured Perceptron* [50] that is trained with the optimal $\mathbf{y}^*$ without added noise. This simulates clean and

exhaustive expert feedback, which is typically unobtainable in practice. As a lower bound, we report performance of uniformly random document rankings.

The results are shown in Figure 3.5. It can be seen that the *3PR* achieves a significantly higher NDCG@5 compared to other online algorithms *PrefP[top]* and *PrefP[pair]* at the end of the runs. In fact, *PrefP[top]* fails catastrophically on two of the datasets like in the toy example from Section 3.5.1. *PrefP[pair]* is more stable, but shows similar deterioration as well. An interesting extension could be the combination of aggressive move-to-top feedback in early iterations with more conservative *3PR* updates later.

Due to the biased training data that violates the IID model, the SVM performs poorly. We conjecture that more frequent retraining would improve performance, but be orders of magnitude more computationally expensive (especially with realistic model selection).

The Structured Perceptron learns faster than *3PR*. However, despite receiving much stronger training data (optimal $\mathbf{y}^*$ without feedback noise), its eventual performance is worse than *3PR* on two datasets. This may be surprising at first glance. However, it is known that Perceptron-style algorithms do not always work well on multiclass/structured problems without good linear fit, and can even degenerate [42, 103]. Intriguingly, the *3PR* seems less affected by this problem.

Table 3.1: NDCG@5 of presented and perturbed rankings after maximum number of iterations.

	Websearch	RCV1	News
Presented $\mathbf{y}$	.717 ± .002	.286 ± .028	.386 ± .035
Predicted $\hat{\mathbf{y}}$	.723 ± .002	.291 ± .028	.397 ± .035

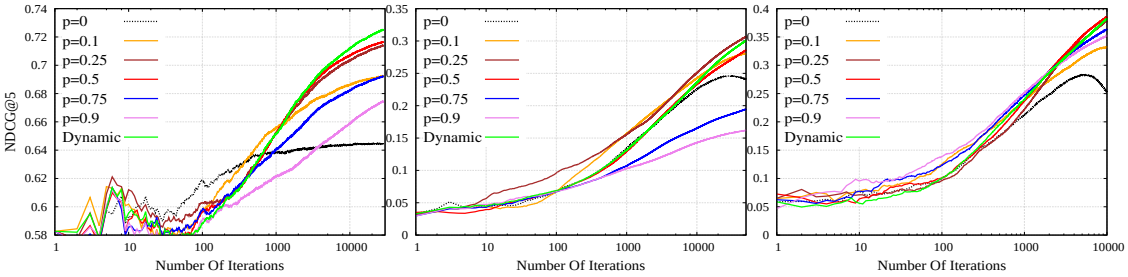

Figure 3.6: NDCG@5 of the *3PR* algorithm for different swap probabilities and the dynamically adapted swap probability on Websearch (left), RCV1 (middle), and News (right).

3.8.2 How does the Perturbed Ranking Compare to the Optimal Prediction?

In the case of *3PR*, the algorithm first computes the argmax ranking $\hat{\mathbf{y}}$ but then presents the perturbed ranking $\mathbf{y}$. While the previous section showed the NDCG@5 of the presented rankings $\mathbf{y}$, Table 3.1 shows the NDCG@5 for both $\mathbf{y}$ and $\hat{\mathbf{y}}$. As expected, the presented rankings are of slightly lower quality than $\hat{\mathbf{y}}$ due to perturbation. However, this small loss in quality leads to a big gain in the overall learning process in the long run — as demonstrated by the poor performance of $PrefP[pair]$. An interesting extension would be to present the perturbed ranking $\mathbf{y}$ and learn in only some of the iteration, but *exploit* by presenting $\hat{\mathbf{y}}$ and not learn in the rest of the iterations.

3.8.3 How much Perturbation is Needed?

While complete lack of perturbation leads to divergence, it is unclear whether a swap probability of 0.5 is always optimal. Intuitively, we expect that with low noise, smaller perturbations suffice to achieve high performance, while at higher noise levels, perturbation probabilities need to be higher to overcome the noise.

Figure 3.6 explores the effect of different perturbation rates p in Algorithm 3 on the performance of the *3PR*. It appears that a swap probability of more than 0.5 usually hurts. While 0.5 typically performs reasonably well, 0.25 produces the best performance on RCV1.

3.8.4 Can we Automatically Adapt the Perturbation Rate?

Ideally, we would like to automatically select an appropriate swap probability. Note that this does not need to be a single fixed number, but can change over the learning run. Proposition 7 defined such a perturbation strategy that accounts for the current affirmativeness and adjusts the swap probability to optimize the regret bound in Theorem 3. The results of this dynamic strategy using $\Delta = 0$ are also included in Figure 3.6. As we see from the figure, the method is able to adjust the swap rates to achieve performance among the best.

Figure 3.7 shows how the swap probability chosen by the dynamic strategy varies. It can be observed that the swap probability first increases and then eventually decreases to exploit more often. When changing the noisiness of the user feedback, we find that the strategy automatically accounts for larger noise by increasing the swap rate relative to the low noise setting.

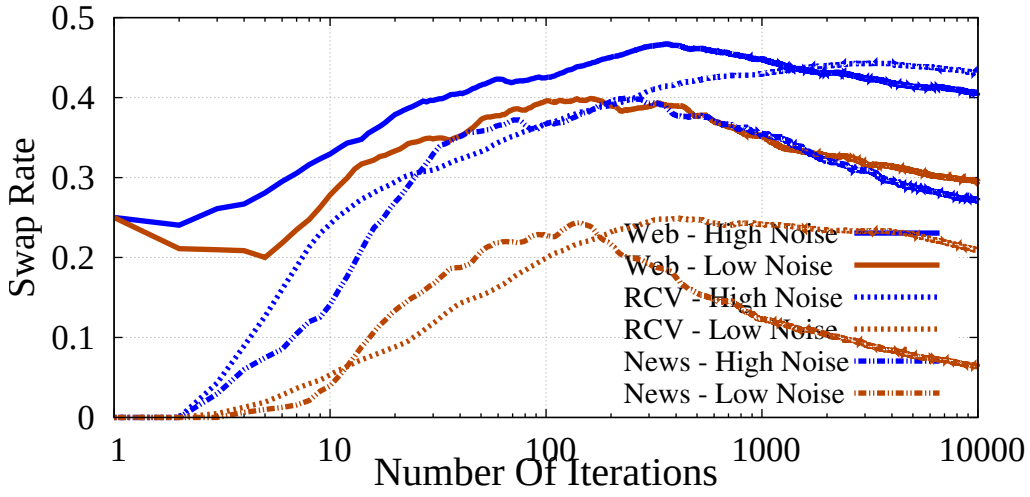

Figure 3.7: Change in average swap probability of the dynamic method with $\Delta = 0$ low and high feedback noise.

3.8.5 Effect of Noise on the Perturbed Preference Perceptron

Our motivation for the *3PR* algorithm was the inability of *PrefP[top]* to handle the noise in the feedback it encountered in the user study. Therefore, all benchmark experiments we reported included feedback noise as described at the start of Sec 3.8. But how does the *3PR* algorithm perform without added noise?

Figure 3.8 compares the performance of *3PR* to that of *PrefP[top]* and *PrefP[pair]* with and without user feedback noise. Even with no feedback noise, *3PR* outperforms *PrefP[top]* and is at least comparable to *PrefP[pair]*. Furthermore, the performance of *3PR* declines much less when noise is introduced, as compared to the other algorithms.

Note that "*no noise*" is somewhat of a misnomer. While we did not add any noise, even the expert provided ratings probably contain some amount of noise. Moreover, any feedback that cannot be explained by a linear model appears as noise to the algorithm, which is likely to be a substantial source of noise in any

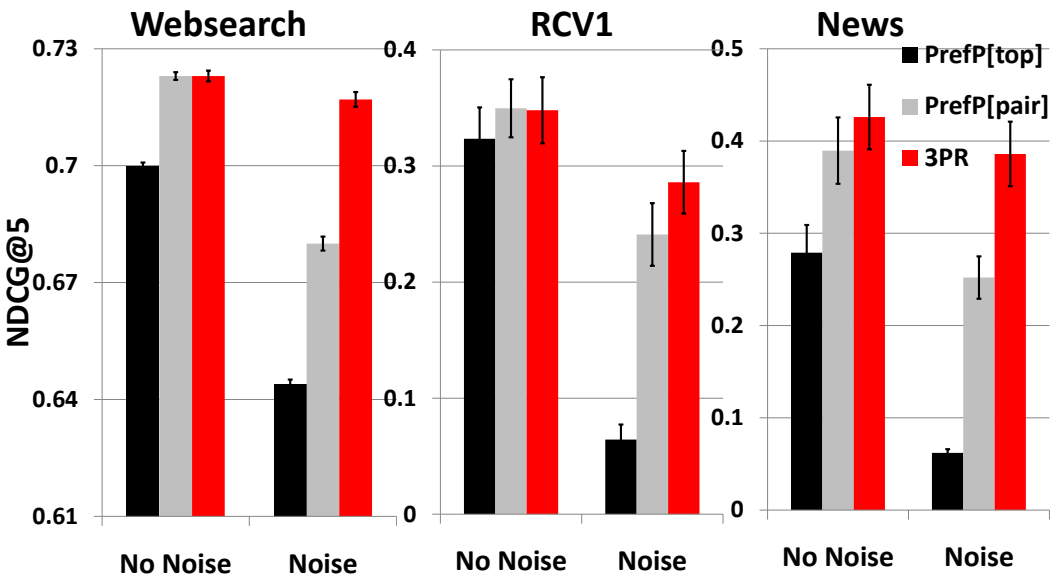

Figure 3.8: Performance of *PrefP[top]*, *PrefP[pair]* and *3PR* at the maximum number of iterations with and w/o feedback noise.

real-world application. The *3PR* algorithm handles this gracefully.

3.9 Summary

This chapter studied an interactive learning model called Coactive Learning. It presented the Perturbed Preference Perceptron, an online algorithm for learning from biased and noisy preferences in the coactive learning model. Unlike existing methods, the presented algorithm was shown to be stable and free of oscillations. The key idea was the use of controlled perturbations of the predictions as feedback interventions. Theoretical regret bounds that characterize the behavior of the new algorithm were also presented. Perturbation strategies were developed focusing on learning to rank. The proposed algorithms were shown to substantially outperforms existing methods in benchmark experiments. Furthermore, an online live user study on a search engine, exemplified the importance of jointly designing learning algorithm, user model and feedback interventions.

Part III

Modeling Complex User Behavior

This part of the dissertation will look at one of the three key aspects of interactive learning: the user behavioral model. In particular, it explores a set of problems where user behavior is highly complex and thus in need of new behavior models and learning algorithms. The problems covered in this part of the dissertation all deal with diversity in search and recommendation tasks.

We will first begin by introducing the problem of intrinsic diversity and understanding the significance of the problem in the context of web search. We will also provide ways to identify these complex tasks, both from logged interaction data as well as on-the-fly. Finally we will explore ways to improve retrieval performance for these complex search tasks.

We will then explore means to interactively learn to diversify for these complex tasks. More specifically, we will describe coactive learning techniques that learn to diversify from user interaction data. These techniques will be developed for both kinds of diversified retrieval tasks: intrinsic and extrinsic. The resulting algorithms will not only have provable theoretical guarantees, but also significantly better empirical performance than existing methods. Along the way, this dissertation will introduce the first-known algorithms for learning to extrinsically diversify across queries, from user interaction data.

This part will conclude by briefly describing how interactivity can be introduced on-the-fly into objects such as rankings, and how this can help alleviate problems in the diversified retrieval field. It will also provide learning algorithms to learn to predict these interactive structures.

CHAPTER 4

EXPLORING INTRINSIC DIVERSITY IN WEB SEARCH

The search and information retrieval literature has primarily focused on improving retrieval for a single query at a time. However, given the ever-increasing complexity of user search needs, there is an urgent need for search engines to help users tackle complex search tasks in an efficient manner [87, 109]. Within the context of this work, we focus on one specific type of information seeking need that drives interaction with web search engines and often requires issuing multiple queries – namely intrinsically diverse tasks [132]. Table 4.1 gives examples of two intrinsically diverse tasks observed in a commercial web search engine. **Intrinsic diversity (ID)**, where diversity is a desired property of the retrieved set of results to satisfy the current user's immediate information need, is meant to indicate that diversity is intrinsic to the need itself. It requires that results are both relevant to a single topical intent as well as diverse across aspects, rather than simply covering additional topical interpre-

Initiator query	Successor queries

`snow leopards`	`snow leopard pics`
	`where do snow leopards live`
	`snow leopard lifespan`
	`snow leopard population`
	`snow leopards in captivity`

`remodeling ideas`	`cost of typical remodel`
	`hardwood flooring`
	`earthquake retrofit`
	`paint colors`
	`kitchen remodel`

Table 4.1: Examples of intrinsically diverse search tasks, showing the first (initiator) query and the following (successor) queries from the same search session.

tations. This is in contrast to extrinsic diversification techniques that provide diversity to cope with uncertainty in query intent (*e.g.,* `[jaguar]`). Unfortunately extrinsic diversification methods like maximal marginal relevance (MMR) [35] do not satisfy these requirements well (*cf.* Sec. 4.4). While most diversification research have focused primarily on extrinsic diversity (see Sec 2.4.2), recent work [19] has indicated that intrinsic diversity (ID) is becoming an increasingly important issue as many real-world web search tasks are commonly ID and require significant user effort. Thus, improvements in retrieval quality that address intrinsically diverse needs have potential for broad impact.

Intrinsically diverse tasks typically are exploratory, comprehensive, survey-like, or comparative in nature. ID tasks that are commonly seen in web search sessions include (along with session statistics such as average number of queries, total time, and prevalence of such sessions as per [19]): discovering more information about a specific topic (6.8 queries, 13.5 minutes, 14% of all sessions); comparing products or services (6.8 queries, 24.8 minutes, 12% of all sessions); finding facts about a person (6.9 queries, 4.8 minutes, 3.5% of all sessions); and learning how to perform a task (13 queries, 8.5 minutes, 2.5% of all sessions). Intrinsically diverse tasks typically result from users seeking different opinions on a topic, exploring or discovering aspects of a topic, or trying to ascertain an overview of a topic [132]. While a single, comprehensive result on the topic may satisfy the need when available, several or many results may be required to provide the user with adequate information [132]. As seen in the example tasks, a user starting with `[snow leopards]` may be about to engage in an exploratory task covering many aspects of snow leopards including their lifespan, geographic dispersion, and appearance. Likewise when investigating remodeling ideas, a user may wish to explore a variety of aspects including

cost, compliance with current codes, and common redecoration options. Note that the user may in fact discover these aspects through the interaction process itself, similar to exploratory and faceted search [118, 130]. However, unlike the more open-ended paradigm provided by exploratory search, we desire a solution that is shaped by the current user's information need and is able to discover and associate relevant aspects for a topic automatically in a data-driven fashion. For example, for the query `[snow leopards]`, our goal is to enable deeper user-driven exploration of that topic, by proactively searching for the relevant information that the user might want during the course of a session on that topic, thus reducing the time and effort involved in manual reformulations, aspect discovery, and so on.

To this end, we aim to design a system that addresses two key problems needed for ID retrieval: detecting the start of an ID task, and computing an optimal set of ID documents to return to the user given engagement on an ID task. For the former, the system must be capable of predicting when a user is likely to issue multiple queries to accomplish a task, based on seeing their first "initiator query". To do this, we first develop a set of heuristic rules to mine examples of authentic intrinsic diversity tasks from the query logs of a commercial search engine. The resulting tasks provide a source of weak supervision for training classification methods that can predict when a query is initiating an intrinsically diverse task. With these predictive models, we characterize how ID initiators differ from typical queries. We then present our approach to intrinsically diversify given a query. In particular, rather than simply considering different intents of a query, we incorporate results related to other important aspects of the topic by estimating the relevance relationship between the aspect and the original query. Given the intrinsically diverse sessions identified through log

analysis, we demonstrate that our approach to intrinsic diversification is able to identify more of the relevant material found during a session given less user effort. We show these methods to be able to proactively retrieve content for future queries before the user has searched for them. Importantly, these future queries are neither simple reformulations nor completely unrelated, but are queries on the particular task that the user has started. Overall, the proposed approach is able to shown to outperform standard baselines.

4.1 Intrinsically Diverse Tasks

An *intrinsically diverse task* is one in which the user requires information about *multiple, different aspects* of the *same topical* information need. In practice, a user most strongly demonstrates this interest by issuing multiple queries about different aspects of the same topic. We are particularly interested in identifying the common theme of an intrinsically diverse task *and* when a user initiated the task. We unify these into the concept of an *initiator query* where, given a set of queries on an intrinsically diverse task, the query among them that is most general and likely to have been the first among these set of queries is called the initiator query. If multiple such queries exist, then the first among them from the actual sequence (issued by the user) is considered the initiator. We give importance to the temporal sequence since the goal is to detect the initiation of the task and provide support for it as soon as possible.

While previous work has defined the concept of intrinsic diversity, there has been no further understanding of the problem or means to obtain data. We now identify and analyze authentic instances of intrinsically diverse search behavior, extracted from large-scale mining and analysis of query logs from a commercial

search engine.

4.1.1 Mining intrinsically diverse sessions

Intuitively, intrinsically diverse (ID) tasks are topically coherent but cover many different aspects. To automatically identify ID tasks *in situ* where a user is attempting to accomplish the task, we seek to codify this intuition. Furthermore, rather than trying to cover all types of ID tasks, we focus on extracting with good precision and accuracy a set of tasks where each task is contained within a single search session. As a "session" we take the commonly used approach of demarcating session boundaries by 30 minutes of user inactivity [166]. Once identified, these mined instances could potentially be used to predict broader patterns of cross-session intrinsic diversity tasks [3, 97], but we restrict this study to mining and predicting the initiation of an ID task within a search session and performing whole-session retrieval at the point of detection.

To mine intrinsically diverse sessions from a post-hoc analysis of behavioral interactions signals with the search results, we developed a set of heuristics to detect when a session is topically coherent but covering many aspects. These can be summarized as finding sessions that are: (1) longer – the user must display evidence of exploring multiple aspects; (2) topically coherent – the identified aspects should be related to the same overall theme rather than disparate tasks or topics; (3) diverse over aspects – the queries should demonstrate a pattern beyond simple reformulation by showing diversity. Furthermore, since the user's interaction with the results will be used in lieu of a contextual relevance judgment for evaluation, we also desire that we have some "satisfied" or "long-

click" results where we define a satisfied (SAT) click similar to other work as having a dwell of $\geq 30s$ or terminating the search session [61, 66].

Given these criteria, we propose a simple algorithm to collect intrinsically diverse user sessions. Our algorithm uses a series of filters, explained in more detail below. When we refer to "removing" queries, we mean they were treated as not having occurred for any subsequent analysis steps. For sessions, with the exception of those we "remove" from further analysis in Step 4, we label all other sessions as intrinsically diverse or *regular* (*i.e.,* not ID). We identify the *initiator* query as the first query that remains after all query removal steps, and likewise a *successor* query is any remaining query that follows the initiator in the session. More precisely, we use the following steps (in sequence) to filter sessions:

1. **Remove frequent queries:** Frequent queries – such as *facebook* or *walmart* – that are often interleaved with more complex tasks can obscure the more complex task the user is accomplishing. Therefore, we remove the top 100 queries by frequency as well as frequent misspellings related to these queries.
2. **Collapse duplicates:** We collapse any duplicate of a query issued later in the session as representing the same aspect but record all SAT clicks across the separate impressions.
3. **Only preserve manually entered queries:** To focus on user-driven exploration and search, we removed queries that were not manually entered, *e.g.,* those obtained by clicking on a link such as by query suggestion or searches embedded on a page.
4. **Remove sessions with no SAT Document:** Since we would like to even-

tually measure the quality of re-rankings for these session queries in a personal and contextual sense, we would like to ensure that there is at least one long-dwell click to treat as a relevance judgment. While this is not required for a session being an ID session, we simply require it for ease of evaluation. Thus, we removed sessions with no SAT clicks.

5. **Ensure topical coherence:** As ID sessions have a common topic, we removed any successor query that did not share at least one common top ten result with the initiator query. Note that this need not be the same result for every aspect. While this restricts the set of interaction patterns we identify, it enables us to be more precise, while ensuring semantic relatedness, and does not rely on the weakness of assuming one fixed static ontology.
6. **Ensure diversity in aspects:** Although we desire topical coherence across the queries, we do not want to identify simple reformulations or spelling corrections as aspects. Thus we restrict the syntactic similarity with the initiator query to avoid identifying trivial difference as substantially different aspects. To measure query similarity robust to spelling variations, we consistently use *cosine similarity with character trigrams* in this work. In particular, we remove queries where the similarity was more than 0.5.
7. **Remove long queries:** We observed a small fraction of sessions matching the above filters appear to consist of copy/paste homework questions on a common topic. While potentially interesting, we focus in this paper on completely user-generated aspects and introduce a constraint on query length, removing queries of length at least 50 characters.
8. **Threshold the number of distinct aspects:** Finally, to focus on diversity and complexity among the aspects, we threshold on the number of dis-

tinct successor queries. We identify a query as distinct when its maximum pairwise (trigram character cosine) similarity with any preceding query in the session is less than 0.6. Any session with less than three distinct aspects (including the initiator) are labeled as regular and those with three or more aspects are labeled as intrinsically diverse.

Putting everything together, we ran this algorithm on a sample of user sessions from the logs of a commercial search engine from the period April 1–May 31, 2012. We used log entries generated in the English-speaking United States locale to reduce variability caused by geographical or linguistic variation in search behavior. Starting with 51.2M sessions comprising 134M queries, applying all but the *SAT-click* filter, with the *Number of Distinct Aspects* threshold at two, led to more than 497K ID sessions with 7.0M queries. These ID tasks accounted for 1.0% of all search sessions in our sample, and 3.5% of sessions having 3 queries or more (14.4M sessions)[1]. Further applying the SAT-click filter reduced the number to 390K. Finally, focusing on the more complex sessions by setting the Number of Distinct Aspects filter to three, reduced this to 146K sessions.

Given that ID sessions require multiple queries, we hypothesize that ID sessions account for a disproportionately larger fraction of *time spent searching* by all users. To test this, we estimated the time a user spent in a session by the elapsed time from the first query to the last action (*i.e.*, query or click). Sessions with a single query and no clicks were assigned a constant duration of 5 seconds. Here, the time in session includes the whole session once an ID task was identified in that session. Our hypothesis was confirmed: while ID sessions with at least 2 distinct aspects represented 1.0% of all sessions, they accounted

[1]Because we do not focus on more complex ID information seeking, such as tasks that span multiple sessions, the true percentage associated with ID tasks is likely to be larger.

for 4.3% of total time spent searching, showing the significant role ID sessions play in overall search activity. For more details on this extraction process as well as an evaluation of its' accuracy (using annotated data), we refer the interested reader to the journal article on this topic [135].

4.2 Predicting Intrinsically Diverse Task Initiation

Given that we may want to alter retrieval depending on whether the user is seeking intrinsic diversity or not, we ask the question of whether we can identify the initiator queries for intrinsically diverse tasks. We do so by treating this as a classification problem. In particular, while we used the behavioral signals of interaction between the initiator and successor queries of a session to automatically label queries with a (weak) supervised label in the previous section, here we ask if we can predict what the label would be in the absence of those interaction signals – a necessary ability if we are to detect the user's need for intrinsic diversity in an operational setting. Ultimately our goal is to enable a search engine to customize the search results for intrinsic diversity only when appropriate, while providing at least the same level of relevance on tasks predicted to be regular. Recognizing that in most operative settings, it is likely important to invoke a specialized method of retrieval only when confident, we present a precision-recall tradeoff but focus on the high precision portion.

4.2.1 Experimental Setting

Data: We used a sample of initiator queries from the intrinsically diverse sessions described in Sec. 4.1.1 as our positive examples. The first queries (after

removing common queries as in Step 1 of Sec. 4.1.1) from *regular* sessions were used as negative examples. Note that since the label of a query, *e.g.,* `[foo]`, comes from the session context, it is possible that `[foo]` occurs in both positive and negative contexts. In order to only train to predict queries that were clearly either ID or regular, we dropped such conflicting queries from the dataset; this only occurred 1 out of every 5K ID sessions. Also to weigh each task equally instead of by frequency, we sample by type: *i.e.,* we treat multiple occurrences of a query in the positive (resp. negative) set as a single occurrence. Finally, we downsample to obtain a 1:1 ratio from the positive and negative sets to create a balanced set. The dataset was sampled to contain 61K queries and split into an 80/5/15 proportion (50000 training, 3000 validation, 8000 test) with no class bias.

Classification: We used SVMs[81] with linear kernels, unless mentioned otherwise. We varied the regularization parameter (C) over the values: $\{10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, \ldots, 500, 10^{3}\}$. Model selection was done using the validation set by selecting the model with the best precision using the default margin score threshold (*i.e.,* 0).

Feature Set	Examples	Size	Coverage	Norm?	Log?
Text	Unigram Counts	44140	100%	**No**	**No**
Stats	# Words, # Characters, # Impressions, Click Count, Click Entropy	10	81%	**Yes**	**Yes**
POS	Part-of-Speech Tag Counts	37	100%	**No**	**No**
ODP	Five Most Probable ODP Class Scores from Top Two Levels	219	25%	**Yes**	**Yes**
QLOG	Avg Similarity with co-session queries, Avg session length, Distribution of occurrences within session (start/middle/end)	55	44%	**Yes**	**No**

Table 4.2: Features used for identification of initiator queries along with cardinality, coverage and information as to whether they were normalized or log transformed

Features: The features are broadly grouped into 5 classes as shown in Table 4.2. Apart from the text and POS tag features, all other features were normalized to zero mean, unit variance. Features with values spanning multiple orders of magnitude, such as the *number of impressions*, were first scaled down via the log function. Due to the large scale of our data, coverage of some features is limited. In particular, query classification was done similar to [25] by selecting the top 9.4M queries by frequency from a year's query logs previously in time and then using a click-based weighting on the content-classified documents receiving clicks[2]. Likewise **Stats** and **QLOG** features were built from four months' worth of query logs and have limited coverage as a result. The query logs chosen to build these features were from prior to April 2012 to ensure a fair experimental setting with no overlap with the data collection period of the intrinsically diverse or regular sessions. We found the coverage of these features to be roughly the same for both the positive and negative classes.

We also note that the cardinality of some feature sets will depend on the training set (*e.g.,* vocabulary size of *Text* grows with more training data); the values listed in Table 4.2 are for the default training set. Most of our experiments will use all of the 5 feature sets; the effect of using only a subset of the feature sets is explored in Sec. 4.2.3.

4.2.2 Can we predict ID task initiation?

To begin with, we would like to know the precision-recall tradeoff that we can achieve on this problem. Figure 4.1 (Left) shows the precision-recall curve for a linear SVM trained on 50K examples with all features. The result is a curve

[2]For greater coverage this could be extended to a rank-weighted back-off as described in that paper [25].

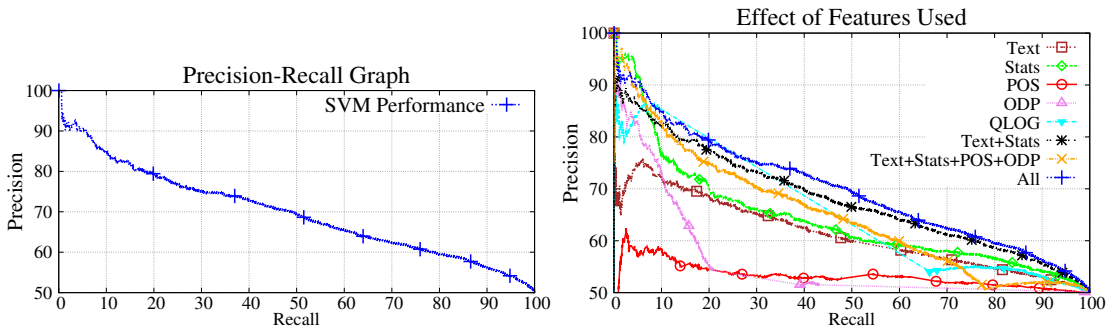

Figure 4.1: P-R curve for predicting ID task initiation (Left) & Change in initiator classification performance with feature set (Right)

with clear regions of high precision, indicating that the SVM is able to identify initiator queries in these regions quite accurately. For example, we can identify 20% of ID tasks with 80% precision. Furthermore, performance is better than random (precision of 50% since classes are balanced) along the entire recall spectrum.

4.2.3 Which features were most important?

We next investigate the effect of using different subsets of the features on performance (Fig 4.1 - Right). First, we note that **Stats, QLOG** and **ODP** feature sets help identify only a small fraction of the initiator queries but do so with high precision. On the other hand, the **Text** and **POS** feature sets, which have high coverage, provide some meaningful signal for all the queries, but cannot lead to high precision classification. We also find that a combination of features, such as the **Text** and **Stats** features, can help obtain higher precision as well as higher recall than either alone. In fact, such combinations perform almost as well as using all features, which is the best out of all feature combinations.

Linguistic features of initiator queries To further understand ID initiator queries, we identified the part-of-speech and text features most strongly associated with them, by computing each feature's log-odds ratio (LOR)[3] compared to regular queries. Looking at the top-ranked features by LOR, we found that initiator queries are more likely to use question words (LOR=0.41); focus on proper nouns (0.40) such as places and people; use more 'filler' words (particles) found in natural language (0.27); and when they use general nouns, these tend to be plural (0.13) instead of singular (−0.052). Predominant text features indicated the importance of *list-like* nouns such as *forms, facts, types, ideas* (LOR=1.59, 1.45, 1.25, 0.92); verbs that are commonly used in questions such as *did* (1.34); and words indicating a broad need such as *information* and *manual* (1.64, 1.18). Strong negative features tend to encode exceptions – such as the most negative word *lyrics* (−2.25) used to find words to specific songs.

4.3 Re-ranking for intrinsic diversity

While the previous section discusses the identification of queries that lead to ID tasks, in this section we discuss changes that can be made to the search results page to support queries for ID tasks. Specifically, we propose a re-ranking scheme that looks to satisfy not only the information need of the issued query, but also the future queries that the user is likely to issue later in the session on other aspects of the task. To the best of our knowledge, we are the first to address the problem of jointly satisfying the current query as well as future queries (unlike anticipatory search [107] which focuses solely on the latter).

We will use an interactive ranking-based paradigm here, using an approach

[3]LOR is a rough approximation to the weight in a single-variable logistic regression.

related to the two-level rankings [139] proposed in Chapter 7. Given an issued query representing the start of an ID task, we consider rankings where each result can be **attributed** to some aspect of that task. We represent each aspect of the ID task by a related query of the issued query. One way this could be surfaced on a results page for a user is by placing the related query for an aspect adjacent to its corresponding search result. In such a setting, clicking on the related query could lead to results for that query being presented, thus enabling the user to explore documents for that aspect. This brings us to the main question of how we find such a ranking.

4.3.1 Ranking via Submodular Optimization

We first describe precisely what we consider as an interactive ranking. In response to an initial query q, an interactive ranking $\mathbf{y} = (\mathbf{y}_D, \mathbf{y}_Q)$ comprises two parts: a ranking of documents $\mathbf{y}_D = d_1, d_2, \ldots$, which we refer to as the *primary* ranking; and a corresponding list of related queries $\mathbf{y}_Q = q_1, q_2, \ldots$, which represent the *aspects* associated with the documents of the primary ranking. The i^{th} query in the list, q_i, represents the aspect associated with d_i. Structurally this can also be thought of as a ranked list of (document, related query) pairs (d_i, q_i).

Given this structure, let us consider four conditions that comprise a good interactive ranking:

1. Since the documents in the primary ranking were displayed in response to the issued query q, they should be relevant to q.
2. As document d_i is associated with the aspect represented by the related query q_i, document d_i should be relevant to query q_i.
3. Aspects should be relevant to the ID task being initiated by the query q.

4. At the same time, the aspects should not be repetitive *i.e.,* there should be diversity in the aspects covered.

We now design a ranking objective function that satisfies these four conditions to jointly optimize the selection of documents and queries $(\mathbf{y}_D, \mathbf{y}_Q)$. Suppose we have an existing interactive ranking $\mathbf{y}^{(k-1)}$ that has $k-1$ (document, related query) pairs, and our goal is to construct a new ranking $\mathbf{y}^{(k)}$ by adding an optimal (document, related query) pair to $\mathbf{y}^{(k-1)}$ – an operation we denote by $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} \oplus (d_k, q_k)$.

Condition 1 above can be met by selecting d_k such that $R(d_k|q)$ is large, where $R(d|q)$ denotes the probability of relevance of document d given query q. Condition 2 can be met by selecting d_k such that its relevance to the related query q_k, $R(d_k|q_k)$, is large. Conditions 3 and 4 imply a standard diversification tradeoff, but here we have that the aspects q_k should be related to the initial query q and diverse. If we use a similarity function between queries to estimate the relevance between queries, Condition 3 implies that the similarity function $Sim(q, q_k)$ between q_k and q should be large. Condition 4 requires that the diversity should be maximized between q_k and all previous queries $Q = q_1, \ldots, q_{k-1}$. Both Condition 3 and 4 can be jointly obtained by optimizing an MMR-like *diversity function* [35], $Div(q_k, Q)$, as described below.

Intuitively, we would also like the change in the objective function on adding document-query pair (d_k, q_k) to the ranking $\mathbf{y}$ to be no smaller than what we would gain if adding the pair to a larger ranking $\mathbf{y} \oplus \mathbf{y}'$: that is, the objective function should be *monotone* and *submodular*. Submodular objectives are desirable because they have the property that they can be optimized using a simple and efficient *greedy* algorithm which iteratively computes the next best (d, q) pair

to add to the ranking. Using the greedy algorithm ensures that the computed solution is at least $(1 - \frac{1}{e})$ times as good as the optimal.

We now consider the following objective satisfying the above conditions[4]:

$$\texttt{argmax}_{(d_1,q_1)\cdots(d_n,q_n)} \sum_{i=1}^{n} \gamma_i \cdot R(d_i|q) \cdot R(d_i|q_i) \cdot e^{\beta Div(q_i,Q)}$$

where Q is shorthand for the set of queries $Q = \{q_1, \ldots, q_n\}$, and $Div(\cdot)$ is an MMR-like diversity function defined as

$$\begin{aligned} Div(q_i, Q) = \lambda \cdot Sim(q_i, Snip(q)) \\ - (1 - \lambda) \max_{j<i} Sim(Snip(q_i), Snip(q_j)). \end{aligned} \tag{4.1}$$

Here, $\lambda \in [0, 1]$ and $\beta > 0$ are parameters, where λ controls the tradeoff between related query aspect relevance and diversity while β controls the rate at which returns diminish from additional coverage. Finally, γ_i refers to the discount factor for position i: we use the common $\frac{1}{\log_2(i+1)}$ DCG discounting.

This objective can be interpreted as maximizing an expected utility (the exponential term) of covering related and diverse aspects where the expectation is over the maximum joint relevance of a document to both the initial query and the related query aspect. Furthermore, the joint probability is assumed to be conditionally independent to factor into the two relevance terms.

In this study, we define $Sim(x, y)$ as the cosine similarity between word-TF representations of x and y, and $Snip(q_j)$ is the bag-of-words representation of caption text from the top-10 search results for q_j using relevance score $R(d|q_j)$ alone. The MMR-like term appears within the exponent to ensure the objective is monotone.

[4]We omit the proof of submodularity for space reasons and instead refer the reader to the journal article [135].

Algorithm 4: Greedy-DynRR($\beta, \lambda, P(\cdot|\cdot), q$)

$(\mathbf{y}_D, \mathbf{y}_Q) \leftarrow \phi$
for all $q' \in RelQ(q)$ **do**
    $Next(q') \leftarrow$ Document Ranking by $R(\cdot|q) \cdot R(\cdot|q')$.
for $i = 1 \rightarrow n$ **do**
    $bestU \leftarrow -\infty$
    for all $q' \in RelQ(q) / \mathbf{y}_Q$ **do**
        $d' \leftarrow Top(Next(q') / \mathbf{y}_D)$
        $v \leftarrow R(d'|q) \cdot R(d'|q') \cdot e^{\beta \cdot Div(q', \mathbf{y}_Q)}$
        if $v > bestU$ **then**
            $bestU \leftarrow v$
            $bestQ \leftarrow q'$
            $bestD \leftarrow d'$
    $(\mathbf{y}_D, \mathbf{y}_Q) \leftarrow (\mathbf{y}_D, \mathbf{y}_Q) \oplus (bestD, bestQ)$
return y

Note that while the final objective optimizes for an interactive ranking, the primary ranking itself aims to present results from other aspects. We optimize this using the greedy algorithm presented in Algorithm 4, which we refer to as the **DynRR** method. In Alg. 4, the function $RelQ(q)$ denotes a function that returns related queries for query q, and $Top(\mathbf{y}_D)$ returns the top element in the ranking $\mathbf{y}_D$.

Theorem 8 *The solution returned by Alg 4 is at least* $\frac{e^{-\beta(1-\lambda)}}{2}$ *as good as the optimal.*

As the proof is fairly involved, we refer the interested reader to [135].

Also **note** that while the presented algorithm is fairly simplistic from a learning perspective, the goal of this chapter is to see if modeling such (intrinsically) diverse needs can help improve ranking performance or not. The next chapter (Chapter 5) explores more sophisticated learning for intrinsic diversity.

4.4 Reranking Evaluation

4.4.1 Experimental Setup

Data: To evaluate the efficacy of the proposed reranking method, we used the data obtained from mining the search logs, as described in Section 4.1. We used two main datasets: MINED (8888 Training queries and 2219 Test) and MIXED (4120/1027 Training/Test). To analyze impact when most of the sessions are ID and more complex, the MINED dataset is obtained directly from the filtering algorithm by setting the threshold on the Number of Distinct Aspects to be 5. To determine the re-ranking impact when sessions may be a mixture of both ID and regular sessions, the MIXED dataset was obtained by predicting when a session was ID using the classifier from Sec. 4.2 over a mixture of the MINED dataset sessions and a random sample of regular sessions of the same size. More specifically, the combined sessions were split in a 45-10-45 split of training-validation and test sets. The trained classifier was used to classify the test set sessions as being ID or not, based on the initiator query. The sessions predicted as ID formed the *MIXED* dataset (prediction accuracy of 68.8% over the combined sessions); for those not predicted to be ID, we assume the standard ranking algorithm would be applied and thus relevance would be the same on those. The MIXED dataset is a reflection of an operational setting, where the query issued is used to predict if the resulting session will be an ID session or not, and the ones predicted to be ID are selected for re-ranking.

Query	Length
Website	Log(PageRank)
Baseline Ranker	Reciprocal Rank (if in top 10)
URL	Length, # of Query Terms Covered, Fraction of Query Covered, TF Cosine similarity, LM Score(KLD), Jaccard Similarity, Boolean AND Match, Boolean OR Match
Anchor (Weighted)	Same as URL
Anchor (Unweighted)	TF-Cosine Sim, KLD Score

Table 4.3: The 21 features used to train $R(d|q)$.

Obtaining Probability of Relevance: For our algorithm, we required the computation of the conditional relevance of a document given a query *i.e.*, $R(d|q)$. Thus, to enable easier reproducibility by others, we learned a model using **Boosted Regression Trees**, on a dataset labeled with the relevance-values for query-document pairs with 20,000 queries using graded relevance judgments (~60 documents per query). The features used are given in Table 4.3. Features were all normalized to zero mean, unit variance. To obtain the final model, we optimized for NDCG@5.

Evaluation metrics: To compare against standard ranking techniques, we simply evaluate the quality of the primary ranking, *i.e.*, completely ignore the related query suggestions attributed to documents. Since our goal is *whole-session relevance*, documents are considered *relevant* if and only if they are relevant to any query in the session. Given this notion of relevance, we compute the Precision, MAP, DCG and NDCG values.

Baselines: As baselines we used the following methods:

- **RelDQ**: Ranking obtained by sorting as per $R(d|q)$.
- **Baseline**: A state-of-the-art commercial search engine ranker (also used to compute the rank feature for training the $R(d|q)$ model).

We also computed performance of other baselines, such as MMR and

relevance-based methods such as BM-25 (using the weighted anchor text), but found them to perform far worse than RelDQ and Baseline and hence do not present the results for such other baselines.

Related Queries: We used three different sources for related queries:

- **API**: We used the publicly available API of a commercial search engine (which returns 6-10 related queries)
- **Click-Graph**: Using co-click data, we obtained a set of $10 - 20$ related queries.
- **Co-Session Graph**: Using data of queries co-occurring in the same session, we obtained $10 - 20$ related queries.

To ensure fairness, the graphs were constructed using data prior to April 2012. For some experiments, we only use the second and third (which we distinguish by the suffix **C+S**).

Settings: The parameters for DynRR were set by optimizing for a DCG-like metric on the training data[5]. All numbers reported here are for the test set. We considered all SAT-clicked results in the session as relevant documents; since we compare *relative* to the baseline search engine, the assumption is that placing the SAT-clicked documents higher is better, rather than being an indication of absolute performance. The candidate document set for re-ranking comprises the union of the top 100 results (from the Baseline method) of the initiator query, and the top 10 results from each related query.

[5]We varied the λ parameter from 0 to 1 in increments of 0.1, while the β parameter was varied across the values $\{0.1, 0.3, 1, 3, 10\}$.

Set	Method	Prec			MAP			DCG			NDCG		
		@1	@3	@10	@1	@3	@10	@1	@3	@10	@1	@3	@10
Mined	RelDQ	1.00	0.94	0.97	1.00	0.97	0.98	1.00	0.97	0.99	1.00	0.97	0.99
	DynRR	**1.06**	**1.03**	**1.02**	**1.06**	**1.05**	**1.04**	**1.06**	**1.04**	**1.04**	**1.06**	**1.05**	**1.05**
	DynRR C+S	**1.10**	**1.09**	**1.09**	**1.10**	**1.10**	**1.10**	**1.10**	**1.10**	**1.11**	**1.09**	**1.10**	**1.11**
Mixed	RelDQ	1.00	0.94	0.99	1.00	0.98	0.98	1.00	0.96	0.98	1.00	0.97	0.98
	DynRR	**1.03**	**1.02**	**1.04**	**1.03**	**1.04**	**1.03**	**1.03**	**1.03**	**1.03**	**1.03**	**1.03**	**1.05**

Table 4.4: Performance of different methods (as a ratio compared to the Baseline)

Set	Comp. Metric	% Gains			% Losses		
		0.2	0.5	1.0	0.2	0.5	1.0
Mined	DCG@10	19.6	5.2	0.3	12.7	3.8	0.3
Mixed	DCG@10	17.7	6.0	0.8	12.9	4.0	0.2

Table 4.5: % of sessions for which the metric performance of DynRR differs from the Baseline DCG@10 by more than a certain threshold.

4.4.2 Results

Reranking Evaluation: We first evaluate the quality of the top-level ranking. As seen in the results of Table 4.4, the re-ranking leads to improvements across the different metrics for both datasets. Thus, even without interactivity, the method is able to outperform the baselines in predicting future results of interest to the user, while also providing results for the current query. In particular, we found the DynRR method works best using the **C+S** related queries (which we return to later) with 9-11% gains over the baselines at position 10 across the various metrics with 3-5% relative gains. We also find that the method improves on the MIXED dataset supporting the question of whether the method can be robustly used in practical scenarios. Thus we improve an important segment of tasks while maintaining high levels of performance elsewhere; further improvements to the initiator classification model will improve the robustness further.

Robustness: A key concern when comparing a new method against a baseline, is the robustness of the method. In particular, we are interested in the

number of queries that are either improved or hurt, when switching from the Baseline method to the proposed re-ranking method. This is particularly crucial for the MIXED dataset, as we would want that the performance on non-ID sessions not be severely affected. Table 4.5 displays the % of examples for which the method either gains or loses above a certain threshold, compared to the Baseline method. We see that the number of gains far exceeds the number of losses, especially while comparing the interactive metric. We should also note that for both datasets and both metrics, the DynRR method is statistically significantly better than the Baseline method, as measured by a binomial test at the 99.99% significance level.

4.5 Summary

This chapter studied *intrinsically diverse tasks*, which are tasks that typically require multiple user searches on different aspects of the same information need. It motivated the problem using real-world data and presented an algorithm to mine data from search logs using behavioral interaction signals within a session. It then looked at the problem of identifying the queries that start these sessions, and treated it as a classification problem, and provided an analysis of these queries. Finally, a re-ranking approach was proposed so as to alter the rankings presented to the user. This reranking aimed to provide users with information on aspects of the task which they were highly likely to search for in the future. The approach was validated empirically using search log data, demonstrating significant improvement over competitive baselines.

CHAPTER 5

COACTIVELY LEARNING INTRINSIC DIVERSITY

Modeling the dependencies between items in a ranking of results is one of the most promising directions for improving the quality of retrieval and recommendation systems. First, consider the example of a search engine and an ambiguous query such as "jaguar" or "apple". For such queries, it is important to present a diverse set of results since diversity hedges against uncertainty about the users intent (*i.e., extrinsic diversity*). *Intrinsic diversity* [132] on the other hand, is important to avoid redundancy and provide a set of results that cover multiple aspects of an information need. The previous chapter (Chapter 4) discussed the prevalence and importance of intrinsic diversity in web search and its' role in improving search performance for complex tasks. Intrinsic Diversity (ID) is equally important and prominent in recommendation tasks. For example, consider a user of a news recommendation service (say NY Times). Of all the articles in the NY Times on a given day, he/she only has time to read a small subset. Therefore, even if they are interested in the Greece Debt Crisis, he/she may not want to read exclusively about this one topic, but rather read one article on the topic while also covering other topics of interest.

Unlike the previous chapter (which focused on the prominence of ID and the impact simple reranking can have for ID tasks), this chapter will focus on coming up with learning algorithms using the joint design principle of interactive learning proposed in this dissertation. In particular, it will develop two coactive learning (Part II) algorithms for learning to intrinsically diversify results from implicit user feedback. The key to these algorithms is that they learn both relevance (of items) and the desired amount of diversity from set-valued preference

data. These algorithms now exploit submodularity and the diminishing returns property to make it possible to avoid redundancy and increase novelty. This results in two easy to implement algorithms, that come with theoretical guarantees on the learning quality (despite the fact that submodular models only allow for approximate inference). Their ability to learn the desired amount of diversity based solely on user feedback, without any a priori knowledge of the user's preferences, makes these algorithms particularly attractive for the ID problem. This is also corroborated in empirical studies, which demonstrate the effectiveness of the proposed approaches in learning both relevance and diversity.

5.1 Modeling Relevance and Diversity

We will use the coactive learning model (Part II) for the algorithms in this chapter (and the next one). Let us briefly recap the coactive model using an example of a personalized news reader that users visit on a daily basis. On day t, the user visits the news reader and is presented a suggested list of documents $\mathbf{y}_t = (d_1, d_2, d_3, d_4, d_5, ...)$ from a corpus $\mathbf{x}_t \in \mathcal{X}$ of candidate news articles. The user then interacts with $\mathbf{y}_t$. We assume that the user acts (boundedly) rationally according to an unknown (complex) utility function $U(\mathbf{x}_t, \mathbf{y}_t)$ that models both relevance of the articles as well as their dependencies with other documents in $\mathbf{y}_t$ (*e.g.*, redundancy). For example, while the user may be interested in the Greece debt crisis, they may prefer to not read more than one article related to this issue, even if $\mathbf{y}_t$ contains 5 relevant articles on the topic. The user's actions, say reading articles d_2 and d_4, can be used to construct a ranking that the user would have preferred, say using the move-to-top feedback model (where **user feedback ranking** $\bar{\mathbf{y}}_t = (d_2, d_4, d_1, d_3, d_4, ...)$) or the pairwise feedback model (where $\bar{\mathbf{y}}_t = (d_2, d_1, d_3, d_4, d_5, ...)$). This type of preference feedback over multiple

rounds t is the input for this sequential learning model. We will thus develop learning algorithms with the goal of rankings with utility close to that of the (unknown) optimal ranking $\mathbf{y}_t^* := \arg\max_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{x}_t, \mathbf{y})$.

As per the joint design principle[1], it is critical that appropriate user models be developed in conjunction with the learning algorithm. In particular we need to accurately model the complex behavior displayed by this *diversity-seeking* user via their utility $U(\mathbf{x}, \mathbf{y})$, which captures both relevance as well as interdependencies between documents and the desired amount of diversity. As this relates to metrics for evaluating retrieval performance for a ranking $\mathbf{y}$ for a given $\mathbf{x}$, we start our design of $U(\mathbf{x}, \mathbf{y})$ based on existing retrieval measures.

While traditional IR metrics are oblivious to diversity (*e.g.,* NDCG, Precision), more recent additions account for diversity in some form (*e.g.,* . [4, 133, 157, 176]). We define our hypothesis space based on the family of performance measures proposed in [139] (and detailed in Sec. 7.2), since it subsumes many of these existing measures. These measures exhibit a *diminishing returns* property (*i.e.,* submodularity), which means that the marginal utility of a document is lower if the intents the document is relevant to are already represented in the ranking.

In particular, we model $U(\mathbf{x}, \mathbf{y})$ as a function that is linear in its parameters $\mathbf{w} \in \mathbf{R}^m$ with $\mathbf{w} \geq 0$,[2] but submodular (and non-linear) in a feature map $\phi(\mathbf{x}, \mathbf{y}) \in \mathbf{R}^m$ with $\phi(\mathbf{x}, \mathbf{y}) \geq 0$:

$$U(\mathbf{x}, \mathbf{y}) := \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}). \tag{5.1}$$

[1]Feedback interventions are the third key component of the interactive learning triple. For the sake of simplicity, we ignore discussing these in this chapter and the next as the perturbation based interventions discussed in Part II are equally suitable for the learning algorithms and user model developed in these chapters.

[2]Denotes component-wise non-negativity.

The parameters $\mathbf{w}$ will be learned by the learning algorithm. The feature vector $\phi(\mathbf{x}, \mathbf{y})$ describes the ranking, but for simplicity of exposition we will consider $\mathbf{y}$ to be the set consisting of the top k results that were viewed by the user, not the full ranking[3]. The function $\phi(\mathbf{x}, \mathbf{y})$ generates a feature vector describing the set $\mathbf{y} = \{d_{i_1}, d_{i_2}, ..., d_{i_k}\}$ under context $\mathbf{x} = \{d_1, d_2, ..., d_{|\mathbf{x}|}\}$ in the following manner: We assume that each document d itself is described by a feature vector $\phi(d)$. These feature vectors are aggregated into the feature vector $\phi(\mathbf{x}, \mathbf{y})$ of $\mathbf{y}$ using an aggregation function F. Let $\phi^j(\mathbf{x}, \mathbf{y})$ be the j-th feature of $\phi(\mathbf{x}, \mathbf{y})$ and $\phi^j(d)$ the j-th feature of $\phi(d)$, then

$$\phi^j(\mathbf{x}, \mathbf{y}) = F(\{\phi^j(d_{i_1}), \phi^j(d_{i_2}), ..., \phi^j(d_{i_k})\}). \tag{5.2}$$

Examples of the per-feature aggregation function F are:

Name	$F(A)$	Subsumes
LIN	$F(A) = \sum_{a \in A} a$	Precision, DCG
MAX	$F(A) = \max_{a \in A} a$	Coverage

The MAX variant, but not LIN, encourages diversity in the following way. For example, consider a boolean bag-of-words representation of documents $\phi(d)$. The first document to contain a term t will increase the feature value of t in $\phi(\mathbf{x}, \mathbf{y})$ by 1. The second document to contain t, however, will not cause any increase. This models the redundancy of multiple occurrences of t, as it does not give any benefit to all but the first occurrence of t. Note that multiple aggregation functions F can be stacked into $\phi(\mathbf{x}, \mathbf{y})$, which allows the linear model to select a desired diminishing-returns profile. Note also that our model is not restricted to the F listed above, and can work with any F that is monotone and

[3] A ranking can be viewed as a nested structure of top-k sets, and the greedy algorithm we will later use to compute rankings uniformly optimizes the utility of the sets at any cutoff in the ranking.

Algorithm 5: Greedy-Ranking($\mathbf{w}, \mathbf{x}$)

$\mathbf{y} \leftarrow 0$

for $i = 1$ **to** k **do**

  $bestU \leftarrow -\infty$

  for all $d \in \mathbf{x}/\ \mathbf{y}$ **do**

    if $\mathbf{w}^\top(\mathbf{x}, \mathbf{y} \oplus d) > bestU$ **then**

      $bestU \leftarrow \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y} \oplus d)$

      $best \leftarrow d$

  $\mathbf{y} \leftarrow \mathbf{y} \oplus best$

return $\mathbf{y}$

submodular [139], including less stringent aggregation functions which allows for some redundancy (like square root).

To compute the ranking that maximizes a utility function, *i.e.,* $\mathbf{y} := \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y})$, one can use the simple and efficient Greedy method (Algorithm 5). At each step, the algorithm greedily chooses the document with the highest marginal utility to be added to the ranking. Note that $\mathbf{y} \oplus d$ is used to refer to the operator that appends document d to ranking $\mathbf{y}$. Also note that Algorithm 5 computes the exact utility optimizer $\mathbf{y}_t$ for the modular measure LIN, whereas it finds a $(1 - \frac{1}{e})$ approximate $\mathbf{y}_t$ for any submodular and monotone function F.

5.2 Coactive Learning Algorithms for Intrinsic Diversity

In this section, we present our coactive learning algorithms for intrinsic diversity, including a perceptron style algorithm and a clipped version of it. Sec-

Algorithm 6: Diversifying Perceptron.

Initialize $\mathbf{w}_0 \leftarrow \mathbf{0}$

for $t = 0$ **to** $T - 1$ **do**

Observe $\mathbf{x}_t$

Present $\mathbf{y}_t \leftarrow \mathit{Greedy{-}Ranking}(\mathbf{w}_t, \mathbf{x}_t)$

Obtain feedback $\bar{\mathbf{y}}_t$

Update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$

tion 5.2.2 also provides an exponentiated gradient algorithm. We prove regret bounds for all the proposed algorithms, where regret is given by:

$$REG_T := \frac{1}{T} \sum_{t=0}^{T-1} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right). \tag{5.3}$$

5.2.1 Diversified Perceptron

The **Diversifying Perceptron (DP)**, shown in Algorithm 6 maintains a weight vector $\mathbf{w}_t$ which is initialized to $\mathbf{0}$. At each time step t, DP presents a ranking $\mathbf{y}_t$ from the corpus $\mathbf{x}_t$ using the current weight vector estimate $\mathbf{w}_t$ (computed via Algorithm 5). DP then uses the **user feedback ranking** $\bar{\mathbf{y}}_t$ to update the weight vector $\mathbf{w}_t$ in the direction of $\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$. This algorithm bears strong resemblance to the Preference Perceptron (Algorithm 1) with one key difference. Unlike the Preference Perceptron, the Diversifying Perceptron optimizes a sophisticated submodular utility function, which captures inter-document dependencies, using a greedy algorithm.

Theoretical Analysis

To analyze the learning algorithms in this section, we will use the **α-informative feedback** characterization (not an assumption):

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \geq \alpha \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right) - \xi_t. \tag{5.4}$$

Analogous results can be easily proven for the **expected α-informative feedback** characterization (presented in Eqn. 3.2) as well. The following theorem describes the generalization performance of the Diversified Perceptron:

Theorem 9 *The average regret of the diversified perceptron algorithm can be upper bounded, for any $\mathbf{w} \in \mathbf{R}_+^m$ that defines the utility in Eq. (5.1), as follows:*

$$REG_T \leq \frac{1}{\alpha T} \sum_{t=0}^{T-1} \xi_t + \frac{\beta R \|\mathbf{w}\|}{\alpha} + \frac{\sqrt{2}\sqrt{4-\beta^2} R \|\mathbf{w}\|}{\alpha \sqrt{T}}. \tag{5.5}$$

Here $\frac{1}{\beta+1}$ is the approx. factor of the greedy algorithm with $\beta \leq 2$ and $\|\phi(\mathbf{x}, \mathbf{y})\|_{\ell_2} \leq R$.

The proof is provided in Appendix A.1 along with the other proofs. Note that bound on the worst-case regret is independent of the dimensionality of the feature space, that the regret converges to its asymptote at the rate of $1/\sqrt{T}$ (where T is equal to the number of examples), and that the informativeness α of the feedback enters the bound only linearly. The first term of the bound captures the noise in the feedback.

For the case of modular utility (LIN), $\beta = 0$ and the bound resembles the preference perceptron bound (Thm. 2). For submodular utilities, $\beta = 1/(e+1)$ in the worst case, although it is typically much smaller in practice. When users provide "clean" feedback according to (5.4), the first term in the bound (5.5) vanishes.

Algorithm 7: Clipped Diversifying Perceptron.

Initialize $\mathbf{w}_0 \leftarrow \mathbf{0}$

for $t = 0$ **to** $T - 1$ **do**

Observe $\mathbf{x}_t$

Present $\mathbf{y}_t \leftarrow Greedy{-}Ranking(\mathbf{w}_t, \mathbf{x}_t)$

Obtain feedback $\bar{\mathbf{y}}_t$

Update: $\bar{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$

Clip: $\mathbf{w}^j_{t+1} \leftarrow \max(\bar{\mathbf{w}}^j_{t+1}, 0) \quad \forall 1 \leq j \leq m.$

While the above theorem holds whenever there is a $\frac{1}{\beta+1}$-approximation for finding $\mathbf{y}_t$, there is a caveat. In the case of submodular utility, to ensure that the approximation guarantee holds, all the weights in $\mathbf{w}_t$ must be positive. This can be done by an additional clipping step that modifies each weight of $\mathbf{w}_t$ by clipping it at zero if it is negative. The clipped version of the algorithm is shown in Algorithm 7.

For Algorithm 7, assuming that the utility is also defined using a vector $\mathbf{w}$ which has only non-negative components, we can still give a regret bound similar to Theorem 9.

Corollary 10 *The average regret of the clipped diversified perceptron algorithm can be upper bounded, for any $\mathbf{w} \in \mathbf{R}^m_+$ that defines the utility, as follows:*

$$REG_T \leq \frac{1}{\alpha T} \sum_{t=0}^{T-1} \xi_t + \frac{\beta R \|\mathbf{w}\|}{\alpha} + \frac{\sqrt{2}\sqrt{4-\beta^2} R \|\mathbf{w}\|}{\alpha \sqrt{T}}, \tag{5.6}$$

where $\frac{1}{\beta+1}$ is the greedy algorithm approximation factor ($\beta \leq 2$) and $\|\phi(\mathbf{x}, \mathbf{y})\| \leq R$.

We obtained the clipped version of the algorithm to avoid non-negative weights. In the next sub-section, we provide an elegant exponentiated algo-

rithm that naturally maintains non-negative weights.

5.2.2 Exponentiated Algorithm

Our exponentiated algorithm for learning to diversify from implicit feedback is shown in Algorithm 8. In this algorithm, the weights are initialized uniformly at the start. There is a rate θ associated with each step. The rate depends on the maximum ℓ_∞ norm of the feature vectors (*i.e.*, $\|\phi(\cdot,\cdot)\|_{\ell_\infty} \leq S$) and time horizon T.

At each step, a context $\mathbf{x}_t$ is observed and an object $\mathbf{y}_t$ is presented just like in the earlier algorithms. However, once the feedback $\bar{\mathbf{y}}_t$ is obtained, the update rules are multiplicative as shown in Algorithm 8. The weights are normalized to one and the steps of the algorithm repeat. Since the updates are multiplicative and the weights are initially positive, $\mathbf{w}_t$ is guaranteed to remain positive in this algorithm.

We now prove the regret bound for Algorithm 8. While the regret bounds for Algorithms 6 and 7 depended on the ℓ_2 norm of the features, and the ℓ_2 norm of $\mathbf{w}$, the bound for the exponentiated algorithm depends on the ℓ_∞ norm of the feature vectors and the ℓ_1 norm of $\mathbf{w}$.

Theorem 11 *For any $\mathbf{w} \in \mathbf{R}_+^m$ such that $\|\mathbf{w}\|_{\ell_1} = 1$, the average regret of the exponentiated algorithm can be upper bounded as follows:*

$$REG_T \leq \frac{1}{\alpha T}\sum_{t=0}^{T-1}\xi_t + \frac{S\beta}{\alpha} + \frac{2\log(m)S}{\alpha\sqrt{T}} + \frac{S}{2\alpha\sqrt{T}}, \tag{5.7}$$

where $\frac{1}{\beta+1}$ is the approximation factor of the greedy algorithm with $\beta \leq 2$ and $\|\phi(\mathbf{x},\mathbf{y})\|_{\ell_\infty} \leq S$.

Like the previous bounds, Theorem 11 also bounds the regret in terms of the

Algorithm 8: Exponentiated Diversifying Algorithm.

Initialize $\mathbf{w}_0^i \leftarrow \frac{1}{m} \;\; \forall 1 \leq i \leq m$.

$\theta \leftarrow \frac{1}{2S\sqrt{T}}$

for $t = 0$ **to** $T - 1$ **do**

Observe $\mathbf{x}_t$

Present $\mathbf{y}_t \leftarrow Greedy - Ranking(\mathbf{w}_t, \mathbf{x}_t)$

Obtain feedback $\bar{\mathbf{y}}_t$

Update: $\mathbf{w}_{t+1}^i \leftarrow \mathbf{w}_t^i \exp(\theta(\phi^i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi^i(\mathbf{x}_t, \mathbf{y}_t)))/Z_t$ where Z_t is such that the weights add to one.

noise in the feedback (first term), the approximation factor of the inference algorithm (second term), and additional terms which converge to zero at the rate $O(1/\sqrt{T})$. The key difference to the previous bounds is that the regret bound of the exponentiated algorithm scales logarithmically with the number of features, and with the ℓ_1-norm of $\mathbf{w}$, which can be advantageous if the optimal $\mathbf{w}$ is sparse.

5.3 Empirical Study

In this section we empirically study different aspects of our proposed algorithms. In particular, we show how using the submodular utility helps achieve diversity. Furthermore, we explore the robustness of our learning method under degraded feedback quality and noise. We also explore learning the *amount* of diversity a user wants and compare the three algorithms that we proposed in this paper against each other.

5.3.1 Experiment Setup

Since there is no large publicly available real-world corpus containing intrinsic diversity judgments[4], we created two artificial datasets from the RCV-1 [104] text corpus and from the 20 newsgroups dataset (abbreviated 20NG).

The RCV-1 corpus contains over 800k documents, each of which is annotated as belonging to one or more of 100+ *topics*. While the original RCV-1 topics are arranged hierarchically, to make the problem non-trivial, we considered only topics from the second level. The 20NG dataset contains about 19k documents (with duplicates removed) with a single class label for each document. We simulate users with multiple different interests, by forming *super-users* with 5 different interests corresponding to 5 different topics/classes. Thus, if a document is relevant to any of these topics it is relevant to that super-user, else it is not. We assume that all topics are equally important unless otherwise mentioned. In addition, for a given *super-user* we removed documents relevant to multiple interests. In this manner, producing a diverse set of results would require being able to truly learn each of the interests separately.

We ran the Diversifying Perceptron algorithm with a fresh set of 1000 documents for RCV1 (100 for *20NG*) in each step as the corpus $\mathbf{x}$ and presented a ranking $\mathbf{y}$ from the current corpus. In particular we focus on the top 5 results for all evaluation measures for brevity, though the trends reported in the following hold true for other ranking lengths as well. All results we report are averaged over 50 runs of the algorithm, each for a different *super-user*. Documents are represented as TF-IDF vectors. The joint feature map $\phi(\mathbf{x}, \mathbf{y})$ is an aggregation

[4]Corpora like the TREC WEB corpus are small and contain relevance judgments only for extrinsic diversity.

of the document vectors using one (or multiple) of the aggregation functions F described in Section 5.1.

5.3.2 Can the algorithm learn to diversify?

We first evaluate if the proposed DP algorithm is really able to learn a function that combines relevance and diversity. In particular, we generated users with 5 different and disjoint interests, and each user wants to read exactly one document relevant to each interest in every iteration. Note that users of this type are seeking maximum diversity in their rankings. To illustrate the performance of the algorithm, we report two quantities. First, we computed how many interests are covered in the top 5 documents of the presented ranking in each iteration. Second, we considered the *median* depth the user needs to search down the ranking to find one document for each of his/her interests.

We ran the DP algorithm with the **MAX** feature map as defined in Section 5.1. This is compared against the conventional Preference Perceptron algorithm, that is effectively utilizing the **LIN** feature map, as it focuses purely on relevance and therefore cannot model diversity directly. For simplicity we simulate $\alpha = 1$ informative feedback. We also compare against a **Random** baseline, which is the performance of a random ranking.

Figure 5.1 shows the average and standard error of the results for this experiment on the two datasets. The left column shows the number of intents covered in the top 5 positions over time. While the LIN method is far better than the Random method and continues to improve over time, it is outperformed by the MAX method, which is able to learn better.

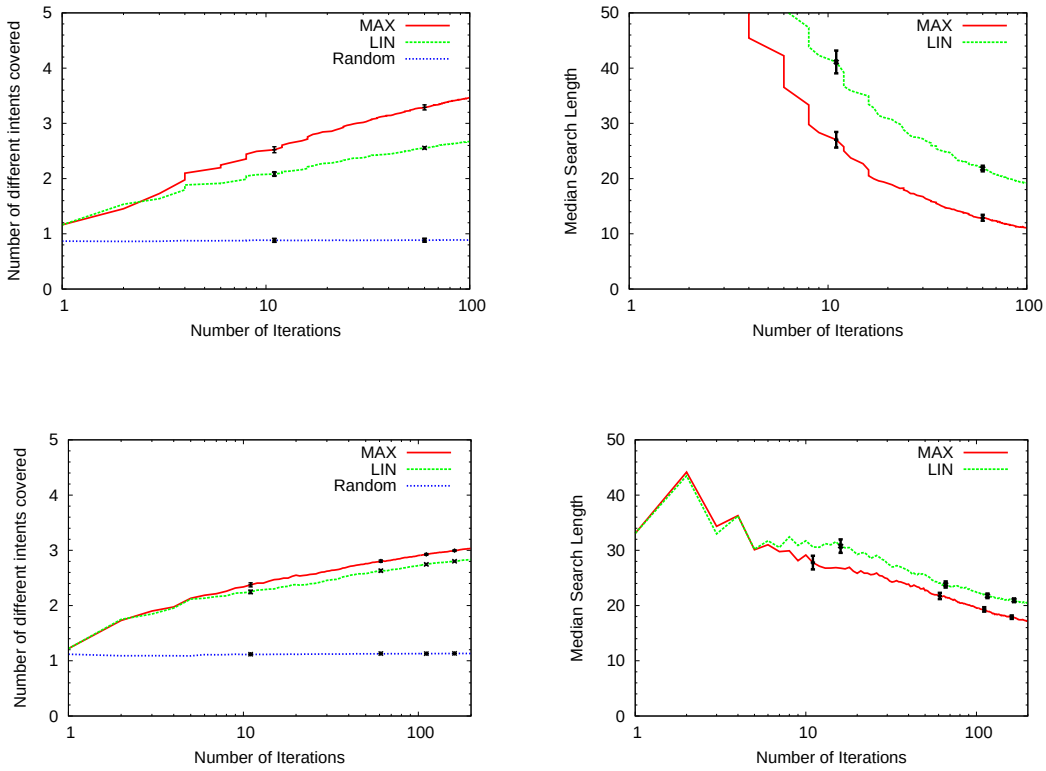

Figure 5.1: Comparison between the submodular (MAX) and independent (LIN) model for users that are purely seeking diversity; top: RCV-1, bottom: 20NG.

The right pane further illustrates this result, as it shows how the median search length (required to find at least one document for each intent) starts at high values, but quickly drops after a few iterations. Both learning methods clearly outperform the Random baseline (whose median length value is too large to plot). In all the plots, the standard errors are small implying statistical significance.

It can be observed that the difference between the MAX and the LIN is much higher in the case of RCV-1 compared to 20NG dataset. This is due to the fact that 20NG has only 20 categories, whereas RCV-1 has more than 100 and is thus much harder to learn for LIN.

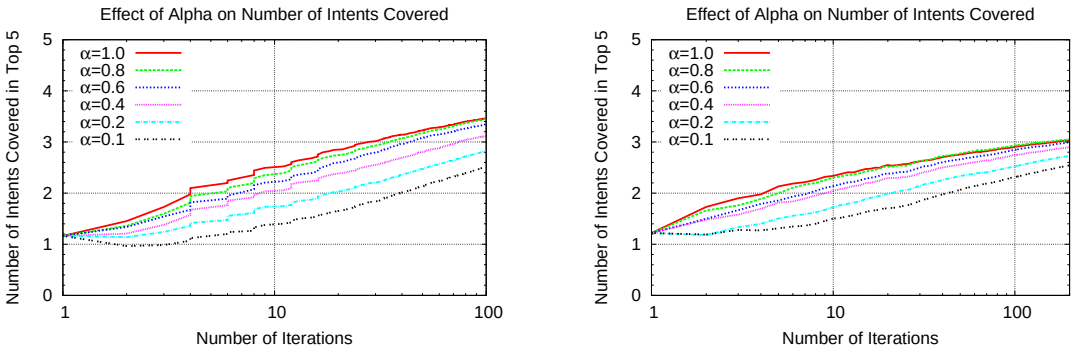

Figure 5.2: Effect of α on performance of the algorithm for users that are purely seeking diversity; left: RCV-1, right: 20NG.

5.3.3 What is the effect of feedback quality?

We next study the effect of the quality of feedback (as described by α) on the performance of the DP method. As real-world users are unlikely to provide perfect feedback, we would like our algorithm to learn even in scenarios where the user-feedback is far from ideal. We varied the quality of the feedback by changing the value of α. A change in α is achieved through the following mechanism: for any intent not covered in the presented ranking, but covered in the optimal ranking, with probability $1-\alpha$, documents covering that intent are absent in the feedback ranking. This leads to expected α-informative feedback.

Figure 5.2 shows the results for this experiment. Most notably, the performance is nearly unchanged for larger values of α. In particular, we find that for $\alpha \geq 0.6$ the performance is very close to that with perfect feedback ($\alpha = 1.0$). At low values of α such as 0.2 or 0.1, the method still makes reasonable progress over time, albeit at a slower rate. We see that for $\alpha = 0.2$ within 100 iterations the number of intents covered more than doubles. These results indicate that the proposed method is still able to learn even when the informativeness of the user feedback is poor.

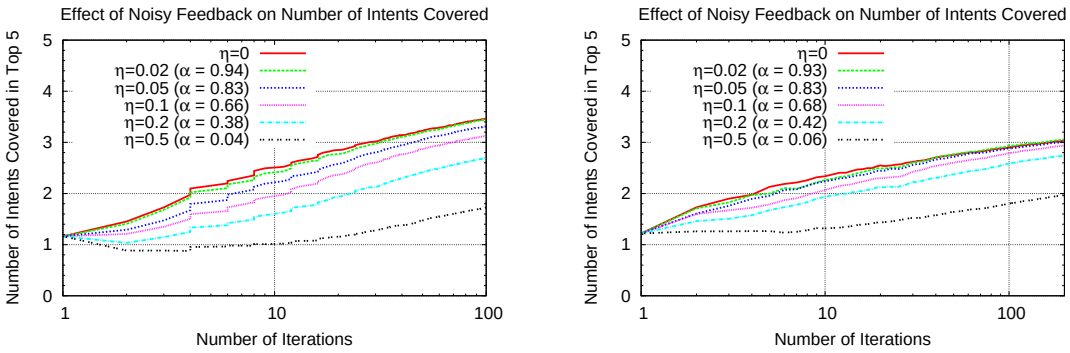

Figure 5.3: Effect of η on performance of the algorithm for users that are purely seeking diversity (number in bracket indicates the effective α of the feedback); left: RCV-1, right: 20NG.

5.3.4 What is the robustness to noise?

While the experiments in the previous section showed robustness to imperfect feedback, we now test the robustness of our algorithm to noisy feedback. One key difference between the two is that with noisy feedback, the user may return a feedback ranking that is worse than the one he was presented. Such a degradation in the quality of the ranking will be captured by the slack variable seen in Eq. (5.4). We would particularly like the noise introduced to be reflective of that expected in the real-world, where users may sometimes be unsure of the relevance of some documents. Thus we modify the user clicking mechanism that produces the feedback in the following manner:

- Each irrelevant document encountered in the ranking may be considered as relevant with probability η.
- Documents relevant to one of the user's topics may be confused for a different topic with probability $\eta/5$.

Like α, η affects only the quality of the user feedback and not the learning algorithm itself. Figure 5.3 shows the effect of varying the noise factor η. As seen in the figure, the algorithm is quite robust to noise. For high values of η, such as 0.2, we find that the algorithm is still able to learn quite well. The figures also indicate the expected α of the feedback received after adding noise. However, note that in this scenario, unlike the experiments varying α, the feedback ranking can be significantly worse than the predicted ranking. Thus we see that for $\eta = 0.2$, although $\alpha \sim 0.4$ in expectation, the performance is noticeably worse than for the case of $\alpha = 0.4$.

5.3.5 Learn the desired amount of diversity?

We next explore whether the algorithm can learn how much diversity the user wants. Furthermore, it is interesting to know how the algorithm performs in settings where the utility that the user optimizes (to provide feedback) is different from the one the algorithm uses.

To study this question, we experimented with the MAX and LIN utility functions mentioned earlier. We varied the user's inherent utility as well as the algorithm's utility to either of these two values. We also experimented with a *combination* method for the DP algorithm, which simply takes the joint feature vector representations used in the MAX and LIN functions and appends them to form a single vector. We refer to this method as $MAX + LIN$. To ensure difference in feedback between the two user utility functions, we weight the different intents (as done in [176]), which results in the utility being higher if a more *popular* topic is covered instead of a less popular one. We ran the DP algorithm for 100 iterations, where at each iteration the feedback provided by the user is as

		User-Utility	
		LIN	*MAX*
	RANDOM	.862(±.007)	.756(±.016)
Algo-Util	*LIN*	.137(±.019)	.447(±.005)
	MAX	.169(±.020)	.274(±.011)
	LIN + MAX	.158(±.021)	.310(±.010)

Table 5.1: Average Regret for different user and algorithm utility functions.

per the utility they optimize. We report performance in terms of the the average regret over these 100 iterations of the user's utility measure (since that is what the true $\mathbf{w}$ captures), thus *lower the better*.

Table 5.1 shows the results for RCV1[5]. First, consider the cases where the algorithm *is given the user's true diversity profile*. As expected, the algorithm performs very well, as seen in the case of the LIN-maximizing algorithm performing best for purely-relevance seeking users (and similarly for the MAX-maximizing algorithm and diversity-seeking users). However, an important result of the experiment is that even when the amount of diversity the user requires is unknown, the *combination* algorithm is able to learn the amount of diversity the user wants. It performs nearly as well as the case where the user's diversity needs are known, as can be seen in the last row of the table. This shows that the combination algorithm is able to learn the tradeoff between relevance and diversity that the user is looking for. This is very encouraging as it allows for the method to be used in scenarios where there is no a priori information about the desired amount of diversity. Compared to recent extrinsic diversification methods such as [149], our method is an online learning technique that utilizes much weaker feedback (that is far more plentiful and cost-effective) than methods in [149] do.

[5]We observe similar results for 20NG but omitted it due to space limitations

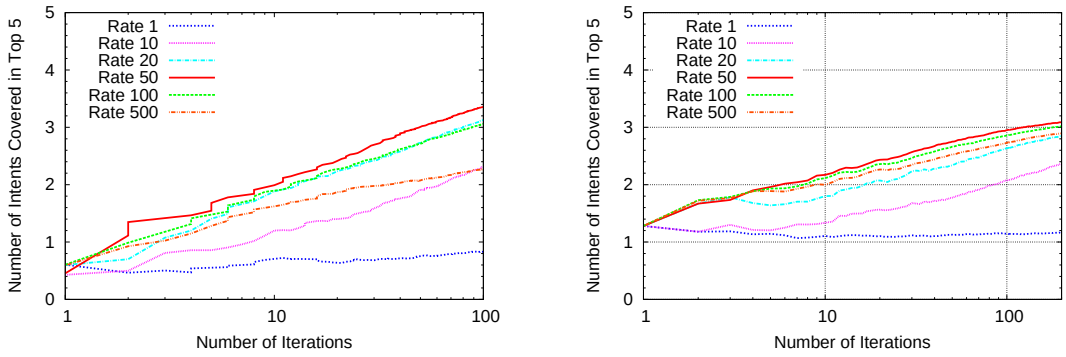

Figure 5.4: Exponentiated algorithm with different rates; left: RCV-1, right: 20NG.

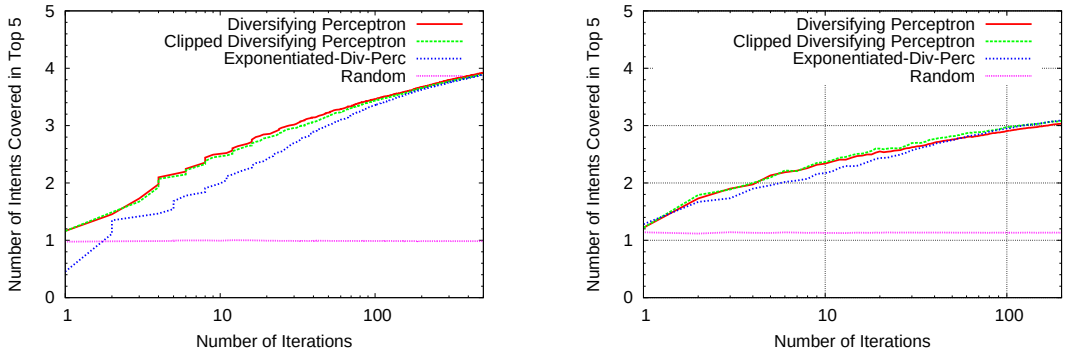

Figure 5.5: Comparison of the three algorithms; left: RCV-1, right: 20NG.

5.3.6 Exponentiated algorithm

Compared to the other two algorithms, the exponentiated algorithm has a rate θ associated with it. This rate needs to be set appropriately. In practice, we observed that the performance of the exponentiated algorithm is sensitive to the value of the rate. In particular, we multiplied the rate θ by a numerical value and studied how the algorithm behaved. Note that this effectively changes the radius of the data, but seemed to significantly affect the behavior of the exponentiated algorithm. The results of this experiment is shown in Figure 5.4. The performance of the algorithm first improves and then deteriorates as the rate factor increases.

5.3.7 How do the three algorithms compare?

We proposed three algorithms to learn diversity from implicit feedback. In this section, we study whether there is a difference in performance of these three algorithms. The clipped DP (Algorithm 7) was proposed mainly due to theoretical considerations. To compare the three algorithms, we followed the same setup as in Section 5.3.2. For the exponentiated algorithm, we considered the best rate parameter from the previous experiment. The results for this experiment are shown in Figure 5.5. It can be seen that there is not much of a difference between the clipped and the non-clipped algorithms in the case of RCV-1. In the case of 20NG, there is hardly any difference between the three algorithms. Even though restricting weights to positive values is required for theoretical purposes, in practice it does not seem to make much of a difference on these two datasets.

5.4 Summary

This chapter explored the use of coactive learning algorithms for learning diversity in rankings. Using the joint design principle, the proposed algorithms when used in conjunction with a sophisticated submodular diversity-seeking user model, are able to learn rankings that balance diversity and relevance. The resulting algorithms learn to optimize the user's utility, using only the implicit set-valued preference feedback from users. In addition to theoretically characterizing the performance of the algorithms and their robustness to noise, the algorithms were found to perform well in empirical studies.

CHAPTER 6

LEARNING EXTRINSIC DIVERSITY FROM USER INTERACTIONS

Many information systems serve a diverse population of users who have conflicting preferences. This poses the challenge of maximizing collective user satisfaction over a distribution of conflicting needs. A typical example is the problem of search result diversification (Sec 2.4.2). For an ambiguous query such as `apple`, a diversified set of results should ideally provide some relevant results for each of the different query intents. Similar challenges also arise in an online store that wants to appeal to a range of customers with different tastes, or in a movie recommendation system where even a single user may have different preferences (*e.g.,* moods, viewing companions) on different days. Unlike the previous two chapters that studied problems where diversity is *intrinsic* to the need of the users, the "diversification" problem studied here is *extrinsic* to any single user's need but instead necessary to hedge against uncertainty about the user's preferences.

Prior work on this problem has generally found learning based methods [99, 149, 176] to outperform manually tuned methods [35, 43]. Unfortunately, the practical use of these learning methods is rather limited, since they all require non-trivial amounts of expert annotated training data that explicitly lists all facets of an information need (*e.g.,* the different moods a user can be in).

The use of implicit feedback from user interactions (*e.g.,* clicks) has the potential to overcome this data bottleneck. Not only is it available in abundance, but it also directly reflects the users' – not the experts' – preferences. However, the challenge here is that the learning algorithm no longer gets (expert constructed) examples of socially optimal results. Furthermore, unlike the task

of intrinsic diversity which was studied in the previous chapter, the users all behave *egoistically*. In other words, user interactions are geared towards improving their own utility, which may be a contradictory goal to the one of the system (*i.e.*, optimize utility for the entire user population). For example, given the query `apple`, users may click on results about the company **or** about the fruit, but rarely both. Thus, the challenge here for the interactive learning algorithm here is to learn to construct a socially optimal compromise from the egoistic actions of the users.

This chapter investigates problem of learning socially optimal rankings using the coactive learning framework discussed in Part II. It provides two new coactive learning algorithms for the extrinsic diversification problem. Guided by the joint design principle of interactive learning, these algorithms were designed in conjunction with suitable user utility models. In particular, submodular utility models were used to capture the differing needs of the user population, as they naturally lead to diverse result sets.

After characterizing the informativeness and noisiness of the implicit feedback, the proposed algorithms are also analyzed theoretically with bounds provided on the regret of the algorithms in terms of the social utility – which is the expected utility over the user distribution. These are also accompanied by empirical studies for single query diversification tasks, which show the resulting algorithms to be able to learn rapidly as compared to existing work. More significantly, experiments on the cross-query diversification task, find the proposed algorithms to be the first-known methods to robustly learn to compose rankings with an appropriate amount of diversity, using only implicit feedback.

6.1 Learning Problem and Model

Let's start with an example to motivate the formalization of the learning problem considered in this paper. Suppose we have a search engine that receives an ambiguous query (*e.g.,* `jaguar`). Say there are three user populations that each have different intents and thus consider documents differently, with regards to query relevance (as detailed in Fig 6.1). The user populations have different sizes, and Fig. 6.1 lists the probability of each type. Note that the search engine has no way of identifying which type of user issued the query (*i.e.,* the search engine does not know whether "jaguar" refers to the cat or the car for any specific user). Suppose the utility of a ranking R to users of type i is $U_i(R) = \sqrt{\#\text{of rel docs in top 4 of R}}$. This means it is beneficial to show at least one relevant document. Furthermore, the marginal utility of showing additional relevant documents is sub-linear.

User Type	Prob.	Relevant docs
1	0.5	$a_1, a_2, a_3, \ldots$
2	0.25	$b_1, b_2, b_3, \ldots$
3	0.25	$c_1, c_2, c_3, \ldots$

Figure 6.1: Illustrative example showing different user preferences.

Now consider two possible rankings that the search engine could show.

- $R_1 = (a_1, a_2, a_3, a_4)$: While ideal for the predominant users (*i.e.,* type 1 users get utility $U_1 = 2$), it provides no value for the other users (utility $U_2 = U_3 = 0$). Thus in expectation, this ranking has expected utility of $\mathbf{E}[U] = 1$.
- $R_2 = (a_1, b_1, c_1, a_2)$: This ranking provides some relevant documents for all user types ($U_1 \sim 1.4; U_2 = 1; U_3 = 1$), maximizing the collective user satisfaction with $\mathbf{E}[U] \sim 1.2$.

Our goal in this chapter is to find rankings of the latter type, which we call *socially optimal* since they maximize expected utility (*i.e., social utility*).

To avoid relying on expensive expert-annotated data, we would like to learn these diverse rankings using the implicit feedback from users. Consider, for example, a user of type 1 that chooses to click/read relevant documents a_1, a_2 from the presented ranking $\mathbf{y}_t = (b_1, c_1, b_2, a_1, c_2, a_2)$. These actions reveal information about the user's utility functions which we can exploit to construct a feedback ranking $\bar{\mathbf{y}}_t$, say $(b_1, c_1, a_1, b_2, a_2, c_2)$, that has higher utility for that user (or at least not worse utility) *i.e.,* $U_1(\bar{\mathbf{y}}_t) \geq U_1(\mathbf{y}_t)$.

The key challenge in learning socially optimal rankings from the feedback of individual users lies in resolving the contradicting feedback from different user types. Each user's feedback reflects only their own utility, not social utility. For example, even if presented with the socially optimal ranking R_2, users may provide feedback indicating a preference for a different ranking (*e.g.,* type 1 users may indicate their preference for R_1). Thus, a successful learning algorithm for this problem should be able to reconcile such differences in preference and display stability despite the *egoistic* feedback.

6.1.1 Learning Problem

We now define the learning problem and user-interaction model more formally. We assume there are N types of users, each associated with a probability p_i according to which individual users accessing the system are sampled. Given a context $\mathbf{x}_t$ (*e.g.,* query), the **personal utility** of an object (*e.g.,* ranking) $\mathbf{y}_t$ for users of type i is $U_i(\mathbf{x}_t, \mathbf{y}_t)$. The **social utility** $U(\mathbf{x}_t, \mathbf{y}_t)$ is defined as the expected utility over the user distribution.

$$U(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{E}[U_i(\mathbf{x}_t, \mathbf{y}_t)] = \sum_{i=1}^{N} p_i U_i(\mathbf{x}_t, \mathbf{y}_t) \tag{6.1}$$

The optimal object for context $\mathbf{x}_t$ and user type i is denoted as

$$\mathbf{y}_t^{*,i} := \arg\max_{\mathbf{y}_t \in \mathcal{Y}} U_i(\mathbf{x}_t, \mathbf{y}_t). \tag{6.2}$$

The socially optimal object for context $\mathbf{x}_t$ is denoted as

$$\mathbf{y}_t^{*} := \arg\max_{\mathbf{y}_t \in \mathcal{Y}} U(\mathbf{x}_t, \mathbf{y}_t). \tag{6.3}$$

Users interact with the system like in the standard coactive learning model, but it is no longer assumed that all users act according to a single utility function. Specifically, at each timestep t the system receives a context $\mathbf{x}_t$ and a user type i is sampled from the user distribution. In response, the system presents the user with an object $\mathbf{y}_t$ for which the user draws utility $U_i(\mathbf{x}_t, \mathbf{y}_t)$. The algorithm then observes (implicit) feedback from the user (who acts according to U_i), updates its model, and repeats. The goal of the algorithm is to present objects as close to the social optimal $\mathbf{y}_t^*$ *i.e.,* minimize regret of the learning process:

$$REG_T := \frac{1}{T} \sum_{t=0}^{T-1} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right). \tag{6.4}$$

User Feedback Characterization To make meaningful theoretical arguments about the approaches to this problem, we need a suitable user feedback characterization. However, unlike the coactive learning problems studied in the earlier parts of this dissertation, in this problem the users do not all provide feedback from a single global utility function (that directly reflects social utility). Instead, users valuate and provide feedback according to their own personal utility. Thus, we use an alternate characterization of the feedback quality:

Definition 12 *User feedback is* **expected** α_i, δ_i**-informative** *for a presented object* $\mathbf{y}_t$ *under context* $\mathbf{x}_t$ *for a user with personal utility function* U_i*, if* $\bar{\xi}_t \in \mathfrak{R}$ *is chosen such that for some given* $\alpha_i \in [0, 1]$ *and* $\delta_i > 0$

$$\mathbf{E}_{\bar{\mathbf{y}}_t}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t)] \geq (1 + \delta_i)U_i(\mathbf{x}_t, \mathbf{y}_t) + \alpha_i \left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) - \bar{\xi}_t.$$

holds. Note that the expectation is over the user feedback.

The expected α_i, δ_i-informative criterion states that the user's feedback object $\bar{\mathbf{y}}_t$ has better personal utility than the presented object $\mathbf{y}_t$ on average. More precisely, the first term on the right-hand side implies that the improvement should be at least by a factor of $(1 + \delta_i)$. Note, though, that this condition is based only on the personal utility of the specific user, not the social utility. The second term on the right-hand side further prescribes that personal utility increases proportional to how far $\mathbf{y}_t$ is away from the optimal object $\mathbf{y}_t^{*,i}$, and the factor $\alpha_i \in [0, 1]$ describes the informativeness of the feedback. This second term captures that it is easier to make large improvements in utility when the presented $\mathbf{y}_t$ is far from optimal for this user. Finally, as it would be unreasonable to assume that user feedback is always strictly α_i, δ_i-informative, $\bar{\xi}_t$ captures the amount of violation.

6.1.2 Submodular Utility Model

The following defines the class of utility function we consider for modeling users. As done in the previous chapters, we will assume that the utility functions $U_i(\mathbf{x}_t, \mathbf{y}_t)$ is linear in its parameters $\mathbf{v}_i \in \mathbf{R}^m$.

$$U_i(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{v}_i^\top \phi_F(\mathbf{x}_t, \mathbf{y}_t) \tag{6.5}$$

$\phi_F(\mathbf{x}_t, \mathbf{y}_t)$ is a feature vector representation of the context-object pair and F is a submodular function as further elaborated on below. We require that all $\mathbf{v}_i$'s and $\phi_F(\mathbf{x}_t, \mathbf{y}_t)$'s are component-wise non-negative. The linear model implies that one can write the *social utility* as

$$U(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{w}_*^\top \phi_F(\mathbf{x}_t, \mathbf{y}_t), \text{ where } \mathbf{w}_* = \sum_{i=1}^{N} p_i \mathbf{v}_i. \tag{6.6}$$

We model $\phi_F(\mathbf{x}_t, \mathbf{y}_t)$ using a submodular aggregation of its components, as done similarly in other parts of this thesis for modeling diversity. To simplify the exposition, we focus on rankings as objects $\mathbf{y}$, but analogous constructions also work for other types of objects. Given context $\mathbf{x}$, each document in ranking $\mathbf{y} = (d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ has a feature representation given by $\phi(\mathbf{x}, d_{i_j}) \in \mathbf{R}^m$. We then obtain the overall feature vector $\phi_F(\mathbf{x}, \mathbf{y})$ as

$$\phi_F^j(\mathbf{x}, \mathbf{y}) = F(\gamma_1 \phi^j(\mathbf{x}, d_{i_1}), \gamma_2 \phi^j(\mathbf{x}, d_{i_2}), \ldots\ldots, \gamma_n \phi^j(\mathbf{x}, d_{i_n})) \tag{6.7}$$

where $\phi^j(\mathbf{x}, d)$ and $\phi_F^j(\mathbf{x}, \mathbf{y})$ represent the j^{th} feature in the vectors $\phi(\mathbf{x}, d)$ and $\phi_F(\mathbf{x}, \mathbf{y})$ respectively. The $\gamma_1 \geq \ldots \geq \gamma_j \geq \ldots \geq \gamma_n \geq 0$ represent position-discounting factors, as they determine how important each position in the ranking is. For instance, the submodular DCG metric proposed in [139] (and detailed in the next chapter), sets the discount factors to be $\gamma_i = \frac{1}{\log_2(1+i)}$. Furthermore, the choice of aggregation function F determines the diminishing returns profile of the users utility. For example, using a coverage-like aggregation function $F(A) = \max_{a \in A} a$, strongly promotes diversity, since a single document can already maximize utility. On the other extreme lies the additive aggregation function $F(A) = \sum_{a \in A} a$, which leads to a diversity-agnostic (*i.e.,* modular) feature vector. More generally, any monotone increasing and concave function of $\sum_{a \in A} a$ can be used. As shown in the next chapter, this modeling allows us to capture a broad class of performance measures, including many common IR performance metrics (*e.g.,* NDCG, Precision, Coverage).

For a component-wise non-negative vector $\mathbf{w}$, we can compute a ranking that approximately maximizes the utility function, *i.e.,* $\mathbf{y} := \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \phi_F(\mathbf{x}, \mathbf{y})$, using the Greedy-Ranking method (Alg. 5). Despite its simplicity the algorithm, which works by iteratively adding the document with the highest marginal utility to the ranking, has good approximation properties for this NP-hard problem.

Lemma 13 *For* $\mathbf{w} \geq 0$ *and monotone, concave* $F : \mathbf{R}^n_{\geq 0} \to \mathbf{R}_{\geq 0}$ *that commutes in all arguments, Algorithm 5 produces a ranking that is a* β_{gr}*-approximate solution, with* $\beta_{gr} = \left(1 - \frac{1}{e}\right)$ *if* $\gamma_1 = \cdots = \gamma_k$ *or* $\beta_{gr} = 1/2$ *otherwise.*

6.2 Social Learning Algorithms

In this section, we present two coactive learning algorithms for predicting rankings that optimize social utility. The first considers rankings with discount factors for each rank while the second considers the special case of evaluating the top k results as a set. For both algorithms, we characterize their regret by providing upper bounds.

6.2.1 Social Perceptron for Rankings (SoPer-R)

Following the utility model introduced in Section 6.1.2, we now present an algorithm for learning rankings $\mathbf{y} = (d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ that aims to optimize social utility.

The Social Perceptron for Rankings (SoPer-R) is detailed in Algorithm 9. It applies to any F that satisfies the conditions of Lemma 13. The algorithm maintains a weight vector $\mathbf{w}_t$, which is an estimate of $\mathbf{w}_*$. For the given context $\mathbf{x}_t$, the algorithm first computes ranking $\mathbf{y}_t$ using the greedy Algorithm 5, which is then presented to the user. The user actions (*e.g.*, clicks) are observed and used to construct the feedback as follows. The ranking is first partitioned into adjacent pairs by randomly selecting an odd or even grouping. The feedback ranking $\bar{\mathbf{y}}_t$ is constructed by swapping the documents whenever the user clicks on the lower element of the pair. This relates to the idea of FairPairs [131], which is used to help de-bias click data. Note that feedback is only generated when-

Algorithm 9: Social Perceptron for Ranking (SoPer-R)

Initialize $\mathbf{w}_0 \leftarrow \mathbf{0}$
for $t = 0$ **to** $T - 1$ **do**
    Observe $\mathbf{x}_t$
    Present $\mathbf{y}_t \leftarrow \mathit{GreedyRanking}(\mathbf{w}_t, \mathbf{x}_t)$ ▷ Present argmax ranking
    Observe user clicks $\mathcal{D}$ ▷ Get User Feedback
    Construct feedback $\bar{\mathbf{y}}_t \leftarrow \mathit{ListFeedback}(\mathbf{y}_t, \mathcal{D})$ ▷ Create Feedback Object
    Update: $\bar{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$ ▷ Perceptron Update
    Clip: $\mathbf{w}^j_{t+1} \leftarrow \max(\bar{\mathbf{w}}^j_{t+1}, 0) \quad \forall 1 \leq j \leq m.$

Function ListFeedback$(\mathbf{y}, \mathcal{D})$ ▷ $\mathbf{y}$: Presented Ranking; $\mathcal{D}$: User clicks
$\bar{\mathbf{y}} \leftarrow \mathbf{y}$ ▷ Initialize with presented object
With probability 0.5: $\mathcal{PR} \leftarrow (\{1,2\},\{3,4\},\{5,6\}\cdots)$
else: $\mathcal{PR} \leftarrow (\{1\},\{2,3\},\{4,5\},\{6,7\}\cdots)$
for $i = 0 \cdots len(\mathcal{PR})$ **do**
    $\{j_{upper}, j_{lower}\} \leftarrow \mathcal{PR}[i]$ ▷ Get Pair
    if $\mathbf{y}[j_{lower}] \in \mathcal{D}$ AND $\mathbf{y}[j_{upper}] \notin \mathcal{D}$ **then**
        Swap($\bar{\mathbf{y}}[j_{upper}], \bar{\mathbf{y}}[j_{lower}]$) ▷ Place clicked doc above the other doc
return $\bar{\mathbf{y}}$

ever the lower elements was clicked but not the upper, otherwise $\bar{\mathbf{y}}_t := \mathbf{y}_t$. After the feedback $\bar{\mathbf{y}}_t$ is received, the algorithm performs a perceptron-style update to the weight vector. To ensure that the weight vector contains only non-negative weights, any negative weights are *clipped* to zero.

The observant reader may recognize the similarities with the 3PR algorithm

from Sec. 3.7 in terms of the pairwise updates. Furthermore, the FairPairs inspired feedback interventions can be trivially incorporated here (though we ignore this in the rest of the exposition in this chapter for the sake of simplicity).

Before we can provide a regret bound for the SoPer-R algorithm we need one additional result. Given function g and constant λ define $\tau_g(\lambda)$ as:

$$\tau_g(\lambda) = \lim_{x \to 0} \frac{g(\lambda \cdot x, 0, \ldots, 0)}{g(x, 0, \ldots, 0)} \tag{6.8}$$

Next, we will bound the change in a concave function on scaling its' arguments.

Lemma 14 *For any function g (satisfying the conditions of Lemma 13), constant $0 \leq \lambda \leq 1$ and values $v_1, v_2, \ldots, v_n \geq 0$, we can bound the change in value of g on scaling the values v_i by λ as follows:*

$$g(v_1, \ldots, v_i, \ldots, v_n) \geq \tau_g(\lambda) \cdot g(\lambda \cdot v_1, \ldots, \lambda \cdot v_i, \ldots, \lambda \cdot v_n) \tag{6.9}$$

We use this to characterize the sequence of position discounts and their smoothness, which is a key parameter of the main theorem. Thus for a utility measure with function F and γ_i discount factors, we define:

$$\Gamma_F = 1 - \min_i \tau_F(\frac{\gamma_{i+1}}{\gamma_i}) \tag{6.10}$$

We can now characterize the regret suffered by the SoPer-R algorithm for list-based utilities, as shown below in Theorem 15.

Theorem 15 *For any $\mathbf{w}_* \in \mathbf{R}^m$ and $\|\phi(\mathbf{x}, \mathbf{y})\|_{\ell_2} \leq R$ the average regret of the SoPer-R algorithm can be upper bounded as:*

$$REG_T \leq \frac{1}{\eta T} \sum_{t=0}^{T-1} \mathbf{E}_i[p_i \bar{\xi}_t] + \frac{\beta R \|\mathbf{w}_*\|}{\eta} + \frac{\sqrt{2}\sqrt{4 - \beta^2} R \|\mathbf{w}_*\|}{\eta \sqrt{T}}. \tag{6.11}$$

with: $\delta_i \geq \left(\Gamma_F \cdot \frac{1-p_i}{p_i}\right)$, $\eta = \min_i \ p_i \alpha_i$ and $\beta = (1 - \beta_{gr}) = \frac{1}{2}$.

Let us analyze this regret bound. The first term on the right-hand side indicates how far the user feedback violates the desired α_i, δ_i-informative feedback characterization due to model misspecification and bias/noise in the user feedback. This term implies that the regret does not necessarily converge to zero in such cases. The second term results from the fact that we can only guarantee a β_{gr}-approximate solution for the greedy algorithm (Alg 5). In practice, however, the solutions computed by the greedy algorithm tend to be much better. The third and final term converges to zero at a rate of $\sqrt{T}$. Note that none of the terms in the bound depend explicitly on the number of features, but that that it scales only in terms of margin $R\|\mathbf{w}_*\|$.

6.2.2 Social Perceptron for Sets (SoPer-S)

While DCG-style position discounts γ_i that decay smoothly are often appropriate, other models of utility require more discrete changes in the rank discounts. The *coverage* metric is an example of such a metric, which measures what fraction of the users will find atleast one document relevant to them in the set of M documents [133, 154, 176]. We call these metrics *set-based*, since they consider the first M documents in a ranking as a set (*i.e.*, position within the top-M positions does not matter). Clearly, we can model such metrics by setting the γ_i in the aggregation step (defined in Eq. 6.7) as

$$\gamma_i = \begin{cases} 1 & \text{if } i \leq M \\ 0 & \text{if } i > M. \end{cases}$$

However, the bound in Theorem 15 can be rather loose for this case, and the pairwise feedback construction model "wastes" information. In particular, since utility is invariant to reordering in the top M or below the top M, only pairwise

Algorithm 10: Social-Set-Based-Perceptron(C, M, p)

Function SetFeedback($\mathbf{y}, \mathcal{D}$)
$\bar{\mathbf{y}} \leftarrow \mathbf{y}$ ▷ Initialize with presented object
$\mathcal{D}_O \leftarrow \mathcal{D}/\mathbf{y}[1:M]$ ▷ Clicks on docs outside top M
for $i = 1 \cdots \min(C, |\mathcal{D}_O|)$ **do**
  $c \leftarrow \mathcal{D}_O[i]$ ▷ Clicked document
  $u \leftarrow$ Random (non-clicked) document from $\mathbf{y}[1:M]$ ▷ Non-clicked document
  Swap($\bar{\mathbf{y}}[j_u]$, $\bar{\mathbf{y}}[j_c]$)
return $\bar{\mathbf{y}}$

feedback between position M and $M+1$ provides information. To overcome this problem, we now present an alternate algorithm that is more appropriate for set-based utility functions.

The *Social Perceptron for Sets* (SoPer-S), shown in Algorithm 10, uses the same basic algorithm, but replaces the feedback mechanism. Now, clicked documents outside the top M are swapped with a random non-clicked document in the top M. This leads to a feedback set $\bar{\mathbf{y}}_t$ (of size M), that contains more (or at least as many) of the user's preferred documents than the top M elements of the presented ranking. Note that during the feedback creation, we only consider the first C clicks outside the top M. This parameter C is used to restrict the difference between the feedback set and the presented set. We now state a lemma we will use to bound the regret suffered by the SoPer-S algorithm for set-based utilities.

Lemma 16 *For any non-negative, submodular function g and set X with $|X| = n$, we*

can lower bound the function value of a random subset of size k as:

$$\mathbf{E}_{Y:Y\subseteq X,|Y|=k}[g(Y)] \geq \frac{k}{n} g(X) \tag{6.12}$$

Theorem 17 *For any* $\mathbf{w}_* \in \mathbf{R}^m$ *and* $\|\phi(\mathbf{x},\mathbf{y})\|_{\ell_2} \leq R$ *the average regret of the SoPer-S algorithm can be upper bounded as:*

$$REG_T \leq \frac{1}{\eta T}\sum_{t=0}^{T-1} \mathbf{E}_i[p_i \bar{\xi}_t] + \frac{\beta R\|\mathbf{w}_*\|}{\eta} + \frac{\sqrt{2}\sqrt{4-\beta^2}R\|\mathbf{w}_*\|}{\eta\sqrt{T}}. \tag{6.13}$$

with: $\delta_i \geq \left(\frac{C}{M}\cdot\frac{1-p_i}{p_i}\right)$, $\eta = \min_i\ p_i\alpha_i$ *and* $\beta = (1-\beta_{gr}) = \frac{1}{e}$.

Note that the proposed algorithms are efficient (due to the online updates) and scalable as the greedy algorithm only requires $O(nk)$ time to find a length k ranking over n documents. This can be further improved using lazy evaluation.

6.3 Empirical Evaluation

In this section, we empirically analyze the proposed learning algorithms for the task of extrinsic search result diversification. In particular, we (a) explore how well the algorithms perform compared to existing algorithms that do single-query learning; (b) compare how close our algorithms get to the performance of algorithms that require expert annotated examples of socially optimal ranking for cross-query learning; and (c) explore the robustness of our algorithm to noise and misspecification of the utility model.

6.3.1 Experiment Setup

We performed experiments using the standard diversification dataset from the *TREC 6-8 Interactive Track*. The dataset contains 17 queries, each with binary

Table 6.1: Summary of key properties of the TREC dataset.

Statistic	Value
Average number of documents per query	46.3
Average number of user types	20.8
Fraction of docs. relevant to > 1 user	0.21
Average number of users a document is relevant for	1.33
Fraction of docs. relevant to most popular user	0.38
Average probability of most popular user	0.29

relevance judgments for 7 to 56 different user types, which we translate into binary utility values. We consider the probability of a user type to be proportional to the number of documents relevant to that user type. We only consider documents that are relevant to at least 1 user type to focus the experiments on learning to diversify, not learning to determine relevance. Table 6.1 summarizes some key properties of the data.

To simulate user behavior, we use the following model. Users scan the documents of a ranking in order and click on the first document they consider relevant. Each (binary) decision of relevance is made incorrectly with a small probability of error. This error probability was set to zero for most experiments but later varied when studying the effect of user noise.

Unless mentioned otherwise, we used the coverage function ($F(x_1, \ldots, x_n) = \max_i x_i$) to define the submodular function for utility aggregation. We measured performance of the different methods in terms of the utility being optimized - *i.e.*, Set Utility (of size 5 sets) for the Set-Based methods and List Utility (up to rank 5) with DCG discounting factors, for the List-Based methods. Additionally

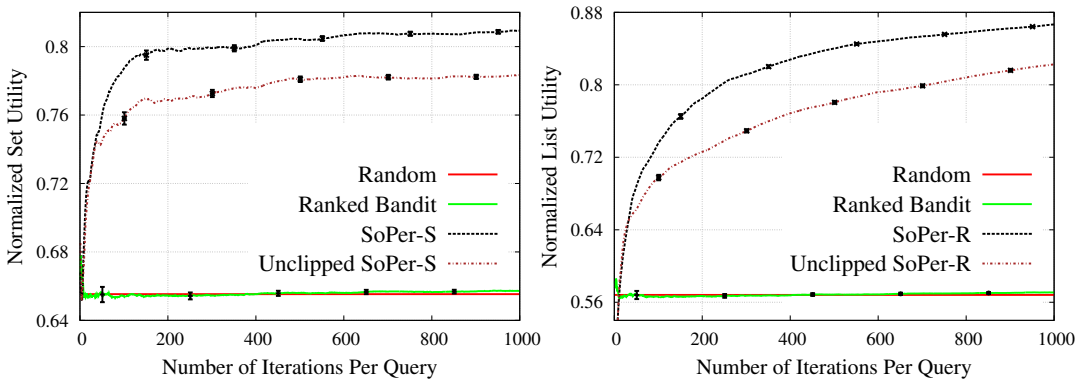

Figure 6.2: Performance of different methods for single-query learning to diversify. Performance is averaged over all queries, separately considering Set Utility (Left) and List Utility (Right). Standard error bars are shown in black.

we normalize the maximum scores per query to 1 (*i.e.*, $\forall x : U(\mathbf{x}, \mathbf{y}^*) = 1$), so as to get comparable scores across queries. We report the performance of each algorithm in terms of its running average of these scores (*i.e.*, $1 - REG_T$).

6.3.2 Can we learn to diversify for a single query?

We first evaluate our algorithms in the setting of the Ranked Bandits algorithm [133], which serves as a baseline. The Ranked Bandit algorithm learns a separate model for each query and cannot generalize across queries. Furthermore, its original version was limited to optimizing the coverage function, corresponding to the *max* aggregation in our framework. We use the *UCB1* variant of the Ranked Bandits algorithm, which was empirically found to be the best variant.

As a second baseline we report randomly ordering the results. Note that this is a competitive baseline, since (a) all documents are relevant to at least 1 user, and (b) the probability of users is proportional to the number of documents relevant to them.

For the SoPer-R and SoPer-S algorithms, documents were represented as unit-normalized TF-IDF word vectors. All learning algorithms were run twice for each of the 17 queries (with different random seeds) and the results are averaged across all 34 runs. As seen from Figure 6.2, the proposed algorithms perform much better than either of the two baselines. The Ranked Bandits algorithm converges extremely slowly, and is barely better than the random baseline after 1000 iterations. Both the SoPer-R and SoPer-S algorithm are able to learn substantially faster. Already within 200 iterations, the SoPer-S method is able to provide at least 1 relevant document to 80% of the user population, while random and Ranked Bandits perform at around 65%. Thus both proposed methods are clearly able to learn the diversity required in such rankings from individual user feedback.

We also explore variants of the SoPer-S and SoPer-R algorithms where we omit the final step of clipping negative weights to 0. While the unclipped versions of both algorithms still perform better than random, they fall short of the corresponding clipped versions as seen from Figure 6.2. Thus we can conclude that ensuring non-negative weights not only guarantees theoretical results, but also helps improve empirical performance.

6.3.3 Can we learn a cross-query model for diversification?

While the previous experiments indicate that the new algorithms can learn to diversify for a single query, such single-query learning is restricted to frequent queries that are issued hundreds of times. Instead, it is more desirable for diversification models to be trained across a distribution of queries.

To get a suitable representation that allows cross-query learning, we use the

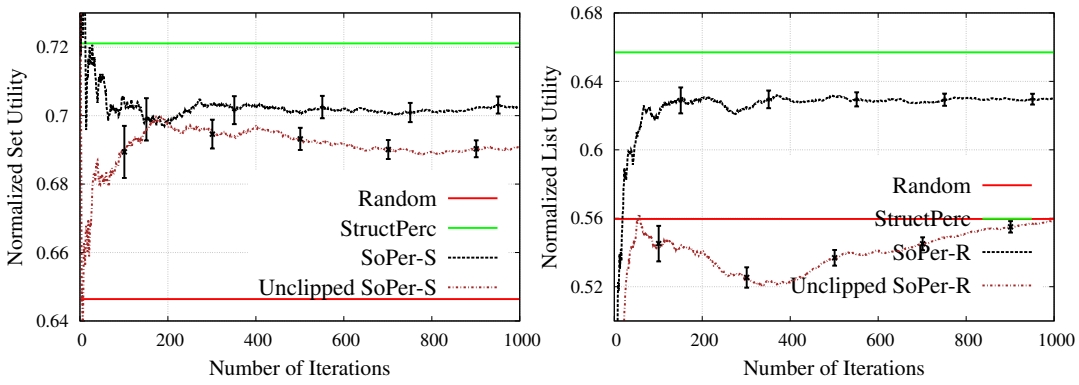

Figure 6.3: Set (L) and List (R) Utilities for learning to diversify across queries.

same *word-importance* feature vectors that were used in previous work on learning from expert-annotated feedback [176]. These features capture both the overall importance of a word (*e.g.*, "Does the word appears in at least x% of the documents?"), as well as the importance in the documents of the ranking (*e.g.*, "Does the word appear with frequency of atleast y% in the document?"). Using different such values of x and y along with other similar features, we get a total of 1197 features.

To produce the following results, all methods were run for 1000 iterations with 5 random seeds. The values reported are averaged across these 5 runs.

In this cross-query setting, we cannot apply Ranked-Bandits as it only works for a single query. Thus we again use the Random baseline in this experiment. Existing supervised learning algorithms for diversification are also not applicable here, as they require explicit training data of socially optimal rankings (*i.e.*, knowledge of all document-user relevance labels). However, we would like to estimate how well our algorithms can learn from (far weaker) implicit feedback data, in relation to conventional methods trained in such a *full information* setting. Thus we trained a structural perceptron, which internally uses the greedy

User's F	SET			
	Max	Sqrt	Lin	Rand
Max	.699 ±.005	.695 ±.005	.683 ±.005	.646 ±.006
Sqrt	.675 ±.006	.686 ±.006	.706 ±.006	.634 ±.006
Lin	.509 ±.006	.532 ±.006	.574 ±.007	.492 ±.006
User's F	LIST			
	Max	Sqrt	Lin	Random
Max	.630 ±.007	.620 ±.006	.618 ±.006	.557 ±.006
Sqrt	.656 ±.007	.654 ±.007	.684 ±.006	.610 ±.007
Lin	.500 ±.006	.504 ±.006	.566 ±.007	.474 ±.007

Table 6.2: Set and List Utilities (with standard error) when the two submodular functions *i.e.*, of the population (fixed for row) and the algorithm (fixed for column) are mismatched.

algorithm for prediction. This uses the same feature vector representation as our proposed algorithms, but is provided the *social optimal* at every iteration.

Figure 6.3 shows the average utility for the SoPer-S and SoPer-R algorithms, as well as the random baseline and the Structured Perceptron after 1000 iterations. Both SoPer-S and SoPer-R substantially outperform the random baseline, indicating that the proposed algorithms can learn to diversify for this cross-query setting. Both methods get close to the performance of the supervised method despite learning from far weaker feedback. For example, the SoPer-S method is able to satisfy 70% of the user population, as compared to the 64% of the baseline and 72% of the Structured Perceptron. We also again evaluate the unclipped versions of the algorithms. For the the unclipped SoPer-R, performance never rises above random, indicating the practical importance of maintaining a positive weight vector to ensure good performance of the greedy algorithm.

6.3.4 How robust are the algorithms to misspecification of the model?

While the previous experiments showed that the algorithms can learn efficiently when the submodular function of the user population (as used in computing the personal and social utilities) and the algorithm match, we now study what happens when there is a mismatch. More specifically, for the *cross-query diversification* setting, we ran the algorithms with three different submodular functions as defined by the concave function F: a) **Max**: $F(x_1, \ldots, x_n) = \max_i x_i$; b) **Lin**: $F(x_1, \ldots, x_n) = \sum_i x_i$; c) **Sqrt**: $F(x_1, \ldots, x_n) = \sqrt{\sum_i x_i}$. We also varied the population utility to each of these three functions, and obtained the average utility value (after 200 iterations) for all 9 combinations of functions. Note that we still ensured that SoPer-R was used to optimize the List based utilities, while SoPer-S was used for set-based ones.

The results (averaged over 5 runs) are shown in Table 6.2. We find that for both methods and all three population utility functions, the utility value is always better than the random baseline, regardless of the algorithm and function used. While the values may be highest when the functions align, we still find significant improvements over the baselines even when there is a mismatch. In fact, for some situations we find that the utility is highest when there is a mismatch: The case of a linear algorithm utility but SQRT population utility is one such example. We conjecture that is due to the relatively small set/list size of 5. On short rankings LIN and SQRT do not differ as much as on longer rankings. Additionally LIN does not suffer any approximation degradation as the greedy algorithm always provides an optimal solution for LIN.

Utility	Random	No Noise	Noise
Set	.646 ±.006	.699 ±.005	.694 ±.006
List	.557 ±.006	.630 ±.007	.631 ±.007

Table 6.3: Ranking performance in the presence of feedback noise.

6.3.5 Is the method robust to noise in the feedback?

In the real world, users make errors in judging the relevance of documents. To model this, we simulated users who make an error in each binary relevance judgment with 0.1 probability. This means that, as users go down the ranking, they may flip the true relevance label. Users now return as feedback the first document they *perceive* as relevant, which contains significant noise. We ran both our algorithms and measured the average utility after 200 iterations in the cross-query setting, with matching algorithm and population utilities using the Max function.

Table 6.3 shows the results (averaged over 5 runs) comparing the performance of the algorithms in both the noise-free and noisy settings. We see that the performance for both SoPer-S and SoPer-R is almost the same, with the gap to the baseline still being significant. The robustness to noise is also supported by the theoretical results. In particular, note that the definition of α_i, δ_i-informative feedback only requires that feedback be informative in expectations, such that the slack terms $\bar{\xi}_t$ may be zero even for noisy feedback. In general, we conclude that the algorithms are robust and applicable in noisy settings.

6.4 Summary

We proposed two sequential interactive learning algorithms in the coactive setting for aggregating the conflicting preferences of a diverse user population into a ranking that aims to optimize social utility. Formalizing the learning problem and model as learning an aggregate utility function that is submodular in the elements of the ranking and linear in the parameters, we were able to provide regret bounds that characterize the worst-case behavior of the algorithm. In an empirical evaluation, the algorithms learned substantially faster than existing algorithms for single-query diversification. For learning cross-query diversification models, the algorithms are robust and the first known algorithms that can be trained using implicit feedback.

CHAPTER 7

ADDING INTERACTIVITY TO RANKINGS: DYNAMIC RANKINGS

The previous chapter tackled the problem of interactive learning for extrinsic diversity *i.e.,* diversifying results to tackle ambiguous queries. While the algorithms introduced last chapter provide us with theoretically and empirically sound ways of learning to diversify from user interaction data, they are still faced with two conflicting goals (as are all other retrieval systems given an ambiguous query). On the one hand, they should diversify and strive to present results for as many query intents as possible. On the other hand, they should provide depth for each intent by displaying more than a single result. Clearly, there is an inherent trade-off between *depth* (number of results provided for an intent) and *diversity* (number of intents served) in the conventional ranked-retrieval setting, since increasing one invariably leads to a decrease of the other. How can we avoid this trade-off and obtain diversity while not compromising on depth?

We argue that a key to solving the conflict between depth and diversity lies in the move to *dynamic* retrieval models [30] that can take advantage of user interactions to optimize user utility. Instead of presenting users with a single one-size-fits-all ranking, dynamic retrieval models allow users to adapt the ranking dynamically through interaction. The idea here is to use interactions to adapt the presented objects *on-the-fly*.

While Brandt et al [30] provided evidence that even limited dynamism in the rankings can greatly improve retrieval effectiveness, they did not provide an efficient algorithm for computing dynamic rankings, nor did they study the problem of learning dynamic ranking functions. In this chapter, we resolve these

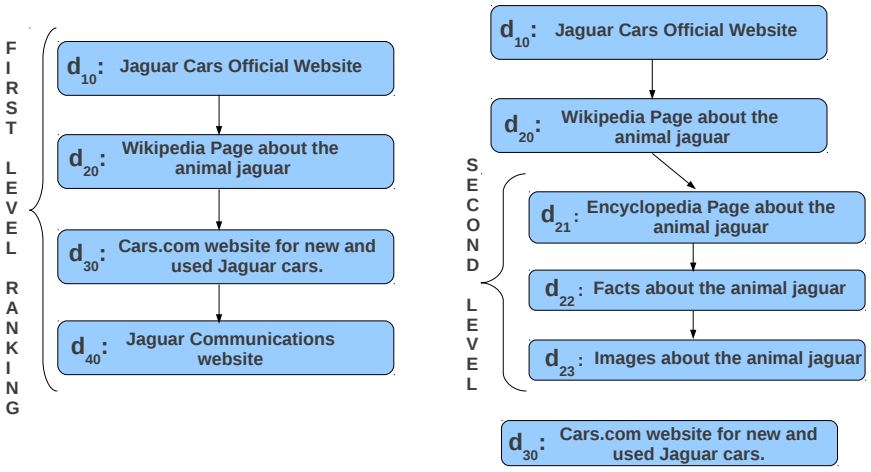

Figure 7.1: A user interested in the animal "jaguar" interacts with the first-level ranking (left) and obtains second-level results (right).

two open questions. In particular, we propose a new *two-level dynamic ranking model*. Intuitively, the first level provides a diversified ranking of results on which the system can sense the user's interactions. Conditioned on this feedback, the system then interactively provides a second-level rankings. A possible layout is given in Figure 7.1. The left-hand panel shows the first-level ranking initially presented to the user. The user then chooses to expand the second document (*e.g.*, by clicking) and a second-level ranking is inserted as shown in the right panel. Conceptually, the retrieval system maintains two levels of rankings, where each second-level ranking is conditioned on the head document in the first-level ranking. This idea relates to relevance feedback [1] where user feedback is used to update the ranking. These two-level dynamic rankings also motivated the interactive ranking approach proposed in Chapter 4.3, where aspects form the second level ranking to improve intrinsic diversity retrieval.

To operationalize the construction and learning of such two-level rankings in a rigorous way, we define a new family of submodular performance measure for diversified retrieval. Many existing retrieval measures (*e.g.*, , Precision@k, DCG, Intent Coverage) are special cases of this family. We then operationalize the

problem of computing an optimal two-level ranking as maximizing the given performance measure. While this optimization problem is NP-hard, we provide an algorithm that has an $1 - e^{-(1-\frac{1}{e})}$ approximation guarantee.

Finally, we also propose a new method for learning the (mutually dependent) relevance scores needed for two-level rankings. Following a structural SVM approach, we learn a discriminant model that resembles the desired performance measure in structure, but learns to approximate unknown intents based on query and document features.

7.1 Two-Level Dynamic Rankings

Current methods for diversified retrieval, including the ones proposed in Chapter 6, are *static* in nature *i.e.,* they stay unchanged through a user session. On the other hand, a *dynamic* model can adapt the ranking based on interactions with the user. The primary motivation for using a dynamic model is addressing the inherent trade-off between depth and diversity in static models.

Consider the example with four (equally likely) user intents $\{t_1, ..., t_4\}$ and documents $\{d_1, ..., d_9\}$ with user utilities $U(d_j|t_i)$ as given in Table 7.1. On the one hand, a non-diversified static ranking method could present $d_7 \rightarrow d_8 \rightarrow d_9$ as its top three documents, providing two relevant documents for intents t_3 and t_4 but none for intents t_1 and t_2. On the other hand, a diversified static ranking $d_7 \rightarrow d_1 \rightarrow d_4$ covers all intents, but this ranking lacks depth since no user gets more than one relevant document.

As an alternative, consider the following two-level dynamic ranking. The user is presented with $d_7 \rightarrow d_1 \rightarrow d_4$ as the first-level ranking. Users can now

$U(d_j\|t_i)$	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
t_1	1	1	1	0	0	0	0	0	0
t_2	0	0	0	1	1	1	0	0	0
t_3	0	0	0	0	0	0	1	1	0
t_4	0	0	0	0	0	0	1	0	1

Table 7.1: Utility $U(d_j|t_i)$ of document d_j for intent t_i.

expand any of the first-level results to view a second-level ranking. Users interested in d_7 (and thus having intent t_3 or t_4) can *expand* that result and receive a second-level ranking consisting of d_8 and d_9. Similarly, users interested in d_1 will get d_2 and d_3; and users interested in d_4 will get d_5 and d_6.

For this dynamic ranking, every user gets at least one relevant result after scanning at most three documents (*i.e.,* the first-level ranking). Furthermore, users with intents t_3 and t_4 receive two relevant results in the top three positions of their dynamically constructed ranking $d_7 \rightarrow d_8 \rightarrow d_9 \rightarrow d_1 \rightarrow d_4$. Users with intent t_1 also receive two relevant results in the top three positions while those with intent t_2 still receive one relevant result. This illustrates how a dynamic two-level ranking can simultaneously provide diversity and increased depth.

In the above example, interactive feedback from the user was the key to achieving both depth and diversity. More generally, we assume the following **model of user behavior**, which we denote as policy π_d. Users expand a first-level document if and only if that document is relevant to their intent. When users skip a document, they continue with the next first-level result. When users expand a first-level result, they go through the second-level rankings before continuing from where they left off in the first-level ranking. It is thus possible for a user to see multiple second-level rankings. Hence we do not allow documents to appear more than once across all two-level rankings.

Unlike the user model proposed in [30], here user feedback is only assumed

only for one level of rankings (*i.e.,* the first-level), whereas [30] requires that users give feedback many levels deep. Furthermore, unlike in [30], we model that users return to the top-level ranking. We conjecture that these differences make the two-level model more natural and appropriate for practical use.

We now define some notation used later in this chapter. The documents shown in a first-level ranking of **length** L are called the **head** documents. The documents shown in a second-level ranking are called the **tail** documents. The number of tail documents is referred to as the **width** W. A **row** denotes a **head** document and all its **tail** documents. Static rankings are denoted as θ while two-level rankings are denoted as $\Theta = (\Theta_1, \Theta_2, ...\Theta_i, ..)$. Here $\Theta_i = (d_{i0}, d_{i1}, ..., d_{ij}, ...)$ refers to the i^{th} row of a two-level ranking, with d_{i0} representing the head document of the row and d_{ij} denoting the j^{th} tail document of the second-level ranking. We denote the candidate set of documents to rank for a query q by $\mathcal{D}(q)$, the set of possible intents by $\mathcal{T}(q)$ and the distribution over an intent $t \in \mathcal{T}(q)$, given a query q, by $\mathbf{P}[t|q]$.

7.2 Performance Measures for Diversified Retrieval

To define what constitutes a good two-level dynamic ranking, we first define the measure of retrieval performance we would like to optimize. We first start with evaluation measures for one-level rankings, and then generalize them to the two-level case.

7.2.1 Measures for Static Rankings

Existing performance measures range from those that do not explicitly consider multiple intents (*e.g.*, NDCG, Average Precision), to measures that reward diversity. Measures that reward diversity give lower marginal utility to a document, if the intents the document is relevant to are already well represented in the ranking. We call this the *diminishing returns* property. The extreme case is the "intent coverage" measure (*e.g.*, [157, 176]), which attributes utility only to the first document relevant for an intent.

We define a family of measures that includes a whole range of diminishing returns models, and that includes most existing retrieval measures. Let $g : \mathbb{R} \to \mathbb{R}$ with $g(0) = 0$ be a concave, non-negative, and non-decreasing function that models the diminishing returns, then we define the utility of the ranking $\theta = (d_1, d_2, ..., d_k)$ for intent t as

$$U_g(\theta|t) = g\Big(\sum_{i=1}^{|\theta|} \gamma_i U(d_i|t)\Big). \tag{7.1}$$

The $\gamma_1 \geq ... \geq \gamma_k \geq 0$ are discount factors and $U(d|t)$ is the relevance rating of document d for intent t. For a distribution of user intents $\mathbf{P}[t|q]$ for query q, the overall utility of a static ranking θ is the expectation

$$U_g(\theta|q) = \sum_{t \in \mathcal{T}(q)} \mathbf{P}[t|q]\ U_g(\theta|t). \tag{7.2}$$

Note that many existing retrieval measures are special cases of this definition. For example, if one chose g to be the identity function, one recovers the intent-aware measures proposed in [4] and the modular measures defined in [30]. Further restricting $\mathbf{P}[t|q]$ to put all probability mass on a single intent leads to conventional measures like DCG for appropriately chosen γ_i. At the other extreme, choosing $g(x) = \min(x, 1)$ leads to the intent coverage measures. Since

g can be chosen from a large class of functions, this family of performance measures covers a wide range of diminishing returns models.

7.2.2 Measures for Dynamic Rankings

We extend this family of performance measures to dynamic rankings. The key change here is that users interactively adapt which results they view. How users expand first-level results was defined in the previous section as π_d. Under π_d, it is natural to define the utility of a dynamic ranking Θ analogous to Equation (7.1).

$$U_g(\Theta|t) = g\Big(\sum_{i=1}^{|\Theta|}\Big(\gamma_i U(d_{i0}|t) + \sum_{j=1}^{|\Theta_i|}\gamma_{ij}U(d_{i0}|t)U(d_{ij}|t)\Big)\Big). \tag{7.3}$$

Like for static rankings, $\gamma_1 \geq \gamma_2 \geq ...$ and $\gamma_{i1} \geq \gamma_{i2} \geq ...$ are position-dependent discount factors. Furthermore, we again take the expectation over multiple user intents as in Equation (7.2) to obtain $U_g(\Theta|q)$.

Note that the utility of a second-level ranking for a given intent is zero unless the head document in the first-level ranking has non-zero relevance for that intent. This encourages second-level rankings to only contain documents relevant to the same intents as the head document, thus providing depth. The first-level ranking, on the other hand, provides diversity as controlled through the choice of function g. The "steeper" g diminishes returns of additional relevant documents, the more diverse the first-level ranking gets.

7.3 Computing dynamic rankings

In this section, we provide an efficient algorithm for computing dynamic rankings that maximize the performance measures defined in the previous sec-

Algorithm 11: Computing a two-level dynamic ranking.

Input: $(q, \mathcal{D}(q), \mathcal{T}(q), \mathbf{P}[t|q] : t \in \mathcal{T}(q))$, $g(\cdot)$, L, W.

Output: A dynamic ranking Θ.

$\Theta \leftarrow$ new_two_level()

while $|\Theta| \leq L$ **do**

  $bestU \leftarrow -\infty$

  for all $d \in \mathcal{D}(q)$ s.t. $d \notin \Theta$ **do**

    $row \leftarrow$ new_row(); $row.head \leftarrow d$

    for $j = 1$ to W **do**

      $bestDoc \leftarrow \texttt{argmax}_{d' \notin \Theta \cup row} U_g(\Theta \oplus (row \oplus d')|q)$

      $row \leftarrow row \oplus bestDoc$

    if $U_g(\Theta \oplus row|q) > bestU$ **then**

      $bestU \leftarrow U_g(\Theta \oplus row|q)$; $bestRow \leftarrow row$

  $\Theta \leftarrow \Theta \oplus bestRow$

tion. In the proposed greedy algorithm (Alg. 11), the operator $\oplus$ denotes either adding a document to a row, or adding a row to an existing ranking. This is an extension to the greedy algorithms we have seen earlier (such as Algorithm 5) to account for the second level of rankings. In each iteration, considers every document in the remaining collection as the head document of a candidate row. For each candidate row, W documents are greedily added to maximize the utility $U_g(\Theta|q)$ of the resulting partial dynamic ranking Θ. Once rows of length W are constructed, the row which maximizes the utility is added to the ranking. The above steps are repeated until the ranking has L rows. Algorithm 11 is efficient, requiring $O(|\mathcal{T}|)$ space and $O(|\mathcal{T}||\mathcal{D}|^2)$ time.

Our greedy algorithm is closely related to submodular function maximiza-

tion. Maximizing monotonic submodular functions is a hard problem, but a greedily constructed set gives an $(1 - 1/e)$ approximation [121] to the optimal. Since the definition of our utility in (7.2) involves a concave function, it is not hard to show that selecting a ranking of rows is a submodular maximization problem. Moreover, given the head document, finding the best row is also a submodular maximization problem. Thus, finding a dynamic ranking to maximize our utility is a *nested* submodular maximization problem, and we can show the following approximation guarantee for Algorithm 11.

Lemma 18 *Algorithm 11 is* $(1 - e^{-(1-\frac{1}{e})})$ *approximate.*

The proof is similar to the one in [78], although adapted for a more general set of measures (beyond just intent coverage).

7.4 Learning Dynamic Rankings

In the previous section, we showed that a dynamic ranking can be efficiently computed when all the intents and relevance judgments for a given query are known. In this section, we propose a supervised learning algorithm that can predict dynamic rankings on previously unseen queries.

Our goal here is to learn a mapping from a query q to a dynamic ranking Θ. We pose this as the problem of learning a weight vector $\mathbf{w} \in \mathbb{R}^N$ from which we can make a prediction as follows:

$$h_{\mathbf{w}}(q) = \texttt{argmax}_{\Theta}\ \mathbf{w}^{\top}\Psi(q, \Theta). \tag{7.4}$$

As further explained below, $\Psi(q, \Theta) \in \mathbb{R}^N$ is a joint feature-map between query q and dynamic ranking Θ.

Given a set of training examples $(q^i, \Theta^i)_{i=1}^n$, the structural SVM framework [162] can be use to learn a discriminant function by minimizing the empirical risk $\frac{1}{n}\sum_{i=1}^n \Delta(\Theta^i, h_{\mathbf{w}}(q^i))$, where Δ is a loss function. Unfortunately, however, the Θ^i are typically not given directly as part of the training data. Instead, we assume that we are given training data of the form $(q^i, \mathcal{D}(q^i), \mathcal{T}(q^i), \mathbf{P}[t|q] : t \in \mathcal{T}(q^i))_{i=1}^n$, using which we then compute the dynamic rankings Θ^i by maximizing the utility U_g (approximately) using Algorithm 11. These Θ^is will be used as the training labels henceforth.

A key aspect of structural SVMs is to appropriately define the joint-feature map $\Psi(q, \Theta)$. For our problem, we propose

$$\mathbf{w}^\top \Psi(q, \Theta) := \sum_{v \in V_{\mathcal{D}(q)}} \mathbf{w}_v^\top \phi_v U_g(\Theta|v) + \sum_{s \in V_{\mathcal{D}(q)\times\mathcal{D}(q)}} \mathbf{w}_s^\top \phi_s(\Theta), \tag{7.5}$$

where $V_{\mathcal{D}(q)}$ denotes an index set over the words in the candidate set $\mathcal{D}(q)$. The vector ϕ_v denotes word-level features (for example, how often a word occurs in a document) for the word corresponding to index v. The utility $U_g(\Theta|v)$ is analogous to (7.3) but is now over the words in the vocabulary (rather than over intents). The word-level features are reminiscent of the features used in diverse subset prediction [176]. The key assumption is that the words in a document are correlated with the intent since documents relevant to the same intent are likely to share more words than documents that are relevant to different intents.

The second term in Equation 7.5 captures the similarity between head and tail documents. In this case, $V_{\mathcal{D}(q)\times\mathcal{D}(q)}$ denotes an index set over all document pairs in $\mathcal{D}(q)$. Consider an index s that corresponds to documents d_1 and d_2 in $\mathcal{D}(q)$. $\phi_s(\Theta)$ is a feature vector describing the similarity between d_1 and d_2 in Θ when d_1 is a head document in Θ and d_2 occurs in the same row as d_1 ($\phi_s(\Theta)$ is simply a vector of zeros otherwise). An example of a feature in $\phi_s(\Theta)$ that

captures the similarity between two documents is their TFIDF cosine.

Using these features, $\mathbf{w}^\top \Psi(q, \Theta)$ models the utility of a given dynamic ranking Θ. During learning, $\mathbf{w}$ should be selected so that better rankings receive higher utility than worse rankings. This is achieved by solving the following structural SVM optimization problem for $\mathbf{w}$:

$$\min_{\mathbf{w}, \xi \geq 0} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{7.6}$$

$$\text{s.t.} \forall i, \forall \Theta : \mathbf{w}^\top \Psi(q^i, \Theta^i) - \mathbf{w}^\top \Psi(q^i, \Theta) \geq \Delta(\Theta^i, \Theta | q^i) - \xi_i$$

The constraints in the above formulation ensure that the predicted utility for the target ranking Θ^i is higher than the predicted utility for any other Θ. The objective function in (7.6) minimizes the empirical risk while trading it off (via the parameter $C > 0$) with the margin. The loss between Θ^i and Θ is given by $\Delta(\Theta^i, \Theta | q^i) := 1 - \frac{U_g(\Theta|q^i)}{U_g(\Theta^i|q^i)}$ which ensures that the loss is zero when $\Theta = \Theta^i$. It is easy to see that a dynamic ranking Θ has a large loss when its utility is low compared to the utility of Θ^i.

Even though Equation (7.6) has an exponential number of constraints, the corresponding quadratic program can be solved in polynomial time using the cutting-plane algorithm [162]. In each iteration of the cutting-plane algorithm, the most violated constraints in (7.6) are added to a working set and the resulting quadratic program is solved. Given a current $\mathbf{w}$, the most violated constraints are obtained by solving:

$$\texttt{argmax}_\Theta \ \mathbf{w}^\top \Psi(q^i, \Theta) + \Delta(\Theta^i, \Theta | q^i). \tag{7.7}$$

Algorithm 11 can be used to solve the above problem, even though the formal approximation guarantee does not hold in this case. Once a weight vector $\mathbf{w}$ is obtained, the dynamic ranking for a test query can be obtained from Eq. (7.4).

7.5 Empirical Study

Experiments were conducted on the TREC 6-8 Interactive Track (TREC) and the Diversity Track of TREC 18 using the ClueWeb collection (WEB). The 17 queries in TREC contain between 7 to 56 different manually judged intents. In the case of WEB, we used 28 queries with 4 or more intents. Similar to the setup described in Section 6.3.1, probability of an intent was set proportional to the number of documents relevant to that intent. A key difference between the two datasets is that the most prevalent intent covers 73.4% of all relevant documents for the WEB dataset, but only 37.6% for TREC.

The number of documents in the first-level ranking was set to 5. The width of the second-level rankings was set to 2. For simplicity, we chose all factors γ_i and γ_{ij} in Equations (7.1) and (7.3) to be 1. Further, we chose $U(d|t) = 1$ if document d was relevant to intent t and set $U(d|t) = 0$ otherwise.

7.5.1 Controlling Diversity and Depth

The key design choice of our family of utility measures is the concave function g. As Algorithm 11 directly optimizes utility, we explore how the choice of g affects various properties of the two-level rankings produced by our method.

We experiment with four different concave functions g, each providing a different diminishing-returns model. At one extreme, we have the identity function $g(x) = x$ which corresponds to modular returns. Using this function in Eq. (7.1) leads to the intent-aware Precision measure proposed in [4], and it is the only function considered in [30]. We therefore refer to this function as PREC. It is not hard to show that Algorithm 11 actually computes the optimal two-

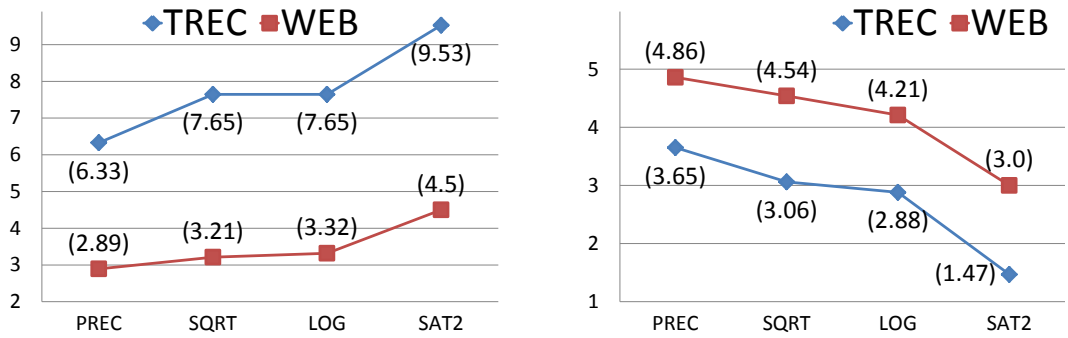

Figure 7.2: Average number of intents covered (left) & average number of documents for prevalent intent (right) in the first-level ranking.

level ranking for this choice of g. On the other end of the spectrum, we study $g(x) = \min(x, 2)$. By remaining constant after two, this function discourages presenting more than two relevant documents for any intent. This measure will be referred to as SAT2 (short for "satisfied after two"). In between these two extremes, we study the square root function (SQRT) $g(x) = \sqrt{x}$ and the log function (LOG) $g(x) = \log(1 + x)$.

To explore how dynamic rankings can differ, we used Algorithm 11 to compute the two-level rankings (approximately) maximizing the respective measure. Figure 7.2 shows how g influences diversity. The left-hand plot shows how many different intents are represented in the top 5 results of the first-level ranking on average. The graph shows that the stronger the diminishing-returns model, the more different intents are covered in the first-level ranking. In particular, the number of intents almost doubles on both datasets when moving from PREC to SAT2. In contrast, the number of documents on the most prevalent intent in the first-level ranking decreases, as shown in the right-hand plot. This illustrates how the choice of g can be used to control the desired amount of diversity in the first-level ranking.

Table 7.2 provides further insight into the impact of g, now also including

Evaluation \ Optimization	PREC	SQRT	LOG	SAT2
PREC	**0.315**	0.302	0.294	0.164
SQRT	1.612	**1.664**	1.659	1.333
LOG	1.216	1.267	**1.27**	1.046
SAT2	1.18	1.335	1.349	**1.487**

Table 7.2: Performance when optimizing and evaluating using different performance measures for TREC.

the contributions of the second-level rankings. The rows correspond to different choices for g when evaluating expected utility according to Eq. (7.3), while the columns show which g the two-level ranking was optimized for. Not surprisingly, the diagonal entries of Tables 7.2 show that the best performance for each measure is obtained when optimizing for it. The off-diagonal entries show that different g used during optimization lead to substantially different rankings. This is particularly apparent when optimizing the two extreme performance measures PREC and SAT2; optimizing one invariably leads to rankings that have a low value of the other. In contrast, optimizing LOG or SQRT results in much smoother behavior across all measures, and both seem to provide a good compromise between depths (for the prevalent intent) and diversity. The results for WEB are qualitatively similar and are omitted for space reasons.

7.5.2 Static vs. Dynamic Ranking

The ability to simultaneously provide depth and diversity was a key motivation for our dynamic ranking approach. We now evaluate whether this goal is indeed achieved. We compare the two-level rankings produced by Algorithm 11 (denoted *Dyn*) with several static baselines. These static baselines are also computed by Algorithm 11, but with zero width second-level rankings.

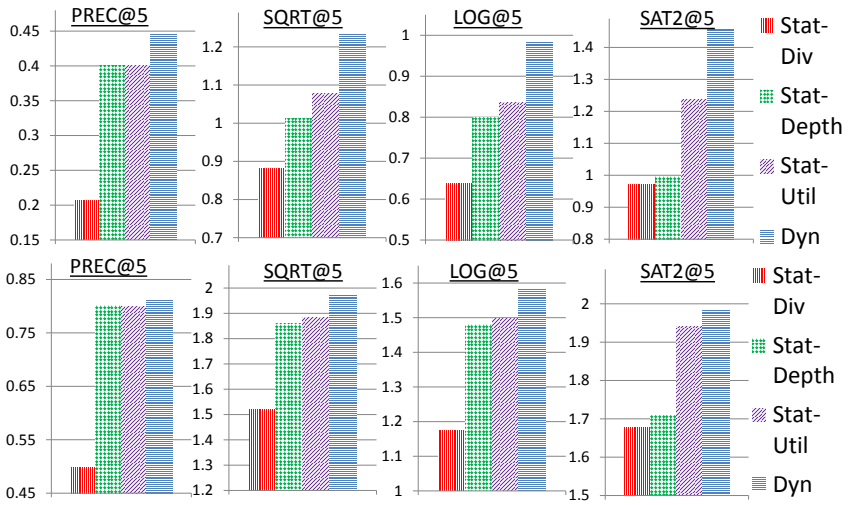

Figure 7.3: Comparing the retrieval quality of Static vs. Dynamic Rankings for TREC (Top) and WEB (Bottom).

First, we compare against a diversity-only static ranking that maximizes intent coverage as proposed in [176] (denoted *Stat-Div*). Second, we compare against a depth-only static ranking by choosing g to be the identity function (denoted *Stat-Depth*). And, third, we produce static rankings that optimize SQRT, LOG, and SAT2 (denoted *Stat-Util*). Note that both Dyn and Stat-Util optimize the same measure that is used for evaluation.

To make a fair comparison between static and dynamic rankings, we measure performance in the following way. For static rankings, we compute performance using the expectation of Eq. (7.1) at a depth cutoff of 5. In particular, we measure PREC@5, SQRT@5, LOG@5 and SAT2@5. For two-level rankings, the number of results viewed by a user depends on how many results he/she expands. So, we truncate any user's path through the two-level ranking after visiting 5 results and compute PREC@5, SQRT@5, LOG@5 and SAT2@5 for the truncated path.

Results of these comparisons are shown in Figure 7.3. First, we see that both

Dyn and Stat-Util outperform Stat-Div, illustrating that optimizing rankings for the desired evaluation measure leads to much better performance than using a proxy measure as in Stat-Div. Note that Stat-Div never tries to present more than one result for each intent, which explains the extremely low "depth" performance in terms of PREC@5. But Stat-Div is not competitive even for SAT2, since it never tries to provide a second result. Second, at first glance it may be surprising that Dyn outperforms Stat-Depth even on PREC@5, despite the fact that Stat-Depth explicitly (and globally optimally) optimizes depth. To understand consider the following situation where A is the prevalent intent, and there are three documents relevant to A and B and three relevant to A and C. Putting those sets of three documents into the first two rows of the dynamic ranking provides better PREC@5 than sequentially listing them in the optimal static ranking.

Overall we find the dynamic ranking method outperforming all static ranking schemes on all the metrics – in many cases with a substantial margin. This gain is more pronounced for TREC than for WEB. This can be explained by the fact that WEB queries are less ambiguous, since the single most prevalent intent accounts for more than 70% of all queries on average.

7.5.3 Learning Two-level Ranking Functions

So far we have evaluated how far Algorithm 11 can construct effective two-level rankings if the relevance ratings are known. We now explore how far our learning algorithm can predict two-level rankings for previously unseen queries. For all experiments in this section, we learn and predict using SQRT as the choice for g, since it provides a good trade-off between diversity and depth

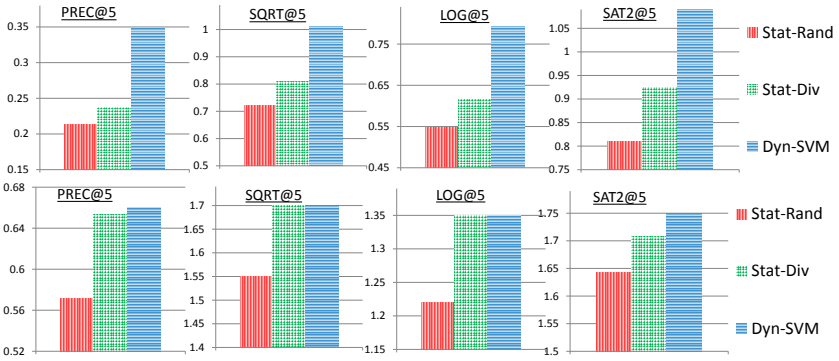

Figure 7.4: Performance of learned functions, comparing static & dynamic rankings for TREC (Top) and WEB (Bottom).

as shown above.

We performed standard preprocessing such as tokenization, stopword removal and Porter stemming. Since the focus of our work is on diversity and not on relevance, we rank only those documents that are relevant to at least one intent of a query. This simulates a candidate set that may have been provided by a conventional retrieval method. This setup is similar to that used in previous work [176].

Many of our features in ϕ_v follow those used in [176]. These features provide information about the importance of a word in terms of two different aspects. A first type of feature describes the overall importance of a word. A second type of feature captures the importance of a word in a document. An example of this type of feature is whether a word appears with frequency at least $y\%$ in the document. Finally, we also use features ϕ_s that model the relationship between the documents in the second-level ranking and the corresponding head document of that row. Examples of this type of feature are binned features representing TFIDF similarity of document pairs and the number of common words that appear in both documents with a frequency of at least $x\%$.

Dynamic vs. Static: In the first set of experiments, we compare our learning method (*Dyn-SVM*) for two-level rankings with two static baselines. The first static baseline is the learning method from [176] which optimizes diversity (referred to as *Stat-Div*). We also consider a *random* static baseline (referred to as *Stat-Rand*), which randomly orders the candidate documents. This is a competent baseline, since all our candidate documents are relevant to at least one intent.

Figure 7.4 shows the comparison between static and dynamic rankings. For TREC, *Dyn-SVM* substantially outperforms both static baselines across all performance metrics, mirroring the results we obtained in Section 7.5.2 where the relevance judgments were known. This shows that our learning method can effectively generalize the multi-intent relevance judgments to new queries. On the less ambiguous WEB dataset the differences between static and dynamic rankings are smaller. While *Dyn-SVM* substantially outperforms *Stat-Rand*, *Stat-Div* is quite competitive on WEB.

7.6 Summary

This chapter introduced the notion of incorporating interactivity into the predicted objects so as to improve user utility on-the-fly for the task of extrinsic diversification. In particular, it proposed a two-level dynamic ranking approach that provides both diversity and depth for ambiguous queries by exploiting user interactivity. We showed that the approach has the following desirable properties. First, it covers a large family of performance measures, making it easy to select a diminishing returns model for the application setting at hand. Second, we

presented an efficient algorithm for constructing two-level rankings that maximizes the given performance measure with provable approximation guarantees. Finally, we provided a structural SVM algorithm for learning two-level ranking functions, showing that it can effectively generalize to new queries.

Part IV

Using Stated Preferences to Scale Student Evaluation

The previous two parts of this dissertation illustrated the importance of designing feedback interventions and the user behavioral model along with the interactive learning algorithm using examples primarily from the search and recommendation domains. In this part, we are going to explore the use of learning for problems from a different domain, namely education. Recently, there has been an increased interest on the use of technology in education. One such problem of interest is peer grading, which is a promising approach to scale up student assessment in large classes (such as online courses).

More specifically, the next chapter proposes the use of ordinal feedback from students (*e.g., project X is better than project Y*) as opposed to the cognitively harder and less-reliable cardinal feedback (*e.g., project X is a B-*). It covers different algorithms that can reliably aggregate the *ordinal* student grades to come up with an overall grade for each assignment. Empirical studies using data collected in a real-world university classroom, demonstrate these "aggregated" grades to be on-par with cardinal grading approaches as well as conventional grading alternatives such as TA and instructor grading. Furthermore, these aggregation approaches can be extended to provide instructors with more detailed grading information, such as the uncertainty in each assignment's grade. The overall ordinal peer grading approach was found to be a valuable learning experience by students as well as a helpful grading resource by their instructors.

CHAPTER 8

ORDINAL PEER GRADING

Massive Online Open Courses (*MOOCs*) have the potential to revolutionize higher education with their accessibility and low costs. While they empower learning across a diverse range of subjects and millions of students [53], they require instructors to adapt traditional classroom logistics to scale to classes with upwards of 20000 students [88]. One such key logistic is the *evaluation of students* in MOOCs.

While scalable automatic-grading schemes — such as multiple-choice questions — exist, they are not suitable in all settings [26, 71, 72, 163]. For instance, liberal-arts courses and research-oriented classes require more open-ended testing such as essays and reports, which are very challenging to evaluate automatically. A lack of reliable assessment techniques for these types of assignments currently limits the kinds of courses offered as MOOCs.

Peer grading, where students — not instructors or TAs/staff — provide feedback on the work of other students in the class, has been proposed as a solution. Peer grading naturally overcomes the problem of scale [8, 63, 100, 115], since the number of "graders" matches the number of students. Despite this inherent scalability of peer grading, a key obstacle for peer grading to work is the fact that the students are not trained graders and are just learning the material themselves. Hence, to ensure good-quality grades it is imperative that grading guidelines are easy to communicate and apply, making the feedback process as easy and unambiguous as possible for the student graders.

Given broad evidence across many different tasks that demonstrates *ordi-*

nal feedback to be easier to provide and more reliable than *cardinal* feedback [20, 36, 98, 119, 155], it is therefore desirable to base peer grading on ordinal feedback (e.g. "project A is better than project B"). Unfortunately, all existing methods for aggregating peer grade feedback into an overall assessment require that students provide *cardinal* feedback (e.g. "project A should get 87 out of 100"). Furthermore, the efficacy of simple techniques for aggregating cardinal feedback, such as averaging, has been questioned [28, 39, 120]. While probabilistic machine learning methods have recently been proposed to improve performance [126], they still face the problem that students may be grading on different scales. For example, students may have a preconception of what constitutes a B+ based on the university they come from. Non-linear grading scales also cause fundamental problems for these cardinal grade based methods, as they rely on the difference between an A+ and an A being the same as the difference between a C+ and a C (which is typically not true in practice).

To overcome the problems of cardinal feedback, we introduce the task of *ordinal peer grading* in this thesis. By having students give ordinal statements and not cardinal statements as feedback, we offload the problem of developing and communicating a precise absolute grading scale onto the peer grading algorithm. The key technical contributions of this work lie in the development of methods for ordinal peer grading, where the goal is to automatically infer an overall assessment of a set of assignments from ordinal peer feedback. Furthermore, a secondary goal of the proposed methods is to infer how accurately each student provides feedback, so that reliable grading can be incentivized (by including grading performance as a component of the overall grade for instance).

To this effect, we propose several machine learning methods for *ordinal peer*

grading, which differ by how probability distributions over rankings are modeled. These methods, which extend classical rank-aggregation algorithms, allow us to jointly infer the assignment grades and grader reliabilities in an efficient manner. We also design Bayesian alternatives to these methods to provide instructors of these courses with more detailed information. More specifically, the resulting Metropolis-Hastings [46] based Markov Chain Monte-Carlo (MCMC) methods, allow us to report the uncertainty and confidence interval estimates for the grade of each assignment.

To study the applicability of the proposed methods in real-world settings, this thesis details a dataset of peer-assessment grades collected as part of a university-level course. Using this data, the efficacy of the proposed ordinal feedback techniques is demonstrated in comparison to the existing cardinal feedback techniques. Furthermore, the proposed ordinal peer grading methods were found to be comparable in quality with traditional evaluation techniques, such as course-staff (TAs) based grading, that were used in the course in parallel. Using this classroom data, other properties of these techniques were also investigated, including their robustness, data dependence, self-consistency and quality of uncertainty estimates. Finally, an analysis of responses to a survey completed by students in the classroom experiment is also provided. The results of the survey indicated that most students found the peer grading experience (of receiving and providing peer feedback) helpful and valuable.

8.1 The Peer Grading Problem

We begin by formally defining the peer grading problem, as it presents itself from a machine learning perspective. We are given a set of $|D|$ *assignments*

$D = \{d_1, ..., d_{|D|}\}$ (*e.g.,* essays, reports) which need to be graded. Grading is done by a set of $|G|$ graders $G = \{g_1, ..., g_{|G|}\}$ (*e.g.,* student peer graders, reviewers), where each grader receives a subset $D_g \subset D$ to assess. The choice of assignments for each grader can be uniformly random, or can follow a deterministic or sequential design. In either case, the number of assignments that any grader assesses $|D_g|$ is much smaller than the total number of assignments $|D|$ (*e.g.,* $|D_g| \approx 5 - 10$).

Each grader provides feedback for the set of assignments D_g they were given (to grade). Ordinal and cardinal peer grading differ in the type of feedback a grader is expected to provide:

Cardinal Peer Grading (CPG): Here, each grader g provides cardinal-valued feedback for each item $d \in D_g$. Typically, this is a numeric or categorical response which we denote as $y_d^{(g)}$ (*e.g.,* Likert scale, letter grade).

Ordinal Peer Grading (OPG): In ordinal peer grading, each grader g returns an ordering $\sigma^{(g)}$ (possibly containing ties) of his or her assignments D_g, indicating relative but not absolute quality. More generally, ordinal feedback could also consist of multiple pairwise preferences, but we focus on the case of a single ordering in this thesis.

Independent of the type of feedback that graders provide, the goal in peer grading is twofold.

The first goal *grade estimation*, is the task of estimating the true quality of the assignments in D from the grader feedback. We distinguish between two types of grade estimation, which differ by how they express assignment quality. In *ordinal grade estimation*, the goal is to infer a ranking $\hat{\sigma}$ of all assignments in D that

$G, g(\in G)$	Set of all graders, Specific grader
$D, d(\in D)$	Set of all assignments, Specific assignment
$D_g(\subset D)$	Set of assignments graded by grader g
$s_d(\in \mathfrak{R})$	Predicted grade for assignment d (larger is better)
$\eta_g(\in \mathfrak{R}^+)$	Predicted reliability of grader g
$\sigma^{(g)}$	Ranking feedback (with possible ties) from g
$r_d^{(\sigma)}$	Rank of assignment d in ordering σ (rank 1 is best)
$\rho^{(g)}$	Set of pairwise preference feedback from g
$d_2 \succ_\sigma d_1$	d_2 is preferred/ranked higher than d_1 (in σ)
$\pi(A)$	Set of all rankings over $A \subseteq D$
$\sigma_1 \sim \sigma_2$	$\exists$ way of resolving ties in σ_2 to obtain σ_1
$\hat{\sigma}$	Estimated ordering of assignments
σ^*	(Latent) True ordering of assignments

Table 8.1: Peer grading notation overview and reference.

most accurately reflects some true ordering (by quality) σ^*. In *cardinal grade estimation*, the goal is to infer a cardinal grade $\hat{s}_d$ for each $d \in D$ that most accurately reflects each true grade s_d^*. Note that the type of feedback does not necessarily determine whether the output of grade estimation is ordinal or cardinal. In particular, we will see that some of our methods can infer cardinal grades even when only provided with ordinal feedback.

The second goal is *grader reliability estimation*, which is the task of estimating how accurate the feedback of a grader is. Estimating grader reliability is important for at least two reasons. First, identifying unreliable grades allows us to downweight their feedback for grade estimation. Second, and more importantly, it allows us to incentivize good and thorough grading by making peer grading itself part of the overall grade. In the following, we will typically represent the reliability of a grader as a single number $\eta_g \in \mathfrak{R}_+$.

Section 8.3 will derive and evaluate methods for grade estimation and grader reliability estimation in the Ordinal Peer Grading setting. Table 8.1 details all the notation used in the rest of this chapter.

8.2 Relation to Prior Work

The grade estimation problem in Ordinal Peer Grading can be viewed as a specific type of rank aggregation problem. Rank aggregation (RA), as described in Section 2.4.3, is a class of problems related to combining information contained in rankings from multiple sources (*i.e.,* graders in this context). Techniques developed for the RA problem have found use in many different application domains, including educational assessment. For instance, [16] introduces a graphical model based approach for modeling the difficulty of multiple-choice questions and estimating the correct answers in a crowdsourced setting. However these approaches are neither applicable for a peer grading setting nor can they handle open-ended answers (like essays).

More generally, conventional rank aggregation differs from Ordinal Peer Grading in several aspects. First, grader reliability estimation is not a goal in itself in conventional rank aggregation. In fact most existing RA approaches assume all the sources (*i.e.,* graders) to be equally reliable. Second, the success in most RA problems depends on correctly identifying the top items of the ranking, unlike in grade estimation where the goal is to accurately estimate the full ranking of assignments. Third, ties and data sparsity are not an issue in many RA problems (such as search result aggregation), since (at least in principle) input rankings are total orders over all results.

Crowdsourcing using rank aggregation is perhaps the most closely related application domain. Here the goal is to merge the feedback obtained from multiple *crowdworkers* [23, 79, 171]. Due to the differing quality of these workers, modeling the worker reliability is essential [45, 143]. The key difference in our

Algorithm 12: **Normal Cardinal-Score (NCS)** Algorithm (called $\mathbf{PG}_1$ in [126]) is used as a baseline in our subsequent experiments.

$s_d \sim \mathcal{N}(\mu_0, \frac{1}{\gamma_0})$	▷ True Scores
$\eta_g \sim Gamma(\alpha_0, \beta_0)$	▷ Grader Reliability
$b_g \sim \mathcal{N}(0, \frac{1}{\gamma_1})$	▷ Grader Bias (Only for NCS+G)
$y_d^{(g)} \sim \mathcal{N}(s_d + b_g, \frac{1}{\eta_g})$	▷ Observed Cardinal Peer Grade
Estimate $\hat{s}_d, \hat{\eta}_g$ and $\hat{b}_g$	▷ Using Maximum Likelihood Estimation (MLE)

setting is that the number of items is large and we would like to correctly order **all** of them, not just identify the top-few.

8.2.1 Prior work on Peer Grading

With the advent of online courses peer grading has seen increased usage in large classes with mixed results [28, 39, 120, 165]. Part of the problem has been the use of simple estimation techniques like averaging cardinal feedback grades/scores. More recently, probabilistic learning algorithms have been proposed for peer grade estimation [126]. However, this method requires that students provide *cardinal* scores as grades. This in turns requires the precise communication absolute grading scales to all students, which is very challenging. A second limitation of the method introduced in [126], is that it incentivizes grader reliability by relating it to the grader's own assignment score. However, such a setup is inappropriate when there are groups (such as the classroom setting studied in this dissertation) or where external graders/reviewers are used (*e.g.*, conference reviewing). In addition, such an indirect incentive is harder to communicate and justify compared to the direct grader reliability estimates used in the ap-

proaches introduced in this dissertation. Lastly their approach requires that each student grades some assignments that were previously graded by the instructor in order to estimate grader reliability. This seems wasteful, given that students are only able to grade a small number of assignments in total. We empirically compare their cardinal peer grading technique (Algorithm 12, using Maximum Likelihood Estimation – MLE – instead of Gibbs sampling) with the ordinal peer grading techniques proposed in this dissertation.

Overall, given the limited amount of attention that the peer grading problem has received in the machine learning literature so far, there is ample opportunity to improve on the state-of-the-art and address current shortcomings [144].

8.3 Ordinal Peer Grading Methods

This section introduces ordinal peer grading methods for grade estimation and then extends these methods to also tackle the problem of grader reliability estimation (Sec 8.3.6). The proposed methods are efficient and simple to implement. They all begin by taking in as input an i.i.d. sample of orderings

$$S = (\sigma^{(g_1)}, ..., \sigma^{(g_{|G|})}), \tag{8.1}$$

where each ordering sorts a subset of assignments according to the judgment of grader g_i. The proposed grade estimation methods are based on models that represent probability distributions over rankings. In particular, we extend the Mallows Model (Sec 8.3.1), the Bradley-Terry model (Sec 8.3.3), the Thurstone model (Sec 8.3.4), and the Plackett-Luce model (Sec 8.3.5) as appropriate for the ordinal peer grading problem.

8.3.1 Mallows Model (MAL and MALBC)

Mallows model [116] describes a distribution over rankings σ in terms of the distance $\delta(\bar{\sigma}, \sigma)$ from a central ranking $\bar{\sigma}$, which in our setting is the true ranking σ^* of assignments by quality.

$$P(\sigma|\bar{\sigma}) = \frac{e^{-\delta(\bar{\sigma},\sigma)}}{\sum_{\sigma'} e^{-\delta(\bar{\sigma},\sigma')}} \tag{8.2}$$

While maximum likelihood estimation of σ^* given observed rankings is NP-hard for many distance functions [57, 129], tractable approximations are known for special cases. In this work we use the following tractable **Kendall-τ distance** [93], which assumes that both rankings are total orderings over all assignments.

Definition 1 *We define the Kendall-τ Distance δ_K between rankings σ_1 and σ_2 as*

$$\delta_K(\sigma_1, \sigma_2) = \sum_{d_1 \succ_{\sigma_1} d_2} \mathbb{I}[[d_2 \succ_{\sigma_2} d_1]] \tag{8.3}$$

It measures the number of incorrectly ordered pairs between the two rankings. In our case, the rankings that students provide can have ties. We interpret these ties as *indifference* (*i.e.*, agnostic to either ranking), which leads to the following model, where the summation in the numerator is over all total orderings σ' consistent with the weak ordering σ.

$$P(\sigma|\bar{\sigma}) = \frac{\sum_{\sigma' \sim \sigma} e^{-\delta(\bar{\sigma},\sigma')}}{\sum_{\sigma'} e^{-\delta(\bar{\sigma},\sigma')}} \tag{8.4}$$

Note also that the input ranking σ may only sort a subset of assignments. In such cases, we appropriately restrict the normalization constant in Eqn. 8.4 [1]. For the Kendall-τ distance, this normalization constant can be computed very

[1] While the Mallows model typically involves an additional dispersion parameter that scales the distance function, for the purpose of simplicity we ignore this for the time being.

Algorithm 13: Computing MLE ranking for Mallows Model

$C \leftarrow D$ ▷ C contains unranked items

for $i = 1 \ldots |D|$ **do**

  for $d \in C$ **do**

    $x_d \leftarrow \sum_{g \in G} \eta_g |d' \in C : d' \succ_{\sigma_g} d| - |d' \in C : d \succ_{\sigma_g} d'|$

  $d^* \leftarrow \min_{d \in C} x_d$ ▷ Select highest scoring item

  $r_{d^*}^{(\hat{\sigma})} \leftarrow i$ ▷ Rank as next item

  $C \leftarrow C / d^*$ ▷ Remove d^* from candidate set

return $\hat{\sigma}$

efficiently, as it only depends on the number of elements in the ranking.

$$Z_M(k) = \prod_{i=1}^{k} \left(1 + e^{-1} + \cdots + e^{-(i-1)}\right) = \prod_{i=1}^{k} \frac{1 - e^{-i}}{1 - e^{-1}}$$

The numerator can likewise be computed efficiently via a similar trick. Note that ties in the grader rankings $\sigma^{(g)}$ do not affect the normalization constant under the interpretation of indifference.

Under this modified Mallows model, the Maximum Likelihood Estimator (MLE) of the central ranking $\hat{\sigma}$ is

$$\hat{\sigma} = \texttt{argmax}_{\sigma} \left\{ \prod_{g \in G} \frac{\sum_{\sigma' \sim \sigma^{(g)}} e^{-\delta_K(\sigma, \sigma')}}{Z_M(|D_g|)} \right\}. \tag{8.5}$$

Computing the MLE $\hat{\sigma}$ as an estimate of the true ranking by quality σ^* requires finding the *Kemeny-optimal aggregate* [172], which is known to be NP-hard [57]. However numerous approximations have been studied in the rank aggregation literature [6, 7, 57, 59, 60, 94]. In this work we use a simple greedy algorithm as shown in Algorithm 13.

As an alternative algorithm for computing the estimated ranking, we utilize a Borda count-like approximation for the Mallows model (which we denote as MAL_{BC}), where Line 13 of Algorithm 13 is replaced with:

$$x_d \leftarrow \sum_{g \in G} r_d^{(\sigma^{(g)})}.$$

In other words, this orders as per the average rank of an item (leading to a 5-approximation for the case of full rankings [59]). We also experimented with techniques such as *Local Kemenization* (*i.e.,* adjacent pairs are swapped in a bubble-sort like manner to increases likelihood [57]), but exclude these results for brevity.

8.3.2 Score-Weighted Mallows (MALS)

The Mallows model presented above has two shortcomings. First, it does not output a meaningful cardinal grade for the assignments, which makes it applicable only to ordinal grade estimation. Second, the distance δ_K does not distinguish between misordering assignments that are similar in quality from those that have a large quality difference.

To address these two shortcomings, we propose an extension which estimates cardinal grades $\hat{s}_d$ for all assignments. To this effect, we introduce the following score-weighted ranking distance, which scales the distance induced by each misranked pairs by its estimated grade difference.

Definition 2 *The* **score-weighted Kendall-τ distance** δ_{SK} *over rankings* σ_1, σ_2 *given cardinal scores* s_d *is*

$$\delta_{SK}(\sigma_1, \sigma_2 | s) = \sum_{d_1 \succ_{\sigma_1} d_2} (s_{d_1} - s_{d_2}) \mathbb{I}[[d_2 \succ_{\sigma_2} d_1]]. \tag{8.6}$$

Treating ties in the grader rankings as described above results in a score-weighted version of the Mallows model (MALS). We use the following maximum a posteriori estimator to estimate the scores $\hat{s}$.

$$\hat{s} = \texttt{argmax}_s \left\{ Pr(\mathbf{s}) \prod_{g \in G} \frac{\sum_{\sigma' \sim \sigma^{(g)}} \exp\left(-\delta_{SK}(\hat{\sigma}, \sigma' | s)\right)}{\sum_{\sigma' \in \pi(D_g)} \exp\left(-\delta_{SK}(\hat{\sigma}, \sigma' | s)\right)} \right\} \tag{8.7}$$

Note that $\hat{\sigma}$ can be obtained by sorting items as per $\hat{s}_d$. $Pr(\hat{\mathbf{s}}) = \prod_{d \in D} Pr(\hat{s}_d)$ is the prior on the latent item scores. In our experiments we model $Pr(\hat{s}_d) \sim \mathcal{N}(0, 9)$. The same prior is used in all of our methods. While the resulting objective is not necessarily convex, we use Stochastic Gradient Descent (SGD) for grade estimation and initialize the grades using a scaled-down Mallows solution.

8.3.3 Bradley-Terry Model (BT)

The above Mallows based models define distributions over rankings as a function of a ranking distance, and require approximate methods for solving the maximum likelihood problem. As an alternative, we can utilize rank aggregation models based on distributions over pairwise preferences, since a ranking of n items can also be viewed as a set of preferences over the $\binom{n}{2}$ item pairs. Using pairwise models can further simplify the grader feedback process as it is cognitively less demanding on the students to break their ordinal assessment task into pairwise comparisons [98], especially if the number of items to assess is large. The Bradley-Terry model [29] is one such model for pairwise preferences, and it derives a distribution based on the differences of underlying item scores s_d through a logistic link function.

$$P(d_i \succ_{\rho^{(g)}} d_j | s) = \frac{1}{1 + e^{-(s_{d_i} - s_{d_j})}} \tag{8.8}$$

Since each preference decision is modeled individually, the feedback from the grader could be a (possibly inconsistent) set of preferences that does not necessarily have to form a consistent ordering. The following is the Maximum a Posteriori (MAP) estimator used in this paper.

$$\hat{s} = \texttt{argmax}_s \left\{ Pr(\mathbf{s}) \prod_{g \in G} \prod_{d_i \succ_{\rho(g)} d_j} \frac{1}{1 + e^{-(s_{d_i} - s_{d_j})}} \right\} \tag{8.9}$$

The resulting objective is (jointly) log-convex in all of the estimated grades $\hat{s}_d$, with the gradients taking a simple form. Hence SGD can be used to estimate the global optimal grades efficiently. We treat ties as the absence of a preference. One can also extend this model (as well as subsequent pairwise models) to incorporate ties more explicitly, but we do not discuss this for the sake of brevity.

8.3.4 Thurstone Model (THUR)

An alternate to the logistic link function of the Bradley-Terry model is to utilize a normal distribution for the pairwise preferences. Like the Bradley-Terry model, the resulting model (*i.e.,* the Thurstone model [161]) can be understood as a random utility model using the following process: For each pair of items d_i, d_j, the grader samples (latent) values $x_{d_i}^{(g)} \sim \mathcal{N}(s_{d_i}, \frac{1}{2})$ and $x_{d_j}^{(g)} \sim \mathcal{N}(s_{d_j}, \frac{1}{2})$, and then orders the pair based on the two values. The mean of the normal distribution of d_i is the quality s_{d_i}. Maximum a posteriori (MAP) estimation of the scores s requires maximization of the following function:

$$\hat{s} = \texttt{argmax}_s \left\{ Pr(\mathbf{s}) \prod_{g \in G} \prod_{d_i \succ_{\rho(g)} d_j} \mathcal{F}(s_{d_i} - s_{d_j}) \right\} \tag{8.10}$$

$\mathcal{F}$ is the Cumulative Distribution Function (CDF) of the standard normal distribution. This objective function is log-convex and can be optimized using SGD.

8.3.5 Plackett-Luce Model (PL)

A drawback of the pairwise preference models is that they can be less expressive than models built on distributions over rankings. An extension to the Bradley-Terry model (the Plackett-Luce model [127]) allows us to use distributions over rankings, while still retaining convexity and simplicity of gradient computation. This model can be best understood as a multi-stage experiment where at each stage, an item d_i is drawn (w/o replacement) with probability $\propto e^{s_{d_i}}$. The resulting probability of observing ranking $\sigma^{(g)}$ under this process is:

$$P(\sigma^{(g)}|s) = \prod_{d_i \in D_g} e^{s_{d_i}} / \left(e^{s_{d_i}} + \sum_{d_i \succ_{\sigma^{(g)}} d_j} e^{s_{d_j}} \right)$$

The resulting maximum a posteriori (MAP) estimator is:

$$\hat{s} = \texttt{argmax}_s \left\{ Pr(\mathbf{s}) \prod_{g \in G} \prod_{d_i \in D_g} \frac{e^{s_{d_i}}}{e^{s_{d_i}} + \sum_{d_i \succ_{\sigma^{(g)}} d_j} e^{s_{d_j}}} \right\}. \tag{8.11}$$

8.3.6 Grader Reliability Estimation for all Methods

While the methods discussed above allow us to estimate assignment grades from ordinal feedback, they still do not give us the means to directly estimate grader reliabilities $\hat{\eta}_g$. However, there is a generic way of extending all methods presented above to incorporate grader reliabilities. Using Mallows model as an

Algorithm 14: Alternating SGD-based Minimization

Require: $N \geq 0$ (Number of iterations), Likelihood L

$Obj \leftarrow -\log L$

$\hat{\mathbf{s}} \leftarrow SGD_{Scores}(Obj, \eta = \mathbf{1})$ ▷ Est. scores w/o reliabilities

for $i = 1 \ldots N$ **do**

  $\eta \leftarrow SGD_{Reliabilities}(Obj, \hat{\mathbf{s}})$ ▷ Estimate reliabilities

  $\hat{\mathbf{s}} \leftarrow SGD_{Scores}(Obj, \eta)$ ▷ Est. scores with reliabilities

return $\hat{\mathbf{s}}, \eta$

example, we can introduce $\hat{\eta}_g$ as a variability parameter as follows:

$$Pr(\sigma|\bar{\sigma}, \eta_g) = \frac{\sum_{\sigma' \sim \sigma^{(g)}} \exp\left(-\eta_g \delta_K(\bar{\sigma}, \sigma')\right)}{Z_M(\eta_g, |D_g|)} \tag{8.12}$$

The resulting estimator of both $\hat{\sigma}$ and $\hat{\eta}$ is

$$\hat{\sigma}, \hat{\eta} = \texttt{argmax}_{\sigma,\eta} \left\{ \prod_{g \in G} Pr(\eta_g) \frac{\sum_{\sigma' \sim \sigma^{(g)}} \exp\left(-\eta_g \delta_K(\sigma, \sigma')\right)}{Z_M(\eta_g, |D_g|)} \right\}, \tag{8.13}$$

where $Pr(\hat{\eta}_g)$ is the prior on the grader reliability. In this work we use a *Gamma* prior $\hat{\eta}_g \sim Gamma(10, 0.1)$ for all the methods in our experiments.

Similarly, the other objectives can also be extended in this manner as seen in Table 8.2. While many of the extended objectives, such as the one above in Eq. (8.13), are convex in the grader reliabilities $\hat{\eta}_g$ (for given $\hat{\sigma}$), they unfortunately are not jointly convex in the reliabilities *and* the estimated grades. We thus use an iterative alternating-minimization technique, which alternates between minimizing the log-objective to estimate the assignment grades and minimizing the log-objective to estimate the grader reliabilities. This iterative alternating approach using stochastic gradient descent is used for all joint estimation tasks in this paper. Note that methods which estimate the reliabilities using Al-

Method	Score?	Convex?	Estimator
MAL+G	No	No	$Pr(\eta) \prod_{g \in G} \sum_{\sigma' \sim \sigma^{(g)}} \exp\left(-\hat{\eta}_g \delta_K(\hat{\sigma}, \sigma')\right) / Z_M(\hat{\eta}_g, \lvert D_g \rvert)$
MALS+G	Yes	No	$Pr(\hat{\mathbf{s}}, \eta) \prod_{g \in G} \sum_{\sigma' \sim \sigma^{(g)}} \exp\left(-\hat{\eta}_g \delta_{SK}(\sigma^{(g)}, \hat{\sigma}, F)\right) / Z(\cdot)$
BT+G	Yes	Yes	$Pr(\hat{\mathbf{s}}, \eta) \prod_{g \in G} \prod_{d_i \succ_{\rho(g)} d_j} 1/(1 + e^{-\hat{\eta}_g (s_{d_i} - s_{d_j})})$
THUR+G	Yes	Yes	$Pr(\hat{\mathbf{s}}, \eta) \prod_{g \in G} \prod_{d_i \succ_{\rho(g)} d_j} \mathcal{F}(\sqrt{\hat{\eta}_g}(s_{d_i} - s_{d_j}))$
PL+G	Yes	Yes	$Pr(\hat{\mathbf{s}}, \eta) \prod_{g \in G} \prod_{d_i \in D_g} 1/(1 + \sum_{d_i \succ_{\rho(g)} d_j} e^{-\hat{\eta}_g (s_{d_i} - s_{d_j})})$

Table 8.2: Summary of ordinal methods studied which model the grader's reliabilities, including the ability to output cardinal scores and if the resulting objective is convex in these scores.

gorithm 14 are denoted by a **+G** suffix to the method, while those that simply estimate the assignment grades are represented by the method name alone.

8.4 Evaluation

In the following we present experiments that compare ordinal and cardinal peer grading methods. We evaluate their ability to predict instructor grades, their variability, their robustness to bad peer grading, and their ability to identify bad graders. We also present the results from a qualitative student survey to evaluate how students perceived the peer grading process.

8.4.1 Data Collection in Classroom Experiment

We collected and used a real dataset consisting of peer feedback, TA grades, and instructor grades for evaluating the peer grading methods proposed in this dissertation. This data was collected as part of a senior-undergraduate and masters-level class with an enrollment of about 170 students. The class was

staffed with 9 Teaching Assistants (TAs) that participated in grading, and a single Instructor. This size of class is attractive, since it is large enough for collecting a substantial number of peer grades, while at the same time allowing traditional instructor and TA grading to serve as a baseline. The availability of instructor grades makes our data different from other peer-grading evaluations used in the past (*e.g.,* [126]).

The dataset consists of two parts that were graded independently, corresponding to the *poster presentation* and the *final report* stages of an 8-week long course project. Students worked in groups of 3-4 students for the duration of the project, and there were a total of 44 project groups. While student worked in groups, peer grading was performed individually via the Microsoft Conference Management Toolkit (CMT) system. The peer grading process was performed single-blind for the posters and double-blind for the reports. The reviewer assignments were made uniformly at random. Students were given clear directives and asked to focus on aspects such as *novelty* and *clarity* (among others) while determining their grade. They were also asked to justify their grade by providing feedback comments. Students were told that a part of their grade depends on the quality of their peer feedback.

All grading was done on a 10-point (cardinal) Likert scale, where 10 was labeled "perfect", 8 "good", 5 "borderline", 3 "deficient" and 1 "unsatisfactory". This will allow us to compare cardinal and ordinal peer grading methods, where ordinal methods merely use the ordering (possibly with ties) implied by the cardinal scores. Note that in a true application of ordinal peer grading, accuracy could improve since it would allow simplifying the grading instructions and reduce cognitive overhead if students did not have to worry about the precise

Data Statistic	PO	FR
Number of Assignments	42	44
Number of Peer Reviewers	148	153
Total Peer Reviews	996	586
Total TA Reviews	78	88
Participating TAs	7	9
Per-Item Peer Grade Devn.	1.16	1.03

Set	Who?	Mean	Devn.
PO	Peers	8.16	1.31
	TAs	7.46	1.41
	Meta	7.55	1.53
FR	Peers	8.20	1.35
	TAs	7.59	1.30
	Instructor	7.43	1.16

Table 8.3: Statistics for the two datasets (PO=Poster, FR=Report) from the classroom experiment along with the staff (TAs/Meta/Instructor) and student grade distributions.

meaning of specific cardinal grades.

The following describes the grading processes used at the two project stages, and Table 8.3 summarizes some of the key statistics.

Grading Process for Poster Presentations

The poster presentations took place in a two-hour poster session. Two groups did not present their poster. Students were encouraged to rotate presenting their poster within their project group members. This likely increased variability of grades, since different reviewers often saw different presenters. Students and TAs took notes and entered their reviews via CMT afterwards.

The *TA Grades* were independent, meaning that the TAs did not see the peer reviews before entering their review. There were on average 1.85 TA reviews for each poster.

The *Peer Grades* totaled on average 23.71 reviews for each poster, with each peer reviewer reviewing 6.73 posters on average.

The final *Meta Grade* for each poster was determined as follows. One of

the TAs that already provided an independent review was selected as a meta-reviewer. This TA was asked to aggregate all the arguments brought forward in the reviews and make a final grade on the same 10-point scale. The instructor oversaw this process, but intervened only on very few grades.

Grading Process for Final Projects

At the end of the project, groups submitted a report of about 10 pages in length. The reviewing process was similar to that of the poster presentations, but with one important difference — namely that all project reports were graded by the TAs and the instructor without any knowledge of the peer reviews, as detailed below.

On average each report received 13.32 *Peer Grades* as the overall score on each of the peer reviews (students were also asked for component scores like "clarity", etc.).

Each report also received two *TA Grades*, which the TAs submitted without knowledge of the peer reviews.

Finally, each report received an *Instructor Grade*, following the traditional process of project grading in this class. The instructor and head TA each graded half the projects and determined the grade based on their own reading of the paper, taking the TA reviews as input. These grades were provided without viewing the peer reviews. We can therefore view the instructor grades as an assessment that is entirely independent of the peer grades (in contrast to the Meta Grades for the posters, which have some dependency).

8.4.2 Evaluation Metrics

A commonly used measure for reporting student performance (among many standardized tests) is the percentile rank relative to all students in the class. Following this practice, we use percentile rank as the grade itself (a letter grade can easily be derived via curving), and report ranking metrics as our main indicators of performance (we investigate cardinal grading performance in Sec. 8.4.8). In particular, we use the following variant of Kendall-τ that accounts for ties.

$$\tau_{KT}(\sigma_1, \sigma_2) = \sum_{d_1 \succ_{\sigma_1} d_2} \mathbb{I}[[d_2 \succ_{\sigma_2} d_1]] + \frac{1}{2}\mathbb{I}[[d_1 \approx_{\sigma_2} d_2]] \tag{8.14}$$

Note that this measure is not symmetric, assuming that the first argument is a target ranking and the second argument is a predicted ranking. It treats ties in the target ranking as *indifference*. Ties in the predicted ranking are treated as a lack of information, incurring a $\frac{1}{2}$ error (*i.e.*, equivalent to breaking ties randomly). Such a correction is necessary for evaluation purposes, since otherwise predicted rankings with all ties (which convey no information) would incur no error. Normalizing $\tau_{KT}(\sigma_1, \sigma_2)$ and accounting for the fact that we may have more than one target ranking[2] leads to the following error measure.

Definition 3 *Given a set of target rankings S_g, we define the* **Kendall-τ error** $\mathcal{E}_K$ *of predicted ranking σ_I as:*

$$\mathcal{E}_K(\sigma_I) = \frac{100}{|S_g|} \sum_{\sigma_t \in S_g} \frac{\tau_{KT}(\sigma_t, \sigma_I)}{\max_{\sigma \in \pi(D)} \tau_{KT}(\sigma_t, \sigma)} \tag{8.15}$$

This error *macro-averages* the (normalized) τ_{KT} errors for each target ranking. Due to the normalization, they lie between 0 (indicating perfect agreement) and

[2]We may have more than one target ranking if the ground-truth rankings were only over subsets of items, as is the case in our case study.

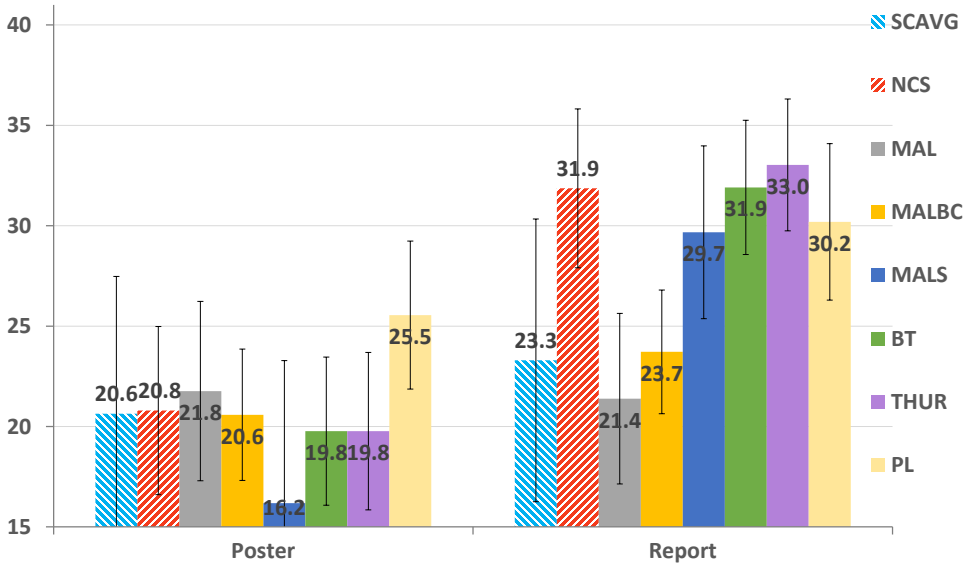

Figure 8.1: Comparing peer grading methods (w/o grader reliability estimation) against Meta and Instructor Grades in terms of $\mathcal{E}_K$ (lower is better).

100% (indicating reversal with target rankings). A random ranking has expected $\mathcal{E}_K$ error of 50%.

8.4.3 How well do Ordinal and Cardinal Peer Grading methods predict the final grade?

The first question we address is in how far peer grading resembles the grades given by an instructor. Specifically, we investigate whether ordinal peer grading methods achieve similar performance as cardinal peer grading methods, even though ordinal methods receive strictly less information.

For all methods considered in this paper, Figure 8.1 shows the Kendall-τ error $\mathcal{E}_K$ compared to the Meta Grades for the Posters, and compared to the Instructor Grades for the Reports. The errorbars show estimated standard devi-

ation using bootstrap-type resampling.

On the posters, none of the methods show significantly worse performance than another method. In particular, there is no evidence that the cardinal methods are performing better than the ordinal methods. A similar conclusion also holds for the reports. However, here the ordinal methods based on Mallows model perform better than the cardinal NCS method[3] [126] (see Algorithm 12), as well as some of the other ordinal methods. Simply averaging the cardinal scores of the peer graders, which we call Score Averaging (SCAVG), performs surprisingly well.

In summary, most methods achieve an $\mathcal{E}_K$ between 20% and 30% on both problems, but all have large standard deviations. The $\mathcal{E}_K$ appears lower for the posters than for the projects, which can be explained by the fact that the Meta Grade was influenced by the peer grades. But how good is an $\mathcal{E}_K$ between 20% and 30%?

8.4.4 How does Peer Grading Compare to TA Grading?

We now consider how Peer Grading compares to having each assignment graded by a TA. For medium sized classes, TA grading may still be feasible. It is therefore interesting to know if TA grading is clearly preferable to Peer Grading when it is feasible. But more importantly, the inter-judge agreement between multiple TAs can give us reference points for the accuracy of Peer Grading.

As a first reference point, we estimate how well the TA Grades reflect the

[3] We tuned the hyperparameters of the NCS model to maximize performance. We also used a fixed grader reliability parameter in the NCS model, since it provided better performance than with reliability estimation (NCS+G).

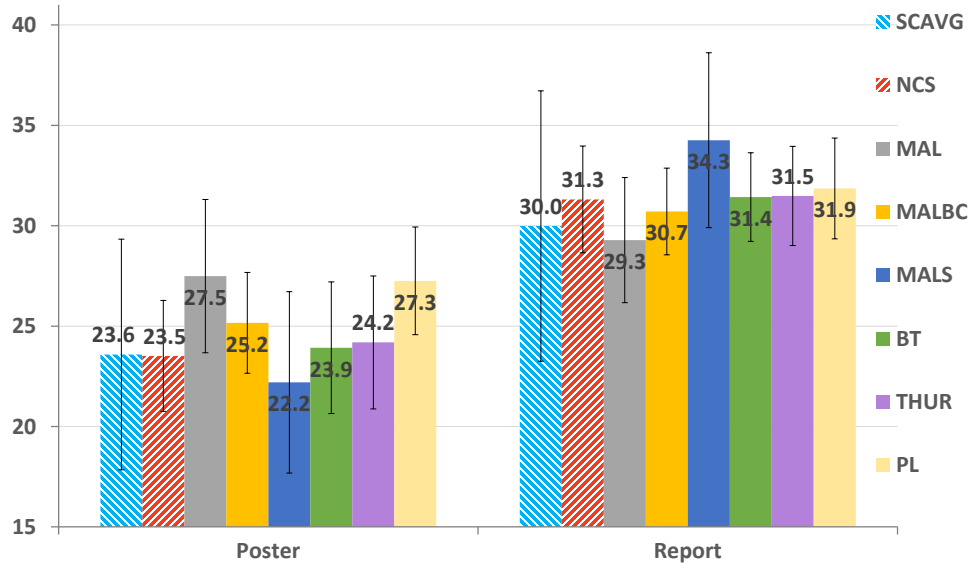

Figure 8.2: Comparing peer grading methods (w/o grader reliability estimation) against TA Grades in terms of $\mathcal{E}_K$, using TA grades as the target ranking.

Meta Grades for the posters and the Instructor Grades for the reports. In particular, we consider a grading process where each assignment is graded by a single TA that assigns a cardinal grade. Each TA grades a fraction of the assignments, and a final ranking of the assignments is then computed by sorting all cardinal grades. We call this grading process *TA Grading*.

We can estimate the $\mathcal{E}_K$ of TA grading with the Meta Grades and the Instructor Grades, since we have multiple TA grades for most assignments. We randomly re-sample a TA grade from the available grades for each assignment, compute the ranking, and then estimate mean and standard deviation of the $\mathcal{E}_K$ over 5000 samples. This leads to a mean $\mathcal{E}_K$ of 22.0 ± 16.0 for the posters and 22.2 ± 6.8 for the reports. Comparing these to the $\mathcal{E}_K$ of the peer grading methods in Figure 8.1, we see that they are comparable to the performance of many peer grading methods — *even though the $\mathcal{E}_K$ of TA grading is favorably biased. Note that Meta Grades and the Instructor Grades were assigned based on the same TA grades*

we are evaluating against.

To avoid this bias and provide a fairer comparison with TA grading, we also investigated how consistent peer grades are with the TA grades, and how consistent TA grades are between different TAs. Figure 8.2 shows the $\mathcal{E}_K$ of the peer grading methods when using TA Grades as the target ranking for both the Posters and the Reports. Variances were again estimated via bootstrap resampling. Note that TA Grades were submitted without knowledge of the Peer Grades. Overall, the peer grades have an $\mathcal{E}_K$ with the TA Grades that is similar to the $\mathcal{E}_K$ with the respective Final grades considered in the previous subsection. Again, there is no evidence that the ordinal peer grading methods are less predictive of the TA Grades than the cardinal peer grading methods.

To estimate $\mathcal{E}_K$ between different TAs, we use the following resampling procedure. In a leave-one-out fashion, we treat the grades of a randomly selected TA as the target ranking and compute the predicted ranking by sampling from the other TAs grades as described above. Averaging over 5000 repetitions reveals that the $\mathcal{E}_K$ between the TAs is 47.5 ± 21.0 for the posters and 34.0 ± 13.8 for the reports.

These numbers can be compared to the $\mathcal{E}_K$ of peer grading methods in Figure 8.2. For the Reports, peer grades are roughly as consistent with the TA grades as other TA grades are. For the posters the peer grading methods are substantially more predictive of TA grades than other TA grades. The reason for this is at least twofold. First, the peer grading methods have access to much more data, which reduces variability (especially since presentations were not always given by the same student). Second, the peer grading methods have enough data to correct for different grading scales, while offsets in grading scales can

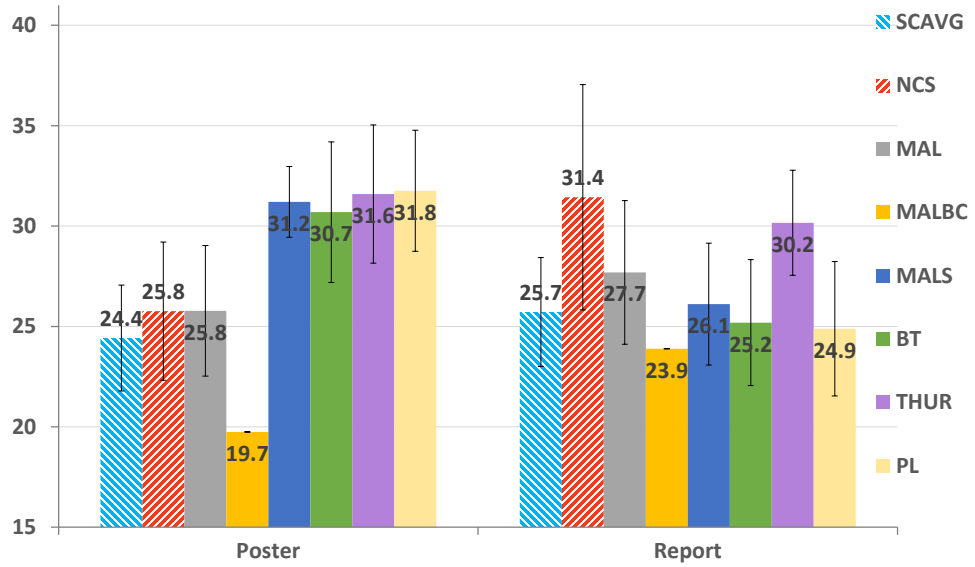

Figure 8.3: Self-consistency of peer-grading methods (w/o grader reliability estimation) in terms of $\mathcal{E}_K$.

have disastrous consequences in TA grading.

Finally, we also consider the self-consistency of the peer grading methods. Analogous to the self-consistency of TA grading, we ask how similar are the grades we get if we repeat the grading procedure with a different sample of assessments. We randomly partition peer reviewers into two equally sized datasets. For each peer grading method, we perform grade estimation on both datasets, which generates two rankings of the assignments. Ties in these rankings are broken randomly to get total orderings. Figure 8.3 shows the $\mathcal{E}_K$ between the two rankings (over 20 sampled partitions). For the posters, peer grading is substantially more self consistent than TA grading, and for the reports all peer grading methods have lower $\mathcal{E}_K$ estimates than TA grading as well.

Overall, we conclude that there is no evidence that TA grading would have led to more accurate grading outcomes than peer grading.

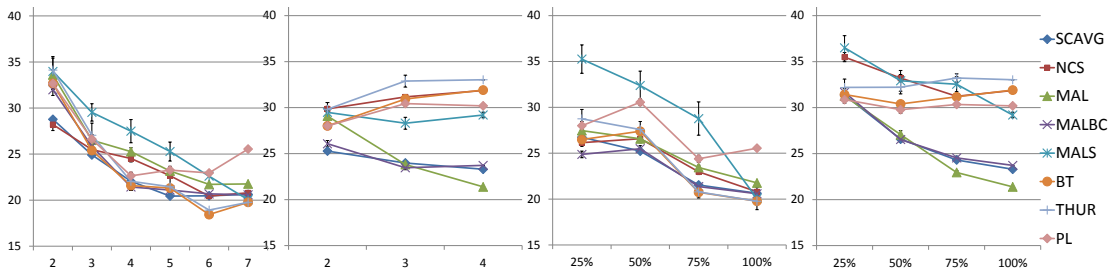

Figure 8.4: Change in $\mathcal{E}_K$ performance of peer grading methods (using Meta and Instructor Grades as target ranking) when varying the number of assignments assigned to each reviewer for Posters (first from left) & Reports (second), and when varying the number of peer reviewers for Posters (third), Reports (last).

8.4.5 How does Grading Accuracy Scale with the Number of Peer Reviews?

How many reviewers are necessary for accurate peer grading, and how many reviews does each peer grader need to do? To gauge how performance changes with the number of peer reviews, we performed two sets of experiments. First, we created 20 smaller datasets by downsampling the number of peer reviewers. The results are shown in the two rightmost graphs of Figure 8.4. Overall, the methods degrade gracefully when the number of reviewers is reduced. Furthermore, we find that most ordinal methods scale as well as cardinal methods, if not better, on both datasets.

A second way of increasing or reducing the amount of available data lies in the number of assignments that each student grades. Thus we repeated the experiment, but instead downsampled the number of assignments per reviewer (corresponding to a lower workload for each grader). The leftmost two plots of Figure 8.4 show the results, with performance again degrading gracefully.

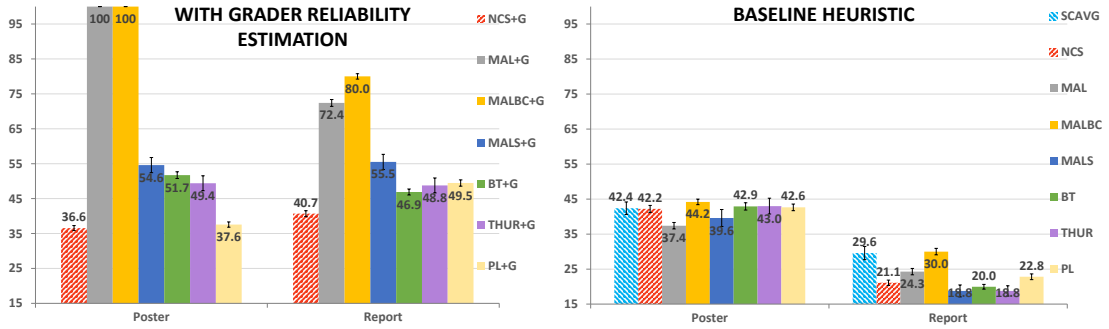

Figure 8.5: Percentage of times a grader who randomly scores and orders assignments is among the 20 least reliable graders.

8.4.6 Can Peer Grading Methods Identify Unreliable Graders?

Peer grading can only work in practice, if graders are sufficiently incentivised to report an accurate assessment. This can be achieved by giving a grade also for the quality of the grading. In the following, we investigate whether the grader reliability estimators proposed in Section 8.3.6 can identify graders that are not diligent.

For both the posters and the projects, we add 10 "lazy" peer graders that report random grades drawn from a normal distribution whose mean and variance matches that of the rest of the graders[4]. For the ordinal methods, this results in a random ordering. We then apply the peer grading methods, estimating the reliability parameters η_g for each grader using 10 iterations of the alternating optimization algorithm. We then rank graders by their estimated η_g.

Figure 8.5 (left) shows the percentage of lazy graders that rank among the 20 graders with the lowest η_g. The error bars show standard error over 50 repeated runs with different lazy graders sampled. Most ordinal methods significantly outperform the cardinal NCS method for both the posters and the reports. The

[4]Otherwise it would be easy to identify these graders.

variants of Mallows model perform very well, identifying around 70-80% of the lazy graders for the reports and all 10 lazy graders for the posters. The better performance for the posters than for the reports was to be expected, since students provide 7 instead of 4 grades.

Figure 8.5 (right) shows the results of a heuristic baseline. Here, grade estimation without reliability estimation is performed, and then graders are ranked by their $\mathcal{E}_K$ with the estimated ranking $\hat{\sigma}$. For almost all methods, this performs worse, clearly indicating that reliability estimation is superior in identifying lazy graders. We find similar results even when there are 100+ lazy graders, as we investigate robustness in the following experiment.

8.4.7 How Robust are Peer Grading Methods to Lazy Graders?

While Section 8.4.6 showed that reliability estimation in ordinal peer grading is well-suited for identifying lazy graders, we would also like to know what effect these lazy graders have on grade estimation performance. We study the robustness of the peer grading methods by adding an increasing number of lazy graders. Figure 8.6 shows the change in $\mathcal{E}_K$ (w.r.t. Instructor/Meta grades) after adding 10/50/100 lazy graders (compared to the $\mathcal{E}_K$ with no lazy graders). We find that in most cases performance does not change much relative to the variability of the methods. Interestingly, in some cases performance also improves on adding this noise. A deeper inspection reveals that noise is most beneficial for methods whose original $\mathcal{E}_K$ performance was weaker than that of the other methods. For example, the Thurstone model showed the weakest performance on the Reports and improves the most.

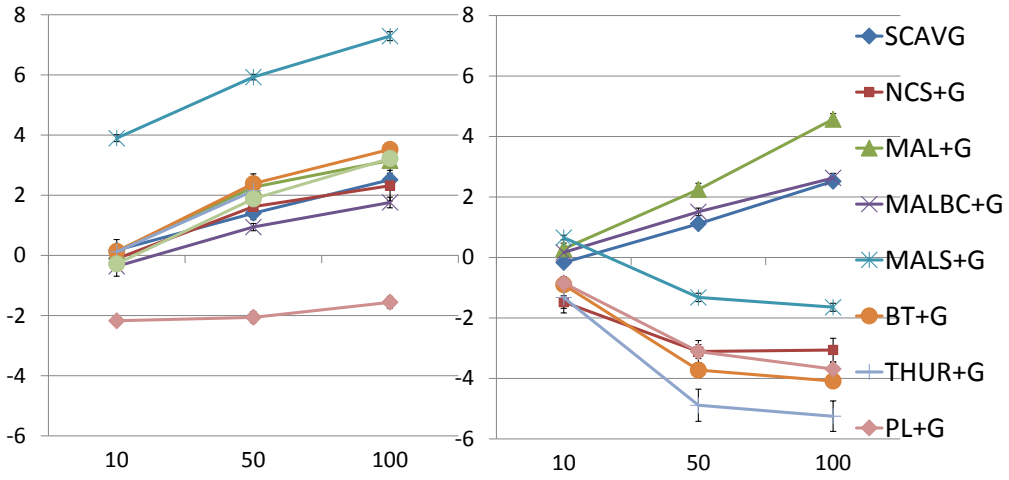

Figure 8.6: Change in $\mathcal{E}_K$ (using Instructor and Meta Grades as target ranking) for (Left) Posters and (Right) Final Reports with the addition of an increasing number of *lazy* graders *i.e.*, $\mathcal{E}_K$(With Lazy) – $\mathcal{E}_K$(Without Lazy). A negative value indicates that performance improves on adding this noise.

8.4.8 Can Ordinal Grading Methods estimate Cardinal Grades?

While the previous sections showed that the ordinal peer grading methods are able to predict the assignment ordering quite well, in this section we explore how well they do at predicting cardinal grades. We first rescale the grades output by all the different methods ($\hat{s}_d$) to have identical mean and deviation as the instructor/meta grades to make all the scores comparable. We measure the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for these rescaled scores, using the instructor/meta grades as labels.

The results are shown in Table 8.4. The results indicate that, despite not receiving any cardinal feedback, the ordinal techniques are able to predict meta/instructor grades nearly as well as the cardinal peer grading methods. Furthermore, when performing the same rescaling and metric computation for the TA grades, we find that the peer grading methods do comparably to the

Method	Poster		Report	
	MAE	RMSE	MAE	RMSE
SCAVG	0.60	0.76	0.74	1.00
NCS	0.64	0.78	0.84	1.15
MALS	0.63	0.81	0.89	1.18
BT	0.64	0.78	0.90	1.24
THUR	0.64	0.78	0.92	1.29
PL	0.68	0.83	0.89	1.23
TAs	0.66	0.98	0.73	0.96

Table 8.4: Cardinal error measures indicating how well the peer grading methods (& TAs) predict the Instructor/Meta grades.

Method	Posters		Reports	
	Runtime	Runtime (+G)	Runtime	Runtime (+G)
NCS	0.32 ±0.03	7.0 ±0.55	0.20 ±0.03	4.6 ±0.25
MAL	0.01 ±0.00	6.1 ±0.11	0.01 ±0.00	2.5 ±0.03
MAL_{BC}	0.01 ±0.00	5.1 ±0.08	0.01 ±0.00	2.5 ±0.03
MALS	151.4 ±12.39	418.7 ±9.10	2.0 ±0.13	4.2 ±0.16
BT	0.46 ±0.06	5.6 ±0.38	0.21 ±0.02	2.2 ±0.10
THUR	57.9 ±0.76	490.1 ±7.45	12.2 ±0.86	120.8 ±1.03
PL	0.36 ±0.03	4.2 ±0.08	0.18 ±0.01	2.0 ±0.10

Table 8.5: Average runtime (and std. deviation) of different methods (with and w/o grader reliability estimation) in CPU seconds.

TA performance as well. This only further exemplifies the suitability of these techniques as a viable alternate to conventional grading techniques.

8.4.9 How Efficient are the Peer Grading Methods?

While prediction accuracy is the prime concern of grade inference, computational efficiency needs to be sufficient as well. Table 8.5 show the average runtimes and their standard deviations for the posters and the reports. All methods are tractable and most finish within seconds. The Score-Weighted Mallows model is less efficient for problems where each grader assesses many assign-

ments, since the gradient computations involves computing the normalization constant (which involves summing over all rankings). However, training scales linearly with the number of graders. Another method that requires more time is the Thurstone model. The main bottleneck here is the computation of the gradient as it involves looking up a CDF value of the normal distribution.

8.4.10 Do Students Value Peer Grading?

A final point that we would like to ascertain is that peer grading is not only about grade estimation, but also about generating useful feedback. In particular, the cardinal or ordinal assessments were only a small part of the peer feedback. Peer graders had to write a justification for their assessment and comment on the work more generally.

To assess this aspect of peer grading, a survey was conducted at the end of class as part of the course feedback process. This survey included two questions about the student's peer grading experience in the class; more specifically, about how *helpful* the feedback they received was, and how *valuable* the experience of providing feedback was to them. Both questions were to be answered in free-form text. Of the 161 students that participated in the project, 120 students responded to at least one of the questions, with 119 answering the question about receiving feedback (mean response length in characters: 62.93; stdev: 77.22) and 118 the question about providing feedback (mean: 100.36; stdev: 105.74). Following standard practice from survey analysis, we created five categories for coding these open-ended responses as show in Table 8.6. While the first four categories (roughly) follow a decreasing scale of approval, the last serves as a catch-all (including missing responses).

Question A) Was getting peer feedback helpful?		**Question B**) Was providing peer feedback valuable?	
A_1	Yes, it was helpful.	B_1	Yes it was a valuable experience
A_2	Helpful, but not as much as instructor feedback.	B_2	Yes, it was valuable, but with caveats (e.g. took lot of time).
A_3	Somewhat helpful (e.g. only few comments were helpful).	B_3	Only little value (e.g. was too difficult / lacked the grading skills)
A_4	No/Not really/Did not help much.	B_4	Not valuable/Not really valuable.
A_5	Other/Missing	B_5	Other/Missing

Table 8.6: Response categories for survey questions.

%	A_1	A_2	A_3	A_4	A_5	Total
B_1	**34.58**	2.08	5.83	10.00	1.67	**54.17**
B_2	5.42	0.00	5.83	7.08	1.67	20.00
B_3	0.42	2.92	2.08	2.50	0.42	8.33
B_4	2.92	0.83	5.00	5.42	0.00	14.17
B_5	0.00	0.00	0.42	1.67	1.25	3.33
Total	**43.33**	5.83	19.17	26.67	5.00	

Table 8.7: Results of the student survey, coded as per Table 8.6.

All free-text responses were manually assigned to these categories by four external annotators (who were not involved with the class and had not seen the comments before). For all the 237 student comments (*i.e.,* responses), the annotators were asked to choose the category that was *most appropriate/best describes the comment*. To check inter-annotator agreement we used the Fleiss Kappa measure. κ values of 0.8389 and 0.6493 for the two questions indicate high annotator agreement. The final assignment of response to category was done by majority vote among the four annotators (score of 0.5 each if tied between categories).

Table 8.7 summarizes the results of the survey after coding. Overall, around 68% found it at least somewhat helpful to receive peer feedback, and around 74% found substantial value in providing the peer feedback. Interestingly, of the 26% of the students who expressed that receiving peer feedback was not

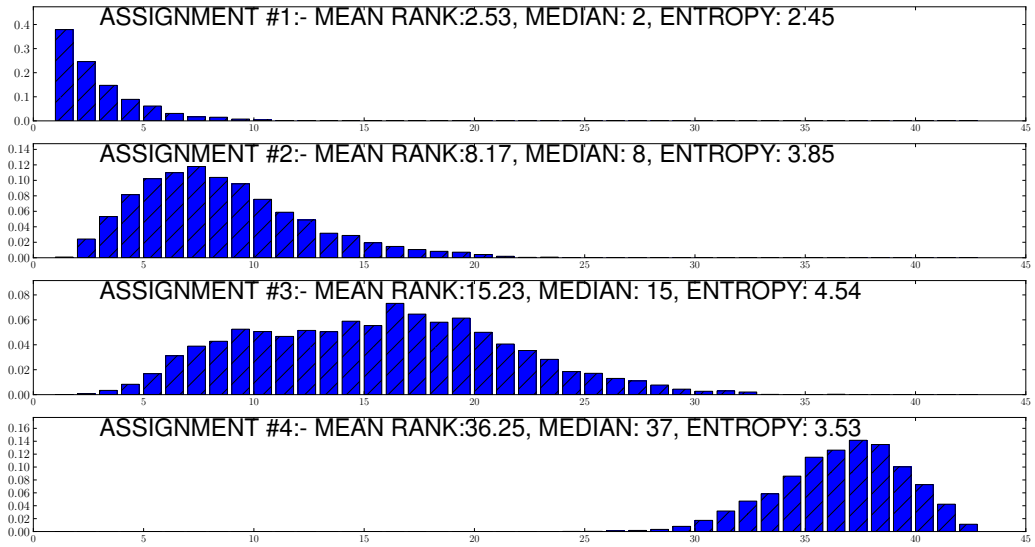

Figure 8.7: An example of detailed grading information for each assignment, including the *posterior marginal distribution* over position in the overall ranking (rank on x-axis, marginal probability on y-axis) along with statistics such as *posterior mean, median* & *marginal entropy*.

(really) helpful to them, 17% still found it valuable to provide peer feedback. Overall, we conclude that the vast majority of students found some value in the peer grading process.

8.5 Bayesian Ordinal Peer Grading

While the ordinal peer grading techniques proposed in Sec 8.3 were shown to estimate accurate rankings of assignments, they are still limited to outputting a single ranking. However, such a ranking does not provide instructors with an estimate of the uncertainty of each assignment's position in the ranking. Suppose instead, we also provide instructors with uncertainty information in the form of posterior distributions, indicating where an assignment lies in the over-

all ranking. Such detailed information of each assignment's performance can be very useful to instructors for determining the final grades. For example, this could be visualized in a manner similar to Figure 8.7. Most importantly, the height of the blue bars shows the probability with which each assignment falls at a specific rank. This information allows instructors to ascertain the algorithm's confidence in the grade (*i.e.,* percentile/position in ranking) of each assignment and discern the uncertainty of the underlying peer grades for each assignment. For instance, in the above example, while there is a high probability that assignment 1 is the best of the four assignments, it is less certain that assignment 2 is better than assignment 3. This is because of the high uncertainty in the position of assignment 3 (as evidenced by its' high entropy of 4.54). If presented with such information, instructors could intervene and improve certainty by soliciting additional reviews for specific assignments, or at least by accounting for the uncertainty when deriving their grades from the ranking.

In this section, we address the problem of uncertainty modeling by employing Bayesian techniques for the ordinal peer grading problem. In particular we extend the Mallows model introduced in Sec 8.3.1 using a Metropolis-Hastings [46] based Markov Chain Monte-Carlo (MCMC) method. The resulting method will allow us to draw samples from the posterior of a Mallows model [116] in an efficient manner. In turn, these samples allow us to empirically estimate the posterior rank distribution of each assignment, allowing us to report confidences and uncertainty information.

8.5.1 Mallows MCMC using Metropolis-Hastings

To help provide more detailed information to instructors, we would like to have access to the posterior distribution of the orderings. In other words, instead of the data likelihood probability we have in Equation 8.5 (ignoring the grader reliabilities for now), we would like to know the posterior distribution of the inferred rankings σ *i.e.,* $P(\sigma|S)$ where S as defined in Eqn 8.1 is the set of all orderings *i.e.,* $S = \{\sigma^{(g)}, \forall g\}$. Using Bayes rule, we get:

$$\begin{aligned} P(\sigma|S) &= \frac{P(S|\sigma)P(\sigma)}{\sum_{\sigma' \in \pi(D)} P(S|\sigma')P(\sigma')} \\ &= \frac{P(S|\sigma)}{\sum_{\sigma' \in \pi(D)} P(S|\sigma')}. \end{aligned} \tag{8.16}$$

where the second line is due to a uniform prior on all orderings (for academic fairness). With this posterior distribution in hand, we can derive the desired marginal rank distributions of each assignment, or we can predict a single ranking that minimizes posterior expected loss.

However, exact computations with this posterior are infeasible given the combinatorial number of possible orderings of all assignments. To help us discern information from the posterior, we will employ Markov Chain Monte Carlo (or MCMC in short) based sampling. MCMC refers to a set of techniques, for sampling from a distribution by constructing a Markov Chain which converges to the desired distribution asymptotically. **Metropolis-Hastings** is a specific MCMC algorithm that is commonly used when the underlying distribution is difficult to sample from (as is the case here with the Mallows model).

To estimate properties of the posterior we will design a Markov Chain whose stationary distribution is the distribution of interest: $P(\sigma|\{\sigma^{(g)}, \forall g\})$. The resulting algorithm, shown in Algorithm 15, is simple and efficient. It begins by pre-

Algorithm 15: Sampling from Mallows Posterior using Metropolis-Hastings

1: **Input:** Grader orderings $\sigma^{(g)}$, Grader reliabilities η_g and MLE ordering $\hat{\sigma}$.

2: Pre-compute $x_{ij} \leftarrow \sum_{g \in G} \eta_g \mathbb{I}[d_i \succ_{\sigma^{(g)}} d_j] - \sum_{g \in G} \eta_g \mathbb{I}[d_j \succ_{\sigma^{(g)}} d_i[$

3: $\sigma_0 \leftarrow \hat{\sigma}$ ▷ Initialize Markov Chain using MLE estimate

4: **for** $t = 1 \ldots T$ **do**

5: Sample σ' from (**MALLOWS**) jumping distribution: $J_{MAL}(\sigma'|\sigma_{t-1})$

6: Compute ratio $r_t = \frac{P(\sigma'|S)}{P(\sigma_{t-1}|S)}$ using Equation 8.17

7: With probability $\min(r_t, 1)$, $\sigma_t \leftarrow \sigma'$ else $\sigma_t \leftarrow \sigma_{t-1}$

8: Add σ_t to samples (if burn-in and thinning conditions met)

computing statistics regarding the (weighted) number of times each assignment d_i is ranked above another assignment d_j. The Markov Chain is then initialized using the MLE estimate ($\hat{\sigma}$) of the ordering (as computed by Algorithm 13). At each timestep, to propose a new sample σ' given the previous sample σ_{t-1}, we sample from a jumping distribution (Line 5). In particular, we use a **Mallows**-based jumping distribution: $\rightarrow \quad J_{MAL}(\sigma'|\sigma) \propto e^{-\delta_K(\sigma', \sigma)}$.
This is a simple distribution to sample from and can be done efficiently in $|D|log|D|$ time. Furthermore as this is a symmetric jumping distribution (*i.e.*, $J_{MAL}(\sigma'|\sigma) = J_{MAL}(\sigma|\sigma')$), the acceptance ratio computation is simplified.

When it comes to computing the (acceptance) ratio r_t (Line 6), we can rely on the pre-computed statistics to do so efficiently. In particular, we can simplify the expression for the ratio to:

$$\begin{aligned} \frac{P(\sigma_a|S)}{P(\sigma_b|S)} &= \prod_{g \in G} e^{\delta_K(\sigma^{(g)}, \sigma_b) - \delta_K(\sigma^{(g)}, \sigma_a)} \\ &= \prod_{i,j} e^{x_{ij}(\mathbb{I}[d_i \succ_{\sigma_a} d_j] - \mathbb{I}[d_i \succ_{\sigma_b} d_j])} \end{aligned} \tag{8.17}$$

This expression is again simple to compute and can be done in time proportional to the number of flipped pairs between σ_a and σ_b, which in the worst case is $O(|D|^2)$. Overall, the algorithm has a **worst-case time complexity** of $O(T|D|^2)$.

The resulting samples produced by the algorithm can be used to *estimate* the posterior distributions including the marginal posterior of the rank of each assignment *i.e.*, $P(r_d|S)$, as well as statistics such as the entropy of the marginal, the posterior mean and median etc. Along with the theoretical guarantees regarding estimates quality that accompany MCMC methods, an added advantage of this algorithm is that we can control the desired estimation accuracy (by selecting the number of samples).

In order to further improve the quality of the resulting estimates, we ensure proper mixing by targeting a moderate acceptance rate and by thinning samples (in our experiments we thin every 10 iterations). Furthermore we draw samples once the chain has started converging *i.e.*, we use a burn-in of 10,000 iterations.

We also derive a Metropolis-Hastings based extension of the Mallows model with grader reliabilities. In addition to sampling the orderings, we also sample the reliabilities using a Gaussian jumping distribution (also symmetric). However the acceptance ratio computation is now more involved and hence less efficient than that for Algorithm 15, but nonetheless can be computed fairly efficiently. We omit the precise equation and computations for brevity.

8.5.2 Evaluating Bayesian Mallows MCMC

Using the dataset described in Sec. 8.4.1, we empirically evaluate the performance of the Bayesian Mallows-based peer grading method, in terms of a) the quality of its predicted rankings; and b) the accuracy of the confidence intervals and uncertainty information. For this evaluation, the Bayesian Mallows MCMC method was run till 5000 samples were drawn from the Markov Chain. These samples were used to estimate the posterior distributions and for obtaining the statistics in the following experiments.

Are the inferred orderings accurate?

A key benefit of the Bayesian approach is that the posterior distribution of the orderings provides uncertainty information. But we can also use the posterior distribution to predict a single ordering of the assignments. How does the accuracy of the orderings predicted by the Bayesian model compare to the accuracy of the orderings estimated via maximum likelihood estimation (MLE)? To address this question, we compare the following techniques:

- **MLE:** Maximum-Likelihood Estimator of the Mallows model(Algorithm Algorithm 13).
- **Mode-MAL:** (One of the) Modes of the posterior of the Mallows distribution. Ties are broken randomly.
- **Mode-MAL+G:** (One of the) Modes of the posterior of the Mallows distribution with grader reliability estimates. Ties are broken randomly.

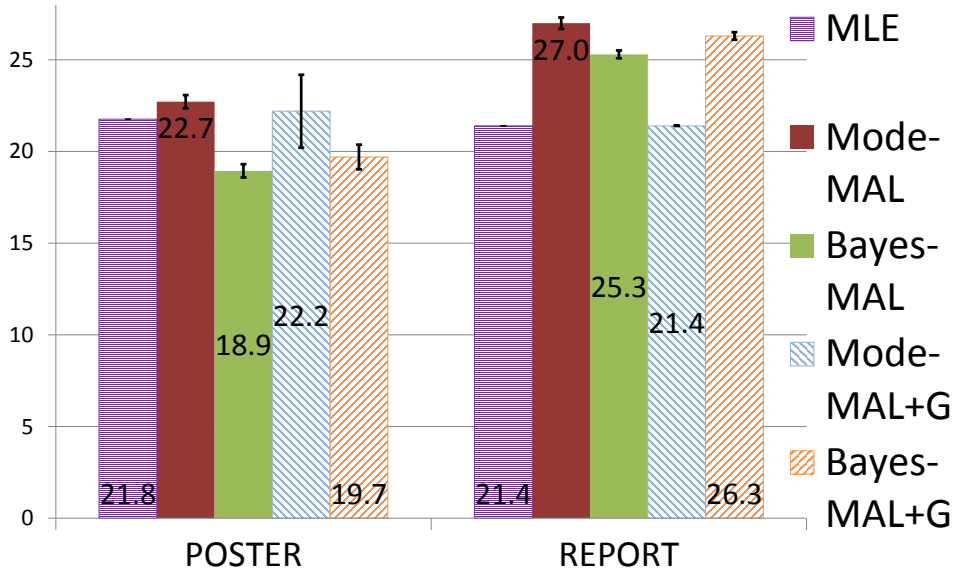

Figure 8.8: $\mathcal{E}_K$ performance of peer grading methods using the instructor grades as the target rankings (lower value is better).

- **Bayes-MAL**: This is the Bayes estimate minimizing posterior expected δ_K over the posterior learned by Alg 15. Formally, the predicted ordering is

$$\hat{\sigma} = \texttt{argmin}_{\sigma} \sum_{\sigma'} \delta_K(\sigma', \sigma) P(\sigma'|D),$$

  where $P(\sigma'|D)$ represents the estimated posterior distribution (as output by the Bayesian MCMC method).

- **Bayes-MAL+G**: The Bayes estimate minimizing posterior expected δ_K over the posterior of the Mallows model with grader reliability estimates.

While computing the Bayes-MAL and Bayes-MAL+G predictions is an NP-hard problem, as it requires computing the Kemeny-optimal aggregate [57], we can efficiently compute a good approximation to this minimization problem using the Borda-Count technique, which is known to be a 5-approximation [51]. In this case, the Borda Count technique also carries a nice semantic meaning as it amounts to simply ordering the assignments by their posterior mean ranks.

As before, we measure performance in terms of $\mathcal{E}_K$ (which can range from 0 to 100 with 50 being random performance). The results are shown in Figure 8.8. On both datasets, the performance of the proposed Bayesian methods are not substantially different from that of the MLE. There appears to be no clear trend that one method is superior to the others, and the differences are probably due to fact that the instructor grades used as a gold standard are themselves subject to uncertainty. One issue to note is that the "Mode" techniques tend to have larger variance, as performance can vary with the mode that was selected (as the distribution tends to be multi-modal).

Lastly, we also note that the performance does not vary much with adding grader reliability estimation. This agrees with the observations made in earlier experiments (for both ordinal and cardinal grading techniques). The most likely reason for observing this behavior is the explicit incentive in terms of grade credit that students were given for doing a thorough job with the peer reviews. Hence the number of truly substandard reviews in the data may be low.

How good are the estimated confidence intervals?

While the previous experiment indicated that the overall quality of the orderings tends to be quite good (with regards to instructor grades), it does not tell us how accurately the Bayesian approach models the uncertainty of the predicted ranks. To address this question, we now evaluate how good the Bayesian confidence intervals (*i.e.*, credible intervals) of the inferred posterior marginal distributions (over position in the overall ranking) for individual assignments are. To evaluate these uncertainty estimates, we again utilize the instructor grades [5].

[5]Since these also have ties, we treat ties as indifference *i.e.,* a uniform probability distribution over all *valid* rank positions.

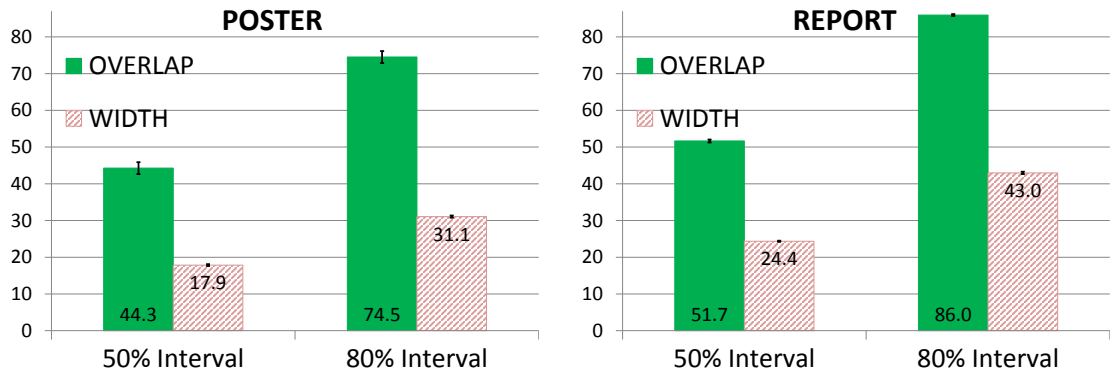

Figure 8.9: Average Overlap (solid green bars) of the 50% and 80% Bayesian credible intervals with the instructor rank distribution, for the intervals produced by the Mallows MCMC method. The red striped bars denote the average size (width) of the interval (as a percentage) of the overall ranking.

In particular we evaluate the quality of the 50% and 80% credible intervals.

For each assignment, we first compute the (posterior) marginal distribution over the ranking positions as shown in Figure 8.7. We then compute the overlap of the credible intervals of these marginals with the instructor ranking distribution. Thus, an assignment whose credible interval contains (all) the instructor-provided ranks has a 100% overlap, whereas an interval with no overlap scores a 0%. We report this overlap averaged over all assignments. Additionally, we report the size of these intervals (as a percentage of the overall ranking length).

The results are shown in Figure 8.9. We find that the intervals produced by the Bayesian MCMC based Mallows technique are well calibrated. In particular, for both the posters and the reports, the 50% and 80% interval cover roughly that percentage of the instructor grades as desired (as indicated by the overlap values). The observed overlap is far greater than the size of the interval, which indicates predictive performance that is far better than random. These results show that the estimated intervals are meaningful and convey accurate

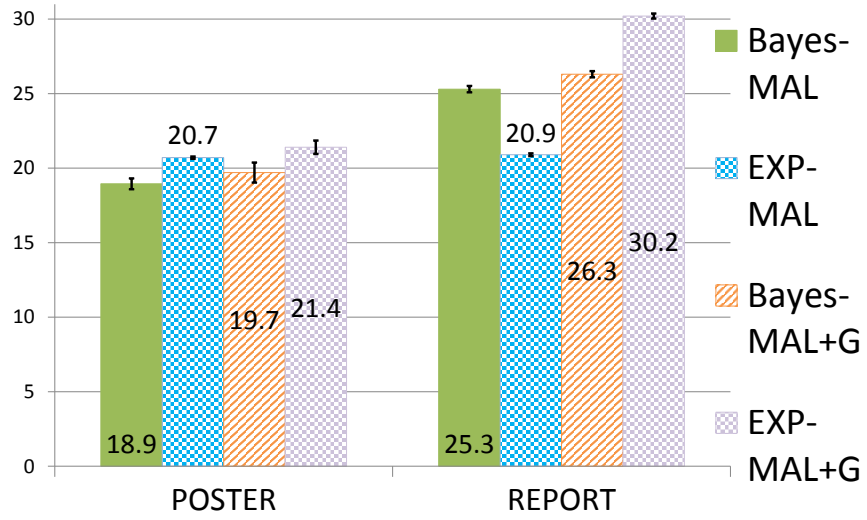

Figure 8.10: $\mathcal{E}_K$ performance of the Bayesian point estimate rankings vs. expected performance of the posterior ranking distribution.

uncertainty information. The results when incorporating grader reliability information are similar and hence left out to avoid redundancy.

How peaked are the posterior distributions?

The above results show that the confidence intervals for the reports have larger width than those for the posters *i.e.,* there is more uncertainty in the marginals of the reports than the posters. This suggests that the posterior distributions are more peaked around the mode for the posters as compared to the reports. To verify this, we computed the expected δ_K under the posterior distribution:

$$\sum_{\sigma} \delta_K(\sigma^*, \sigma) P(\sigma|D)$$

where σ^* refers to the instructor ranking and $P(\sigma|D)$ is the learned posterior. We refer to these values as *EXP-MAL* (without grader reliabilities) and *EXP-MAL+G* (with grader reliability estimation). The results are shown in Figure 8.10.

We find that the difference in performance between the Bayes estimate

(Bayes) and the expected value (EXP) of the full posterior is typically larger for the reports than for the posters. For the posters, it appears that the posterior is so narrow that almost any sample from the posterior is close to the Bayes estimate. For the reports, the posterior is less peaked. This may be explained by the fact that a larger number of reviews were available for the posters.

8.6 Summary

This part of the thesis studied the problem of student evaluation at scale via peer grading using ordinal feedback. The peer grading problem was cast as a rank aggregation problem and approached via different probabilistic modeling techniques. The resulting methods not only produced student grades, but also estimated the reliability of the peer graders. Using data collected from a real course, the performance of ordinal peer grading methods was found to be at least competitive as cardinal methods for grade estimation, even though they require strictly less information from the graders. For grader reliability estimation, the Mallows model outperformed all other methods, and it showed consistently good and robust performance for grade estimation as well. Furthermore, the Mallows model could be extended using a Metropolis-Hastings based MCMC sampler so as to provide instructors with richer information *i.e.*, communicate accurate uncertainty estimates in addition to the predicted ordinal grades. In general, we find that ordinal peer grading is robust and scalable, offering a grading accuracy that is comparable to TA grading in our course.

The methods developed in this chapter have been made *publicly available as software at* `peergrading.org`, where we also provide a **web service for peer grade estimation**. This in turn has led to the successful usage of these methods

in other large classes (300+ students); and in conference reviewing, as evidenced in the 2015 KDD conference [86], where it received positive feedback from senior program committee members.

Part V

Conclusions

CHAPTER 9

IMPLICATIONS OF WORKING WITH HUMAN DECISION DATA

Intelligent technologies are becoming increasingly critical to our everyday functioning. A key to the optimal functioning of these technologies, is maintaining a symbiotic relationship with their users. On one hand, users greatly benefit from the services provided to them by these technologies. Simultaneously, these systems benefit greatly by learning from the interactions of the users with these technologies.

Unlike conventional learning problems, this interaction data does not directly provide the system with expert labels to learn from. Rather, these interactions are the result of (potentially complex) decisions made by these human users using the rich world knowledge they each have in their minds. Thus, while still being incredibly rich sources for learning, it is imperative for these systems to account for the human decision making process and underlying factors – such as motivation, context, expertise – that affect this decision.

Towards this goal of principled learning from such human decision data, this thesis introduced a new approach to designing learning techniques for these systems. In particular, it highlighted the significance of jointly designing the three fundamental keys to interactive learning – the algorithm, the user model and the feedback interventions. Using this learning triple philosophy, this dissertation introduced different learning systems that were not only theoretically sound, but also effective in practice as demonstrated in real user studies.

This triple based interactive learning approach was demonstrated on learning problems from different domains, including search and ranking (Part II);

recommendation and complex task retrieval (Part III); and education (Part IV). Despite the diversity of the problem domains, certain common themes were identified in the design of the three key interactive learning components.

In particular, the learning algorithms that worked well with this interaction data, were those that treated the human decisions as preferences/choices over different alternatives rather than those that treated them as absolute judgments. Additionally, learning algorithms that were robust to the noise and biases in the underlying data, were found to work the best. Similarly, with regards to accounting for the user behavior while interacting with these systems, tools from behavioral sociology and micro-economics such as rational choice theory provided valuable insights into reasoning about the user decision making process. The notion of using feedback interventions for interactive learning was introduced in this thesis, and was found to be particularly effective for these learning problems given the additional control interactive systems have over what users are presented with. These interventions were found to improve learning, as users and the system shared exploration towards finding a better solution.

The insights developed in this dissertation have also opened up new research possibilities and avenues. For instance, while Part II illustrated the effectiveness of introducing appropriate feedback interventions using a live search engine as an example, the notion of feedback interventions is still very new and understudied. Determining what the right set of interventions are for a new learning problem, and rigorously understanding the science and theory behind them poses intriguing research questions. Exploiting interventions in other learning paradigms could also be beneficial for improving learning.

Part III of this dissertation made significant inroads into the increasingly,

critical problem of assisting users in their complex tasks. However, there is still significant work that needs to be done in this field before we can provide users with effective tools to perform highly non-trivial, multi-step tasks such as planning a business trip. Utilizing proactive technologies to assist users, as advocated and demonstrated in Chapters 4 and 7, is a particularly promising option to help users tackle these tasks and improve overall user experience. While proactively predicting the immediate future needs of a user is far more challenging than the problems current information systems address (due to the paucity of context), the insights gained in this dissertation can help build robust systems to learn to proactively assist using large-scale interaction logs

Intelligent interactive technologies continue to grow in importance as new disruptive applications constantly surface. Part IV studied the application of these learning techniques for one such promising domain – educational technologies. Given the promise of online education platforms, there is ample incentive to explore applying the insights from this thesis towards newer technical challenges in these domains as they arise. For instance, the interactive learning triple philosophy can help guide the building of adaptive personal tutoring assistants, which can adapt a student's curriculum based on how quickly the student grasps specific concepts, and adjust depth based on student interest, and customize tests to minimize testing overhead. Furthermore, other disruptive technologies such as smart homes and self-driving cars, are well suited for the interactive learning techniques studied in this dissertation, as the user-system interactions are rich in valuable signals and knowledge that can be used to push the frontier in these technologies.

APPENDIX A

FURTHER DETAILS AND PROOFS

A.1 Proofs

A.1.1 Proof of Theorem 3

First, we bound $\mathbf{E}[\|\mathbf{w}_{T+1}\|^2]$:

$$\begin{aligned}
\mathbf{E}[\mathbf{w}_{T+1}^\top \mathbf{w}_{T+1}] &= \mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_T + 2\mathbf{w}_T^\top \phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) \\
&\quad - 2\mathbf{w}_T^\top \phi(\mathbf{x}_T, \mathbf{y}_T) + \|\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)\|^2] \\
&\leq \mathbf{w}_1^\top \mathbf{w}_1 + 2\sum_{t=1}^{T} \mathbf{E}[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)] + 4R^2 T \\
&\leq (4R^2 + 2\Delta)T
\end{aligned}$$

The first line utilizes the update rule from algorithm 2. The second line follows from $\|\phi(\mathbf{x}, \mathbf{y})\| \leq R$ and repeating the inequality for $t = T - 1, \cdots, 1$. The last inequality uses the premise on affirmativeness.

Using the update rule again, we get:

$$\begin{aligned}
\mathbf{E}[\mathbf{w}_{T+1}^\top \mathbf{w}_*] &= \mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_* + (\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T))^\top \mathbf{w}_*] \\
&= \sum_{t=1}^{T} \mathbf{E}[(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t))] \\
&\geq \alpha \sum_{t=1}^{T} (U(\mathbf{x}_t, \mathbf{y}_t^*) - \mathbf{E}[U(\mathbf{x}_t, \mathbf{y}_t)]) - \sum_{t=1}^{T} \xi_t
\end{aligned}$$

where the last line uses Eq. (3.4). Using the Cauchy-Schwarz inequality and concavity of $\sqrt{x}$, we get $\mathbf{E}[\mathbf{w}_{T+1}^\top \mathbf{w}_*] \leq \|\mathbf{w}_*\| \mathbf{E}[\|\mathbf{w}_{T+1}\|] \leq \|\mathbf{w}_*\| \sqrt{\mathbf{E}[\|\mathbf{w}_{T+1}\|^2]}$ from which the claimed result follows.

A.1.2 Proof of Corollary 4

Note that:

$$\hat{\mathbf{y}}_t = \texttt{argmax}_{\mathbf{y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$$

Therefore:

$$\forall t, \bar{\mathbf{y}}_t : \mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) \leq \mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)$$

Hence:

$$\begin{aligned} &\forall t : \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)\Big] - \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \\ &\leq \mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \end{aligned} \tag{A.1}$$

Given the condition of the corollary, and the above Equation A.1, we get that:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)\Big] - \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \leq \Omega$$

which using Theorem 3 gives us the corresponding regret bound.

A.1.3 Proof of Theorem 5

This proof is very similar to the one given below (Appendix A.1.6) for Thm 9, though it solves a different problem. In particular since:

$$\forall t : \mathbf{E}\Big[\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t)\Big] \geq (1 - \beta) \mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)$$

we have that:

$$\mathbf{E}[\mathbf{w}_t^\top (\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t))] \leq \beta \mathbf{w}_t^\top \phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)$$

From here on, the proof from [141] can be used, to prove the corresponding regret bound. Thus in other words, the perturbation can be thought of as a way to produce an $(1 - \beta)$-*approximate* solution to the argmax problem.

A.1.4 Proof of Proposition 6

Consider the case when documents in positions i and $i+1$ (call them d_i and d_{i+1}) are swapped[1]:

$$\begin{aligned}&\mathbf{w}_t^\top(\gamma_i - \gamma_{i+1})(\phi(\mathbf{x}_t, d_i) - \phi(\mathbf{x}_t, d_{i+1}))\\ &\leq \Big(1 - \frac{\gamma_{i+1}}{\gamma_i}\Big)\mathbf{w}_t^\top(\gamma_i\phi(\mathbf{x}_t, d_i) + \gamma_{i+1}\phi(\mathbf{x}_t, d_{i+1}))\end{aligned}$$

Note that this factor $1 - \frac{\gamma_{i+1}}{\gamma_i}$ is largest for $i = 1$. Thus we can state for every swapped pair:

$$\begin{aligned}&\mathbf{w}_t^\top(\gamma_i - \gamma_{i+1})(\phi(\mathbf{x}_t, d_i) - \phi(\mathbf{x}_t, d_{i+1}))\\ &\leq \Big(1 - \frac{\gamma_2}{\gamma_1}\Big)\mathbf{w}_t^\top(\gamma_i\phi(\mathbf{x}_t, d_i) + \gamma_{i+1}\phi(\mathbf{x}_t, d_{i+1}))\end{aligned}$$

Summing this over all swapped pairs, and using the fact that each pair has some probability p to be swapped:

$$\begin{aligned}&\mathbf{w}_t^\top(\phi(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \mathbf{E}[\phi(\mathbf{x}_t, \mathbf{y}_t)])\\ &\leq p\Big(1 - \frac{\gamma_2}{\gamma_1}\Big)\mathbf{w}_t^\top\phi(\mathbf{x}_t, \hat{\mathbf{y}}_t)\end{aligned}$$

A.1.5 Proof of Proposition 7

We prove a more general proposition here:

Proposition 19 *For $\Delta \geq 0$, dynamically setting the swap prob. of 3PR to be*

$$p_t \leq max\Big(0, min\Big(1, c(\Delta \cdot t - R_t)\Big)\Big), \tag{A.2}$$

[1]This holds assuming the inner products with documents are non-negative. Thus algorithmically this can be implemented by only ranking documents with non-negative scores.

for some positive constant c, has regret

$$\leq \frac{1}{\alpha T}\sum_{t=1}^{T}\xi_t + \frac{\|\mathbf{w}_*\|}{\alpha\sqrt{T}}\sqrt{4R^2 + 2\Delta + (\gamma_1-\gamma_2)R\sqrt{\frac{4R^2+2\Delta}{T}}}.$$

Proof We prove this by using Theorem 3. In particular, we show:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t^\top(\phi(\mathbf{x}_t,\bar{\mathbf{y}}_t)-\phi(\mathbf{x}_t,\mathbf{y}_t)) < \Delta + \Gamma\sqrt{\frac{4R^2+2\Delta}{T}} \tag{A.3}$$

where $\Gamma = (\gamma_1 - \gamma_2)R$. We will show this holds by induction on T. Note that this condition trivially holds for $T = 0$ (base case). Now assume it holds for $T = k-1$. We will show it is true for $T = k$. Consider the cumulative affirmativeness $R_k = \sum_{i=1}^{k-1}\mathbf{w}_i^\top\phi(\mathbf{x}_i,\bar{\mathbf{y}}_i) - \mathbf{w}_i^\top\phi(\mathbf{x}_i,\mathbf{y}_i)$. There are 2 cases to consider:

- $R_k \geq k\Delta$: If this is the case $p_k = 0$ *i.e.,* no perturbation is performed for iteration k and hence $\mathbf{y}_k = \hat{\mathbf{y}}_k = \texttt{argmax}_{\mathbf{y}}\mathbf{w}_k^\top\phi(\mathbf{x}_k,\mathbf{y})$. Therefore $\mathbf{w}_k^\top(\phi(\mathbf{x}_k,\bar{\mathbf{y}}_k) - \phi(\mathbf{x}_k,\mathbf{y}_k)) \leq 0$; thus $R_{k+1} \leq R_k$ and hence the induction hypothesis is satisfied.
- $R_k < k\Delta$: We have $\|\mathbf{w}_k\| \leq \sqrt{k(4R^2+2\Delta)}$ as shown in the proof of Thm 3. As per the perturbation, for all $\mathbf{y}_k$ we have $\|\phi(\mathbf{x}_k,\hat{\mathbf{y}}_k) - \phi(\mathbf{x}_k,\mathbf{y}_k)\| \leq \Gamma^2$. Next by Cauchy-Schwarz we get $\mathbf{w}_k^\top(\phi(\mathbf{x}_k,\hat{\mathbf{y}}_k) - \phi(\mathbf{x}_k,\mathbf{y}_k)) \leq \|\mathbf{w}_k\|\Gamma$. Thus $R_{k+1} \leq R_k + \Gamma\sqrt{k(4R^2+2\Delta)}$; hence satisfying the induction hypothesis.

Thus the induction holds for $T = k$. Since equation (A.3) holds for all $\mathbf{y}_t, \bar{\mathbf{y}}_t$, this condition is also satisfied under expectation (over $\mathbf{y}_t, \bar{\mathbf{y}}_t$). Hence the condition for Theorem 3 is satisfied, thus giving us the bound. Note that the second term on the RHS of Eq. (A.3) asymptotically disappears. ■

[2]This assumes that the document feature vectors are component-wise non-negative. If this is not true, then the bound still holds but with $\Gamma = 2R$

A.1.6 Proof of Theorem 9

Proof Consider the ℓ_2 norm of $\mathbf{w}_T$:

$$\begin{aligned}
\|\mathbf{w}_T\|^2 &= \|\mathbf{w}_{T-1}\|^2 + 2\mathbf{w}_{T-1}^\top(\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})) \\
&\quad + \|\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})\|^2 \\
&\leq \|\mathbf{w}_{T-1}\|^2 + 2\beta\, \mathbf{w}_{T-1}^\top \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1}) + 4R^2 \\
&\leq \|\mathbf{w}_{T-1}\|^2 + 2\beta\|\mathbf{w}_{T-1}\|R + 4R^2 \qquad \text{(A.4)}
\end{aligned}$$

The first line comes from the update rule in Algorithm 6. The second line is from the fact: $\mathbf{w}_{T-1}^\top \phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) \leq (\beta + 1)\mathbf{w}_{T-1}^\top \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})$ since the greedy algorithm produces an $\frac{1}{\beta+1}$ approximation and that $\|\phi(\cdot, \cdot)\| \leq R$. The third line comes by using the Cauchy-Schwarz inequality.

Let us inductively assume that $\|\mathbf{w}_t\| \leq c_1 R(t + c_2)$ for $t = \{0, ...T - 1\}$ where the values $c_1, c_2 \geq 0$ will be determined later. The base case is trivially shown as $\|\mathbf{w}_0\| = 0$. Thus to complete the induction step, we have:

$$\begin{aligned}
\|\mathbf{w}_T\|^2 &\leq \|\mathbf{w}_{T-1}\|^2 + 2\beta\|\mathbf{w}_{T-1}\|R + 4R^2 \\
&\leq \|\mathbf{w}_{T-2}\|^2 + 2\beta R(\|\mathbf{w}_{T-1}\| + \|\mathbf{w}_{T-2}\|) + 8R^2 \\
&\leq \|\mathbf{w}_0\|^2 + 2\beta R \sum_{t=0}^{T-1} \|\mathbf{w}_t\| + 4R^2 T \\
&\leq \beta R^2 c_1 (T^2 - T) + 2\beta R^2 T c_1 c_2 + 4R^2 T \\
&\leq R^2\Big(\beta c_1 T^2 + T(-\beta c_1 + 2\beta c_1 c_2 + 4)\Big)
\end{aligned}$$

We now choose c_1 and c_2 such that the induction step holds. This is done by ensuring that the coefficients of T^2 and T in the above expression are smaller than the corresponding terms in $c_1^2 T^2 + 2c_1^2 c_2 T + c_1^2 c_2^2$. First, set $c_1 = \beta + \epsilon$, which will ensure the inequality for T^2. Next, we can ensure $-\beta c_1 + 2\beta c_1 c_2 + 4 \leq 2c_1^2 c_2$, by

setting $c_2 = \frac{4-\beta(\beta+\epsilon)}{2\epsilon(\epsilon+\beta)}$. We therefore have $\|\mathbf{w}_T\| \leq (\epsilon+\beta)TR + \frac{(4-\beta^2)R}{2\epsilon} - \frac{\beta R}{2}$. Minimizing the above bound over ϵ, we get $\epsilon = \sqrt{\frac{4-\beta^2}{2T}}$. Substituting this in the upper bound for $\|\mathbf{w}_T\|$, we get $\|\mathbf{w}_T\| \leq (\beta T + \sqrt{4-\beta^2}\sqrt{2T})R$.

Thus using the update rule of Algorithm 6, we have,

$$\begin{aligned}\mathbf{w}_T^\top \mathbf{w} &= \mathbf{w}_{T-1}^\top \mathbf{w} + U(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - U(\mathbf{x}_{T-1}, \mathbf{y}_{T-1}) \\ &= \sum_{t=0}^{T-1} U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t).\end{aligned}$$

We now use the fact that $\mathbf{w}_T^\top \mathbf{w} \leq \|\mathbf{w}\|\|\mathbf{w}_T\|$ (Cauchy-Schwarz inequality) which implies,

$$\sum_{t=0}^{T-1} U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \leq (\beta T + \sqrt{4-\beta^2}\sqrt{2T})R\|\mathbf{w}\|.$$

The above inequality, along with the condition of *α-informative feedback gives:*

$$\alpha REG_T - \frac{1}{T}\sum_{t=0}^{T-1} \xi_t \leq \left(\beta + \sqrt{4-\beta^2}\sqrt{\frac{2}{T}}\right) R\|\mathbf{w}\|$$

from which the claimed result follows. ■

A.1.7 Proof of Corollary 10

Start by observing that, for any t,

$$\|\mathbf{w}_t\|^2 \leq \|\bar{\mathbf{w}}_t\|^2 \quad \text{and} \quad \mathbf{w}^\top \mathbf{w}_t \geq \mathbf{w}^\top \bar{\mathbf{w}}_t \tag{A.5}$$

The first inequality holds because the product of any clipped value with itself is positive. Since all the components of $\mathbf{w}$ are positive and since only negative values in $\bar{\mathbf{w}}_T$ are set to zero in the clipping step, the second inequality holds. With these two steps, the remaining steps in the proof of Theorem 10 follow and we get the corollary.

A.1.8 Proof of Theorem 11

Proof We look at how the KL divergence between $\mathbf{w}$ and $\mathbf{w}_t$ evolves,

$$\begin{aligned} KL(\mathbf{w}||\mathbf{w}_t) - KL(\mathbf{w}||\mathbf{w}_{t+1}) &= \sum_{i=1}^{m} \mathbf{w}^i \log(\mathbf{w}^i_{t+1}/\mathbf{w}^i_t) \\ &= \sum_{i=1}^{m} \mathbf{w}^i(\theta(\phi^i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi^i(\mathbf{x}_t, \mathbf{y}_t))) - \log(Z_t) \\ &= \theta \mathbf{w}^\top(\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) - \log(Z_t). \end{aligned} \tag{A.6}$$

On the second line, we pulled out $\log(Z_t)$ from the sum since $\sum_{i=1}^{m} \mathbf{w}^i = 1$. Now, consider the last term in the above equation. Denoting $\phi^i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi^i(\mathbf{x}_t, \mathbf{y}_t)$ by $\Delta^i\phi_t$ for brevity, we have, by definition,

$$\begin{aligned} \log(Z_t) &= \log\left(\sum_{i=1}^{m} \mathbf{w}^i_t \exp(\theta \Delta^i \phi_t)\right) \\ &\leq \log\left(\sum_{i=1}^{m} \mathbf{w}^i_t (1 + \theta\Delta^i\phi_t + \theta^2 {\Delta^i\phi_t}^2)\right) \\ &\leq \log\left(1 + \theta \mathbf{w}_t^\top \Delta\phi_t + \theta^2 S^2\right) \\ &\leq \theta \mathbf{w}_t^\top \Delta\phi_t + \theta^2 S^2. \end{aligned} \tag{A.7}$$

On the second line we used the fact that $\exp(x) \leq 1 + x + x^2$ for $x \leq 1$. The rate θ ensures that $\theta(\Delta^i\phi) \leq 1$. On the last line, we used the fact that $\log(1 + x) \leq x$. Combing (A.6) and (A.7), we get,

$$(\mathbf{w} - \mathbf{w}_t)^\top \Delta\phi_t \leq \frac{KL(\mathbf{w}||\mathbf{w}_t) - KL(\mathbf{w}||\mathbf{w}_{t+1})}{\theta} + \theta S^2.$$

Adding the above inequalities, we get:

$$\begin{aligned} &\sum_{t=0}^{T-1} (\mathbf{w} - \mathbf{w}_t)^\top(\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) \\ \leq &\sum_{t=0}^{T-1} \frac{KL(\mathbf{w}||\mathbf{w}_t) - KL(\mathbf{w}||\mathbf{w}_{t+1})}{\theta} + \sum_{t=0}^{T-1} \theta S^2. \\ \leq &\frac{KL(\mathbf{w}||\mathbf{w}_0)}{\theta} + \theta S^2 T. \end{aligned} \tag{A.8}$$

Rearranging the above inequality, and substituting the value of θ from Algorithm 8, we get:

$$\begin{aligned}
&\sum_{t=0}^{T-1}(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)) \\
\leq &\sum_{t=0}^{T-1} \mathbf{w}_t^\top(\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) + 2\log(m)S\sqrt{T} + \frac{S\sqrt{T}}{2} \\
\leq &\sum_{t=0}^{T-1} \beta \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t) + 2\log(m)S\sqrt{T} + \frac{S\sqrt{T}}{2} \\
\leq &\beta S T + 2\log(m)S\sqrt{T} + \frac{S\sqrt{T}}{2}. \qquad \text{(A.9)}
\end{aligned}$$

In the above, we also used the fact that $KL(\mathbf{w}||\mathbf{w}_0) \leq \log(m)$ since $\mathbf{w}_0$ is initialized uniformly. On line three, we used the fact that the greedy algorithm finds a $\frac{1}{1+\beta}$ approximation. Moreover, from a generalized version of Cauchy-Schwarz inequality, we obtained

$$\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t) \leq \|\mathbf{w}_t\|_{\ell_1} \|\phi(\mathbf{x}_t, \mathbf{y}_t)\|_{\ell_\infty} \leq S.$$

The above inequality along with α-informative feedback gives the claimed result. ■

A.1.9 Proof of Lemma 13

Proof For $\gamma_1 = \ldots = \gamma_k$ this is a straightforward reduction to monotone submodular maximization with a cardinality constraint for which the greedy algorithm is $(1 - \frac{1}{e})$-approximate [121]. For the more general case we reduce it to submodular maximization over a partition matroid. Suppose we have documents $\{d_1, \ldots, d_N\}$ and want to find a ranking of length k. Let the new ground set A contain k copies

$d_{i,j}$: $j \in \{1, k\}$ of each document d_i, one for each position. The matroid only permits sets containing at most one document per position. Define set function H over A: For set $B(\subseteq A)$, let $C = \{\ldots d_{i_j,j} \ldots\}$ be the set obtained by removing all duplicates from B (*i.e.,* keep only the highest ranked occurrence of a document). Define $H(B) = F(\ldots, \gamma_j \phi(\mathbf{x}, d_{i_j}), \ldots)$. The lemma follows from observing that Algorithm 5 is equivalent to the greedy algorithm for maximizing H over A under a matroid constraint, which is known to provide a $\frac{1}{2}$-approximate solution [121]. ■

A.1.10 Proof of Theorem 15

Proof From Lemma 13, we get that:

$$\mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t) \geq \beta_{gr} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t)$$

$$\mathbf{w}_t^\top (\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) \leq (1 - \beta_{gr}) \mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) \leq \beta R \|\mathbf{w}_t\| \tag{A.10}$$

Next, we bound the ℓ_2 norm of $\mathbf{w}_T$:

$$\begin{aligned} \|\mathbf{w}_T\|^2 &= \|\mathbf{w}_{T-1}\|^2 + 2\mathbf{w}_{T-1}^\top (\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})) \\ &\quad + \|\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})\|^2 \\ &\leq \|\mathbf{w}_{T-1}\|^2 + 2\beta \|\mathbf{w}_{T-1}\| R + 4R^2 \\ &\leq (\beta T + \sqrt{4 - \beta^2}\sqrt{2T})^2 R^2 \end{aligned} \tag{A.11}$$

Eq. (A.10) is used for the second inequality. The last line is obtained using the inductive argument made in the proof in Appendix A.1.6. Similarly we bound $\mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_*]$ using Cauchy-Schwartz and concavity:

$$\|\mathbf{w}_*\| \mathbf{E}[\|\mathbf{w}_{T+1}\|] \geq \mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_*] = \sum_{t=0}^{T-1} \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] \tag{A.12}$$

Now we use the α_i, δ_i-informativeness condition:

$$\begin{aligned}\mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] &\geq \alpha_i \left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) + \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - \bar{\xi}_t \\ &\geq \frac{\eta}{p_i}\left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) + \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - \bar{\xi}_t \qquad \text{(A.13)}\end{aligned}$$

Next we bound the expected difference in the social utility between $\bar{\mathbf{y}}_t$ and $\mathbf{y}_t$ IF a user of type i provided feedback at iteration t:

$$\begin{aligned}\Delta_i &= \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] \geq -\Gamma_F \sum_{j \neq i} p_j U_j(\mathbf{x}_t, \mathbf{y}_t) + p_i \mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] \\ &= -\Gamma_F(U(\mathbf{x}_t, \mathbf{y}_t) - p_i U_i(\mathbf{x}_t, \mathbf{y}_t)) + p_i \mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] \\ &\geq -\Gamma_F U(\mathbf{x}_t, \mathbf{y}_t) + p_i \Gamma_F U_i(\mathbf{x}_t, \mathbf{y}_t) + \eta\left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) + p_i \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - p_i \bar{\xi}_t \\ &\geq \eta\left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) + \Gamma_F\left(U_i(\mathbf{x}_t, \mathbf{y}_t) - U(\mathbf{x}_t, \mathbf{y}_t)\right) - p_i \bar{\xi}_t \qquad \text{(A.14)}\end{aligned}$$

The first line is obtained by using Lemma 14 and definition of Γ_F (Eq. 6.10). The second line uses the definition of the social utility (Eq. 6.1). The third line uses Eq. A.13. The fourth step uses the condition on δ_i and rearranging of terms. Note that the expectations in the above lines are w.r.t. the user feedback (and the feedback construction process).

We next consider the expected value of Δ_i (over the user distribution):

$$\begin{aligned}\mathbf{E}_i[\Delta_i] = \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] &\geq \eta\left(\mathbf{E}_i[U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i})] - U(\mathbf{x}_t, \mathbf{y}_t)\right) - \mathbf{E}_i[p_i \bar{\xi}_t] \\ &\geq \eta\left(U(\mathbf{x}_t, \mathbf{y}_t^{*}) - U(\mathbf{x}_t, \mathbf{y}_t)\right) - \mathbf{E}_i[p_i \bar{\xi}_t] \qquad \text{(A.15)}\end{aligned}$$

where the second line uses the fact that $\mathbf{E}_i[U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i})] \geq U(\mathbf{x}_t, \mathbf{y}_t^{*})$. We can put together Eqns. A.11, A.12 and A.15 to give us the required bound. ■

A.2 Additional Details of arXiv User Study

The ranking function in the ArXiv search engine used 1000 features which can be categorized into the following three groups.

- Features the corresponded to rank as per query similarity with different components of the document (authors, abstract, article *etc.*.). We used different similarity measures. For each of these document-components and similarity measures, we had multiple features of the form $\texttt{r}\text{ank} \leq a$, where a was a value we varied to create multiple features (we used $2, 5, 10, 15, 25, 30, 50, 100, 200$).
- Second-order features the represented pairwise combinations of rank (for the default similarity measure) for 2 different document-components.
- Query-independent features representing the document age and the document category (e.g. AI, NLP, ML, Statistics *etc.*.).

Our baseline, was a hand-coded solution using 35 features considered the most important by us.

BIBLIOGRAPHY

[1] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 11–22, 1992.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1–, 2004.

[3] E. Agichtein, R.W. White, S.T. Dumais, and P.N. Bennett. Search, Interrupted: Understanding and Predicting Search Task Continuation. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 315–324, 2012.

[4] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 5–14, 2009.

[5] Shipra Agrawal and Navin Goyal. Analysis of Thompson Sampling for the multi-armed bandit problem. *CoRR*, abs/1111.1797, 2011.

[6] Nir Ailon. Aggregation of partial rankings, P-ratings and top-m lists. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 415–424, 2007.

[7] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, November 2008.

[8] Russ Allen. `http://housedivided.dickinson.edu/sites/blogdivided/2013/07/22/the-absent-professor-grading-assignments-for-moocs`, July 2013.

[9] Kenneth J. Arrow. *Social Choice and Individual Values*. Yale University Press, 2nd edition, September 1970.

[10] Javed A. Aslam and Mark Montague. Models for Metasearch. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 276–284, 2001.

[11] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[12] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[13] Anne Aula, Päivi Majaranta, and Kari-Jouko Räihä. Eye-tracking reveals the personal styles for search result evaluation. In *Human-Computer Interaction-INTERACT 2005*, pages 1058–1061. Springer, 2005.

[14] Leif Azzopardi. The economics in interactive information retrieval. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 15–24, 2011.

[15] Leif Azzopardi. Modelling interaction with economic models of search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12, 2014.

[16] Yoram Bachrach, Thore Graepel, Tom Minka, and John Guiver. How to grade a test without knowing the answers - a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *International Conference on Machine Learning (ICML)*, 2012.

[17] Ricardo Baeza-yates, Carlos Hurtado, Marcelo Mendoza, and Georges Dupret. Modeling user search behavior. In *In LA-WEB 05: Proceedings of the Third Latin American Web Congress*, page 242. IEEE Computer Society, 2005.

[18] Antonio Bahamonde, Gustavo F Bayón, Jorge Díez, José Ramón Quevedo, Oscar Luaces, Juan José Del Coz, Jaime Alonso, and Félix Goyache. Feature subset selection for learning preferences: a case study. In *International Conference on Machine Learning (ICML)*, page 7, 2004.

[19] Peter Bailey et al. User task understanding: a web search engine perspective. http://research.microsoft.com/apps/pubs/default.aspx?id=180594, 2012.

[20] William Barnett. The modern theory of consumer behavior: Ordinal or cardinal? *The Quarterly Journal of Austrian Economics*, 6(1):41–65, 2003.

[21] Gábor Bartók, Dávid Pál, and Csaba Szepesvári. Toward a classification

of finite partial-monitoring games. In *Algorithmic learning theory*, pages 224–238, 2010.

[22] Gábor Bartók, Navid Zolghadr, and Csaba Szepesvári. An adaptive algorithm for finite stochastic partial monitoring. In *International Conference on Machine Learning (ICML)*, 2012.

[23] Maryam Bashir, Jesse Anderton, Jie Wu, Peter B. Golbus, Virgil Pavlu, and Javed A. Aslam. A document rating system for preference judgements. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 909–912, 2013.

[24] N. J. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining evidence for information retrieval. In *In D. Harman (Ed.), TREC-2, Proceedings of the Second Text Retrieval Conference*, pages 35–44, 1994.

[25] Paul N. Bennett, K. Svore, and S. Dumais. Classification-Enhanced Ranking. In *World Wide Web Conference (WWW)*, pages 111–120, 2010.

[26] Menucha Birenbaum and Kikumi K. Tatsuoka. Open-ended versus multiple-choice response formatsit does make a difference for diagnostic purposes. *Applied Psychological Measurement*, 11(4):385–395, 1987.

[27] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[28] L'hadi Bouzidi and Alain Jaillet. Can online peer assessment be trusted? *Educational Technology & Society*, 12(4):257–268, 2009.

[29] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):pp. 324–345, 1952.

[30] Christina Brandt, Thorsten Joachims, Yisong Yue, and Jacob Bank. Dynamic ranked retrieval. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 247–256, 2011.

[31] Chris Buckley. Implementation of the smart information retrieval system. Technical report, Cornell University, 1985.

[32] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole

Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML)*, pages 89–96, 2005.

[33] Christopher JC Burges, Krysta Marie Svore, Paul N Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. In *Yahoo! Learning to Rank Challenge*, pages 25–35, 2011.

[34] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine Learning (ICML)*, pages 129–136, 2007.

[35] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 335–336, 1998.

[36] Ben Carterette, Paul N. Bennett, David Maxwell Chickering, and Susan T. Dumais. Here or there: Preference judgments for relevance. In *European Conference on Information Retrieval (ECIR)*, pages 16–27, 2008.

[37] Nicolò Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *International Conference on Machine Learning (ICML)*, pages 100–108, 1998.

[38] Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[39] Chi-Cheng Chang, Kuo-Hung Tseng, Pao-Nan Chou, and Yi-Hui Chen. Reliability and validity of web-based portfolio peer assessment: A case study for a senior high school's students taking computer course. *Comput. Educ.*, 57(1):1306–1316, August 2011.

[40] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge*, pages 1–24, 2011.

[41] Oliviier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, 30(1):6:1–6:41, 2012.

[42] D. Chen and D. Xiang. The consistency of multicategory support vector machines. *Adv. Comput. Math*, 24(1-4):155–169, 2006.

[43] Harr Chen and David R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 429–436, 2006.

[44] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning (ICML)*, pages 151–159, 2013.

[45] Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 193–202, 2013.

[46] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, 1995.

[47] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *ICML*, 2005.

[48] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 659–666, 2008.

[49] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 1996.

[50] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8, 2002.

[51] Don Coppersmith, Lisa K. Fleischer, and Atri Rurda. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Trans. Algorithms*, 6(3):55:1–55:13, July 2010.

[52] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 758–759, 2009.

[53] Coursera. `http://blog.coursera.org/post/64907189712/`

a-triple-milestone-107-partners-532-courses-5-2, October 2013.

[54] Kolby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS)*, pages 641–647, 2001.

[55] S. Dasgupta and J. Langford. Tutorial summary: Active learning. In *International Conference on Machine Learning (ICML)*, page 178, 2009.

[56] Jorge Dıez, Gustavo F Bayón, José R Quevedo, Juan José Del Coz, Oscar Luaces, Jaime Alonso, and Antonio Bahamonde. Discovering relevancies in very difficult regression problems: applications to sensory data analysis. In *European conference on artificial intelligence (ECAI)*, pages 993–994, 2004.

[57] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *World Wide Web Conference (WWW)*, pages 613–622, 2001.

[58] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 439–447, 2011.

[59] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing partial rankings. *SIAM Journal on Discrete Mathematics*, 20:47–58, 2004.

[60] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 28–36, 2003.

[61] S. Fox, K. Kulddep, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM TOIS*, 23(2):147–168, 2005.

[62] Peter Frazier. *Knowledge-Gradient Methods for Statistical Learning*. PhD thesis, 2009.

[63] Scott Freeman and John W. Parks. How accurate is peer grading? *CBE-Life Sciences Education*, 9(4):482–488, 2010.

[64] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[65] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[66] J. Gao, W. Yuan, X. Li, K. Deng, and J-Y. Nie. Smoothing clickthrough data for web search ranking. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 355–362, 2009.

[67] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2 edition, March 2011.

[68] David F. Gleich and Lek-heng Lim. Rank aggregation via nuclear norm minimization. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 60–68, 2011.

[69] John Guiver and Edward Snelson. Bayesian inference for plackett-luce ranking models. In *International Conference on Machine Learning (ICML)*, pages 377–384, 2009.

[70] Qi Guo, Haojian Jin, Dmitry Lagun, Shuai Yuan, and Eugene Agichtein. Mining touch interaction data on mobile devices to predict web search result relevance. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 153–162, 2013.

[71] Jonathon Haber. `http://degreeoffreedom.org/between-two-worlds-moocs-and-assessment`.

[72] Jonathon Haber. `http://degreeoffreedom.org/mooc-assignments-screwing/`, October 2013.

[73] Ahmed Hassan, Yang Song, and Li-wei He. A task level metric for measuring web search satisfaction and its application on improving relevance estimation. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 125–134, 2011.

[74] Ahmed Hassan and Ryen W. White. Task tours: helping users tackle complex search tasks. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 1885–1889, 2012.

[75] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. pages 115–132, 1999.

[76] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. 1999.

[77] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{tm}}$: A bayesian skill rating system. In *Advances in Neural Information Processing Systems (NIPS)*, pages 569–576, 2007.

[78] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45:615–627, 1998.

[79] Panagiotis G. Ipeirotis and Praveen K. Paritosh. Managing crowdsourced human computation: a tutorial. In *World Wide Web Conference (WWW)*, pages 287–288, 2011.

[80] T. Joachims, L. Granka, Bing Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Science (TOIS)*, 25(2), April 2007.

[81] Thorsten Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[82] Thorsten Joachims. Optimizing search engines using clickthrough data. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[83] Thorsten Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

[84] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[85] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 154–161, 2005.

[86] Thorsten Joachims and Karthik Raman. Bayesian Ordinal Aggregation of Peer Assessments: A Case Study on KDD 2015.

[87] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 699–708, 2008.

[88] Katy Jordan. `http://www.katyjordan.com/MOOCproject.html`, February 2013.

[89] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

[90] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[91] Emilie Kaufmann, Olivier Capp, and Aurlien Garivier. On bayesian upper confidence bounds for bandit problems. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 592–600, 2012.

[92] John G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):pp. 577–591, 1959.

[93] Maurice G. Kendall. *Rank correlation methods*. Griffin, London, 1948.

[94] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *STOC*, pages 95–103, 2007.

[95] Christian Kohlschutter, Paul-Alexandru Chirita, and Wolfgang Nejdl. Using link analysis to identify aspects in faceted web search. In *SIGIR Faceted Search Workshop*, pages 55–59, 2006.

[96] Weize Kong and James Allan. Extracting query facets from search results. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 93–102, 2013.

[97] Alexander Kotov, Paul N. Bennett, Ryen W. White, Susan T. Dumais, and Jaime Teevan. Modeling and analysis of cross-session search tasks. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 5–14, 2011.

[98] Jon A. Krosnick. Survey research. *Annual Review of Psychology*, 50(1):537–567, 1999. PMID: 15012463.

[99] Alex Kulesza and Ben Taskar. Learning determinantal point processes. 2011.

[100] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott Klemmer. Peer and self assessment in massive online classes. *ACM Trans. Comput.-Hum. Interact.*, 20(6):33:1–33:31, December 2013.

[101] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 120–127, 2001.

[102] Guy Lebanon and John D. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *International Conference on Machine Learning (ICML)*, pages 363–370, 2002.

[103] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

[104] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[105] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 897–904, 2007.

[106] Zhen Liao, Yang Song, Li-wei He, and Yalou Huang. Evaluating the effectiveness of search task trails. In *World Wide Web Conference (WWW)*, pages 489–498, 2012.

[107] Daniel J. Liebling, Paul N. Bennett, and Ryen W. White. Anticipatory search: using context to initiate search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1035–1036, 2012.

[108] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2:285–318, April 1988.

[109] Jingjing Liu and Nicholas J. Belkin. Personalizing information retrieval

for multi-session tasks: the roles of task stage and task type. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 26–33, 2010.

[110] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

[111] Lori Lorigo, Maya Haridasan, Hrönn Brynjarsdóttir, Ling Xia, Thorsten Joachims, Geri Gay, Laura Granka, Fabio Pellacini, and Bing Pan. Eye tracking and online search: Lessons learned and challenges ahead. *Journal of the American Society for Information Science and Technology*, 59(7):1041–1052, 2008.

[112] Tyler Lu and Craig Boutilier. In *International Conference on Machine Learning (ICML)*, pages 145–152, 2011.

[113] Tyler Lu and Craig E. Boutilier. The unavailable candidate model: A decision-theoretic view of social choice. In *European Commerce (EC)*, pages 263–274, 2010.

[114] R. Duncan Luce. *Individual Choice Behavior: A theoretical analysis*. Wiley, 1959.

[115] Heng Luo. `http://sloanconsortium.org/conference/2014/et4online/peer-grading-valid-assessment-method-massive-open-onli` April 2014.

[116] C. L. Mallows. Non-null ranking models. *Biometrika*, 44(1/2):pp. 114–130, 1957.

[117] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[118] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.

[119] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, March 1956.

[120] Markus Mostert and Jen D. Snowball. Where angels fear to tread: on-

line peer-assessment in a large first-year class. *Assessment & Evaluation in Higher Education*, 38(6):674–686, 2013.

[121] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[122] John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[123] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 663–670, 2000.

[124] Shuzi Niu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Stochastic rank aggregation. *CoRR*, abs/1309.6852, 2013.

[125] Maeve O'Brien and Mark T. Keane. Modeling user behavior using a search-engine. In *International Conference on Intelligent User Interfaces (IUI)*, pages 357–360, 2007.

[126] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller. Tuned models of peer assessment in MOOCs. In *EDM*, 2013.

[127] R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2):193–202, 1975.

[128] Jeffrey Pound, Stelios Paparizos, and Panayiotis Tsaparas. Facet discovery for structured web search: a query-log mining approach. In *ACM International Conference on Management of data (SIGMOD)*, pages 169–180, 2011.

[129] Tao Qin, Xiubo Geng, and Tie-Yan Liu. A new probabilistic model for rank aggregation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1948–1956, 2010.

[130] Pernilla Qvarfordt, Gene Golovchinsky, Tony Dunnigan, and Elena Agapie. Looking ahead: query preview in exploratory search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 243–252, 2013.

[131] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *National Conference on Artificial Intelligence (AAAI)*, pages 1406–1412, 2006.

[132] Filip Radlinski, Paul N. Bennett, Ben Carterette, and Thorsten Joachims. Redundancy, diversity and interdependent document relevance. *SIGIR Forum*, 43(2):46–52, December 2009.

[133] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *International Conference on Machine Learning (ICML)*, pages 784–791, 2008.

[134] Karthik Raman, Paul N. Bennett, and Kevyn Collins-Thompson. Toward whole-session relevance: exploring intrinsic diversity in web search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 463–472, 2013.

[135] Karthik Raman, Paul N. Bennett, and Kevyn Collins-Thompson. Understanding intrinsic diversity in web search: Improving whole-session relevance. *ACM Trans. Inf. Syst.*, 32(4):20:1–20:45, October 2014.

[136] Karthik Raman and Thorsten Joachims. Learning Socially Optimal Information Systems from Egoistic Users. In *European Conference on Machine Learning (ECML)*, pages 128–144, 2013.

[137] Karthik Raman and Thorsten Joachims. Methods for ordinal peer grading. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1037–1046, 2014.

[138] Karthik Raman and Thorsten Joachims. Bayesian ordinal peer grading. In *ACM Learning at Scale Conference (LAS)*, pages 149–156, 2015.

[139] Karthik Raman, Thorsten Joachims, and Pannaga Shivaswamy. Structured learning of two-level dynamic rankings. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 291–296, 2011.

[140] Karthik Raman, Thorsten Joachims, Pannaga Shivaswamy, and Tobias Schnabel. Stable Coactive Learning via Perturbation. In *International Conference on Machine Learning (ICML)*, pages 837–845, 2013.

[141] Karthik Raman, Pannaga Shivaswamy, and Thorsten Joachims. Online

learning to diversify from implicit feedback. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 705–713, 2012.

[142] Karthik Raman, Krysta M. Svore, Ran Gilad-Bachrach, and Chris J. C. Burges. Learning from mistakes: towards a correctable learning algorithm. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 1930–1934, 2012.

[143] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research (JMLR)*, 11:1297–1322, August 2010.

[144] Jonathon Rees. `http://www.insidehighered.com/views/2013/03/05/essays-flaws-peer-grading-moocs`, March 2013.

[145] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[146] Stephen E Robertson. The probability ranking principle in ir. *Journal of documentation*, 33(4):294–304, 1977.

[147] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.

[148] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[149] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. Selectively diversifying web search results. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 1179–1188, 2010.

[150] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[151] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[152] Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. In *International Conference on Machine Learning (ICML)*, 2012.

[153] Adish Singla, Ryen White, and Jeff Huang. Studying trailfinding algorithms for enhanced web search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 443–450, 2010.

[154] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *The Journal of Machine Learning Research*, 14(1):399–436, 2013.

[155] Neil Stewart, Gordon D. A. Brown, and Nick Chater. Absolute identification by relative judgment. *Psychological Review*, 112:881–911, 2005.

[156] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[157] Ashwin Swamintahan, Cherian Metthew, and Darko Kirovski. Essential pages. In *Technical Report, MSR-TR-2008-15*, Microsoft Research, 2008.

[158] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *International Conference on Machine Learning (ICML)*, pages 896–903, 2005.

[159] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. 2004.

[160] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):pp. 285–294, 1933.

[161] L. L. Thurstone. The method of paired comparisons for social values. *Journal of Abnormal and Social Psychology*, 27:384–400, 1927.

[162] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. pages 1453–1484, 2005.

[163] J. J. Veloski, H. K. Rabinowitz, M. R. Robeson, and P. R. Young. Patients don't present with five choices: an alternative to multiple-choice tests in assessing physicians' competence. *ACADEMIC MEDICINE-PHILADELPHIA-*, 74(5):539–546, 1999.

[164] Maksims N. Volkovs and Richard S. Zemel. A flexible generative model

for preference aggregation. In *World Wide Web Conference (WWW)*, pages 479–488, 2012.

[165] Andrii Vozniuk, Adrian Christian Holzer, and Denis Gillet. Peer Assessment Based on Ratings in a Social Media Course. In *LAK*, 2014.

[166] R.W. White and S.M. Drucker. Investigating behavioral variability in web search. In *World Wide Web Conference (WWW)*, pages 21–30, 2007.

[167] Ryen White. Beliefs and biases in web search. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–12, 2013.

[168] Ryen W. White, Paul N. Bennett, and Susan T. Dumais. Predicting short-term interests using activity-based search context. In *ACM Conference on Information and Knowledge Management (CIKM)*, pages 1009–1018, 2010.

[169] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[170] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *International Conference on Machine Learning (ICML)*, pages 1192–1199, 2008.

[171] Peng Ye and David Doerman. Combining preference and absolute judgements in a crowd-sourced setting, June 2013. ICML'13 workshop: Machine Learning Meets Crowdsourcing.

[172] H. Peyton Young. Condorcet's Theory of Voting. *The American Political Science Review*, 82:1231–1244, 1988.

[173] Xiaojun Yuan and Ryen White. Building the trail best traveled: effects of domain knowledge on web search trailblazing. In *Conference on Human Factors in Computing Systems (CHI)*, pages 1795–1804, 2012.

[174] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[175] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2483–2491, 2012.

[176] Yisong Yue and Thorsten Joachims. Predicting diverse subsets using structural svms. In *International Conference on Machine Learning (ICML)*, 2008.

[177] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *International Conference on Machine Learning (ICML)*, pages 1201–1208, 2009.

[178] Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 10–17, 2003.

[179] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 363–370, 2010.

[180] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, 2003.