Understanding User Goals by Analyzing Logged Interactions and Asking the Right Questions

Anna Sepliarskaia

Understanding User Goals by Analyzing Logged Interactions and Asking the Right Questions

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op 19 November 2019, te 12:00 uur

door

Anna Sepliarskaia

geboren te Kuibyshev

Promotiecommissie

Promotor:		
	prof. dr. M. de Rijke	Universiteit van Amsterdam
Co-promotor:		
	dr. F. Radlinski	Google
	dr. Y. Kiseleva	Microsoft
Overige leden:		
	dr. Z. Akata	Universiteit van Amsterdam
	dr. S. Genc	Amazon Seattle
	prof. dr. H. Haned	Universiteit van Amsterdam
	prof. dr. M.A. Larson	Radboud Universiteit Nijmegen
	prof. dr. S.J.L. Smets	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the Microsoft Research Ph.D. program under project number 2014-056.

Cover by Antonis Krasakis and Anna Sepliarskaia, drawings by Elizaveta Muzychka, Anna Sepliarskaia, Antonis Krasakis, Yifan Chen, Spyretta Leivaditi, Chang Li, Jie Zou, Georgios Sidiropoulos
Printed by Off Page, Amsterdam, The Netherlands

ISBN: xxx-xx-xxxx-xxx-x

Acknowledgements

Doing a PhD was a great time for me. I enjoy the academic environment and freedom of making research that fits my interests. I don't know whether I grew personally but I am definitely a different person than 4 years ago with a strong desire to continue my path as a researcher. This transformation would not be possible without the help, support and guidance from my supervisors, friends, and family.

First of all, I would like to thank my main supervisor Prof. Maarten de Rijke, who trusted, supported and navigated my research. I impressed with your attitudes towards research and your students. You created a great research team ILPS and I was lucky to be a part of it. During my PhD you are the one who helped me to be not only an independent researcher but also improve my ability to be a part of a team. Moreover you said one of the most important thing for me as a researcher during our first meeting: "Use your mathematical background". It took me some time to follow this advice, but eventually, it leads me to the point, where I am, finishing my PhD.

I also thank my co-promoter, dr.Julia Kiseleva. Though we had some misunderstanding, in the beginning, we successfully overcome them. You always fight for the papers and most importantly to their quality. These characteristics helped a lot to improve the papers we wrote together.

In addition, I thank my co-promoter, dr.Filip Radlinski. You helped me a lot with the first papers.

I am honored to have Sahika, Zeynep, Hinda, Martha and Sonja to be my committee members. Now I can be sure that at least 5 great scientists read my thesis.

I thank all the members of ILPS, that I met. While working with them I knew,that I can always have a useful discussion about research. Thanks a lot: Ali A, Ali V, Alexey, Amir, Ana, Andreas, Arezo, Artem, Bob, Boris, Chang, Christof, Christophe, Chuan, Dan, Dat, David D, David S, Evangelos, Fei, Hamid, Harrie, Hinda, Hosein, Ilya, Jiahuan, Jie, Jin, Julia, Julien, Katya, Ke, Maarten dR, Maarten M, Maartje, Mahsa, Mariya, Marzieh, Maurits, Mostafa, Mozhdeh, Nikos, Oliver, Pengjie, Petra, Praveen, Rolf, Sami, Shaojie, Spyretta, Svitlana, Trond, Vera, Wanyu, Xiaohui, Xinyi, Yangjun,Yifan, Zhaochun, and Ziming. Especially, thanks Xinyi, Chang, Yifan, Spyretta, and Svitlana to be not only colleagues but also great friends.

My PhD studies were not only ILPS. I was lucky to spend 4 months as an intern at Amazon. Thanks, Sahika for acquaintance me with reinforcement learning world.

Finally, I also would like to thanks people that were involved in my PhD, but their presence was not so direct and obvious. First, I would like to thanks Lisa, which name is not present in any of my papers, but you definitely were my permanent coauthor. I am very grateful to have such a great daughter. Thanks, Vovan, Irchik and Varvara for supporting me and showing me the beauty of biology. Tanya and Misha you always help me to choose the right path. Miranda, Svitlana, Evgeny and Galka you are the wonderful friends and researchers. You explore various parts of the scientific world, but you all see the beauty of it and share it with me.

Anna Sepliarskaia
September 25th, 2019

Contents

1
Introduction

Search engines and recommender systems have become an essential part of everyday life [41]. People use them for a variety of tasks — from finding a specific recipe [121], book [110], or movie [23] that matches their preferences, to shopping online [67]. These systems should provide not only information that is relevant to a user's query and up to date, but also satisfy user preferences, that is, be *personalized*. Indeed, users are diverse and have different information needs, depending on their background, personal preference and so on [10]. For example, when searching for a scientific term different users may have different knowledge and expertise in the field. The documents returned by the system should vary for a person who just starts learning the subject and for the expert. When shopping online, users who want to buy a black dress, mean different dresses and will buy different dresses, although they submit the same initial query. Personalized systems actively promote documents or items that are relevant to a user's interest and might be missed due to the large number of relevant documents for the current query [10]. To be able to accurately perform personalization, the system collects information about the users and infer a user's *profile* [10].

There are two types of signal that are gathered about users to generate their profile: *explicit* and *implicit* [52]. To collect explicit signals, users are encouraged to take the initiative and explicitly provide information about their preferences. For example, users can be asked to complete a questionnaire [48, 95, 97], to provide a natural language response [40, 97], to draw [71], to provide ratings or to indicate preferences about some items [33, 72, 89]. The algorithmic task that arises in this setting is what questions to ask users. For example, if users fill out a questionnaire, the system should understand which questions to ask in order to get the most information about users. The advantage of using explicit signals is that they are reliable. However, the system burdens users by asking them to provide information, therefore, relies on the users' willingness to do so.

Implicit signals are signals from user interactions with the system such as clicks, document viewing time, purchase history, history of adding to cart, and browser history. A problem that arises when using implicit signals to infer a user's profile is that these signals are noisy. To illustrate the challenge of this problem, let us look at one of the most common signals — user clicks. It is known that users usually examine the search engine result page from top to bottom, and therefore the majority of users does not explore all displayed documents [20] (this phenomenon is called *position bias*). Consequently, even if the last document is the most relevant, users can click on the first

document without any feedback on the last one, simply because they do not see the last document. The advantage of using methods that derive a user profile from implicit feedback is that users provide a lot of implicit signals and this information about users is available without disturbing them. However, these signals are noisy, and the system sometimes promotes documents or items that do not satisfy the user's taste.

A user profile consists of two components: *long-term user preferences* and *short-term preferences* [60, 87, 129]. Long-term preferences are preferences that represent lasting user interests. They are reflected in most cases of a user's interactions with the system. Therefore, long-term user preferences can be investigated using behavior signals from all previous user sessions, or in other words, from *long-term behavior signals*, where a session is defined as a continuous series of interactions between a user and a system.

To be able to understand a user's long-term preferences is important, but research has found that the current context or, in other words, the user's short-term preferences strongly influences her preferences [60]. Short-term preferences are present only in the current session and can be inferred only from it, or in other words only from *short-term behavior signals*. Together, both short-term and long-term user preferences help to identify relevant documents to display on the result page. For example, a user may prefer to watch drama movies in general. Thus, the preference for drama is a long-term user preference. But the concrete type of drama films may vary depending on the mood of the user. In one evening she may prefer to watch historical drama, while in another she prefers to watch docudrama. The current mood of a user is reflected in her short-term preference. The fact that the user loves dramatic films can be derived from past interactions. What kind of drama the user wants to watch during the current evening, the system can only understand from the current user session. In the domain of online shopping, long-term preferences usually determine the preferred style of clothing for users. While the user's short-term preferences reflect the type of clothing she wants to buy now.

In this thesis, we investigate solutions that allow us to infer users preferences at the lowest cost to users:

(1) Firstly, we study how to infer user's preferences, using implicit signals from all sessions to perform personalized re-ranking of a non-personalized list of documents.

(2) Secondly, we use explicit feedback to infer a user's preferences. In particular, we ask users to fill out a questionnaire consisting of pairwise preference questions and we investigate which questions to include in the questionnaire to make it informative.

(3) Then, we combine users' long-term and short-term preferences to improve a list of recommendations in the current session:

   (a) we use implicit signals from users' historical logs to infer long-term preferences;

   (b) we ask users to provide explicit feedback by specifying relative preferences in the current session to understand short-term preferences.

We investigate which questions to ask in order to infer information about a user's current need, with the constraint that the shown items satisfy a user's preferences.

(4) Finally, we investigate how to make a relative preference questionnaire both interpretable and informative. To this end, we need a representation of the items that captures all the information about them and in which each factor in the representation reflects only one interpretable factor of variation in the collection. This type of representation is called a *disentangled* representation. We propose a metric of disentanglement of representations.

1.1 Research Outline and Questions

We divided the thesis based on what types of signals are used for personalization. The thesis covers three general research themes:

(1) inferring long-term user preferences using implicit signals (Chapter 2);

(2) generating questionnaires to infer a user's short-term preferences using explicit signals (Chapters 3, 5);

(3) inferring a user profile by combining implicit signals from historical logs and explicit signals from a completed questionnaire (Chapter 4).

Below, we list the main research questions that we study in this thesis.

1.1.1 Inferring long-term user preferences from implicit signals

Search engines and recommender systems have become an essential component of everyday life [41]. As a result, a lot of information about users is being collected. In the case of search engines, users usually interact with the system by submitting queries, clicking on documents, reformulating queries, exploring documents [10, 51]. From these interactions we can derive topics of interest to users [10]. However, most of the models that utilize this information use feature engineering and, therefore, they require some expert knowledge [79, 112, 125]. In Chapter 2 we use past user interactions to infer user profiles without feature engineering. More precisely, we answer the following research question:

RQ1 How to infer a user's long-term preferences, using only their click interactions with the system?

To answer this question, we use click signals. It is known that a user's clicks are a very important indicator of her interests [10, 19]. However, clicks are biased: the number of clicks on a document depends on the probability that a user will explore the document [19, 53, 77]. Models that infer the relevance of documents using click signals are called *click models* [19]. Typically, a click model first assumes some expected user behavior and then calculates the relevance of documents taking into account the behavior of users [19]. Most click models assume that the likelihood that a user will examine a document depends on the position in which the document was displayed [19].

In addition, it is usually assumed that a user clicks on a document only if the document has been examined [19]. For example, the cascade model [20] assumes that a user scans documents on the result page from top to bottom until they find the relevant document. Under this assumption, all documents that are displayed higher than the clicked document are always examined, while all the documents below the clicked document are never examined. Click models are used a lot in search engines, but only few click models are personalized [130]. In Chapter 2 we propose a click model, which in addition to previous click models has a parameter that reflects a user's taste: the attractiveness of documents for a user.

1.1.2 Creating a pairwise preference questionnaire to collect explicit signals

Implicit signals are very effective in determining a user's long-term preferences [52]. However, they cannot be used when a user is new to a system. The problem of inferring preferences of a new user is very common in recommender systems and is called *new user cold-start problem* [36, 48]. The new user cold-start problem arises in recommender systems because one of the most successful approaches to a recommendation cannot give a good recommendation without users' interaction history.

In particular, though many approaches to generating recommendations have been proposed in the past several years [75], one of the most common approaches is *Collaborative Filtering* (CF) [10]. CF-based methods use past interactions between users and items to infer users' preferences and important properties of the items. For example, in the movie domain, movies that a user watched are marked as movies that match the user's preferences. From this type of user feedback, CF-based methods can infer that some users like comedies and others like documentaries. Moreover, CF approaches infer that some movies are comedies, while some are documentaries. Using such predictions, CF-based methods recommend to users who like documentaries only documentary movies. CF-based methods are not only very effective, but they are also domain agnostic. That is why they are used in a lot of domains, such as song recommendation [100], movie recommendation [83], news recommendation [21], fashion recommendation [44]. Although CF approaches have achieved great success, as we already said, they suffer from new user cold-start problem. In Chapter 3, we study one of the solutions to the new user cold-start problem for CF-based methods. More precisely, we study the following research problem:

RQ2 How to create an informative preference questionnaire to infer cold users' preferences?

In the case when a user is new to the system, explicit signals are much more efficient than implicit ones [52]. To illustrate this, let us take as a practical example of a recommender system in which there are hundreds of thousands of items and users like an only small fraction of them. It is almost impossible to guess which items will satisfy a user's preferences without any knowledge about that user. This means that the system starts showing to a user documents that are all equally non-relevant to her. Consequently, user interactions with the system do not contain useful information. The common approaches

to the new user cold-start problem are questionnaire-based [48, 95] Usually, users are asked to provide absolute ratings for some items [18, 72, 89]. These ratings can be used in exactly the same way as all others to infer a user's preferences by CF-based methods.

However, absolute ratings suffer from several problems. For example, the same rating, of say three out of five, can mean different things to different people [56]. For this reason, in Chapter 3 we create a questionnaire that consists of *relative preference* questions. That is, a user is asked to answer which of the shown items are more relevant to her. In particular, in Chapter 3 we formulate the problem of constructing a pairwise preference questionnaire as an optimization problem and create an efficient algorithm that solves it.

1.1.3 Combining implicit and explicit signals to infer user profiles

Explicit and implicit signals are very useful signals for understanding users' preferences, however, we should combine them to achieve good results [10, 52]. In Chapter 4 we focus on this problem. In particular, although a user's long-term preferences remain the same for almost all requests, the specific document that a user wants to find varies from session to session and depends on their short-term preferences. That is why it is important to combine a user's preferences in order to show a document that is relevant for a user in the current session [10, 52].

In this thesis, we use implicit signals to understand a user's long-term preferences, while for understanding their short-term preferences we use explicit signals. We use implicit signals to determine long-term preferences because for many users there are many past interactions with the system, and past interactions cannot be used to obtain user's short-term preferences. Consequently, despite the fact that implicit signals are noisy, due to their number it is still possible to reliably infer the user's long-term preferences. On the other hand, a session usually does not last long, therefore, there are not so many implicit signals that can be used to infer a user's short-term preferences. That is why we ask users to explicitly indicate their preferences, by clicking on the best document from the displayed documents. Also, we understand that to show to users documents that are informative but not relevant will bother her. That is why we study the following research question:

RQ3 How to find the optimal strategy for selecting documents to display, which helps to increase user satisfaction with an ongoing session, combining short-term and long-term preferences?

User satisfaction with a session is based on the overall search experience including information gain as well as the effort spent on an examination result pages [113]. We model user satisfaction with a session as the sum of user satisfaction with pages shown during the session, where user satisfaction with a page depends not only on the relevance of the documents displayed on the page but also on the time that the user spent to reach this page, i.e., on the page sequence number in the session. We approximate user satisfaction with a page as a multiplication of relevance of the page with a discount factor that depends on the page sequence number in the session. This setting can be viewed as a game in which the system interacts with a user and receives rewards equal to the user satisfaction with the shown pages. Each episode of the game is a session of

one user, and the task of the system is to maximize the reward. The standard approach to finding an optimal strategy for a game is a Reinforcement Learning (RL) approach [84].

The difficulty of applying an RL method to the recommendation task is that the number of actions, that is, the number of all possible result pages, is very large. Moreover, the current state of the game is a continuous high dimensional vector, since a user profile inferred so far depends on all of the user's past interaction with the system, i.e., it depends on her interactions in previous sessions and the answers given during the current session. In Chapter 4 we propose to overcome these problems by using a model-free Deep Reinforcement Learning [69]-based algorithm.

1.1.4 Creating interpretable and informative questions to collect explicit signals

In the previous sections, we focused on asking informative questions and to use answers to them to infer a user profile. However, we did not take into account one important aspect of the questionnaire, namely the difficulty of the questions to users. In the previous sections, we assumed that a user always spends the same effort to give an answer, and that she always gives the correct answers. However, this is not always true, even for questions about preferences [89]. For example, in a movie recommendation domain, when we ask a user to elicit her preferences about some movies, she sometimes cannot answer because she did not watch them [89]. To take this possibility into account, some methods ask questions only about popular items [89].

Another case when users cannot answer questions about preferences arises in fashion search. Suppose a person wants to buy a pair of white shoes with high heels. If the system shows a user two items, e.g., one being white shoes with a flat sole and the other a pair of black high-heeled shoes, it is difficult to determine which pair of shoes is more suitable for the user. One solution to this problem is to show to users products that differ only in one attribute. For example, we can show to a user the shoes with the same color and shape, but with a different length of the heels: from flat shoes to the shoes with a high heel. To this end, we need a latent representation of the items such that each dimension in the latent representation reflects only one attribute, and different latent dimensions capture different attributes. These latent representations are called *disentangled* latent representations. Although it is generally accepted that disentangled latent representations are very important to build, so far, there is no single metric and notion of disentanglement [5]. That is why in Chapter 5 we answer the following research question:

RQ4 How to measure the disentanglement of latent representations?

While there is no single formalized notion of disentangled representation, the key intuition is that a disentangled representation should capture and separate the generative factors [5]. Recently, researchers have come to understand the importance of building disentangled representations and, hence, several methods that aim to build a disentangled representation of the items in the collection have been proposed [22, 46, 58, 63]. But due to the lack of a standard disentanglement metric, authors of newly proposed representation learning algorithms also propose new disentanglement metrics. However,

disentanglement metrics have usually not been the main purpose of the work published preeviously, and therefore a formal analysis of previously proposed metric has not given. In Chapter 5 we formally analyze all previously proposed disentanglement metrics and propose a new one that satisfies theoretical guarantees.

1.2 Main Contributions

In this section, we list theoretical, algorithmic and empirical contributions of this thesis. For each contribution, we list the chapter from which it originates.

1.2.1 Theoretical contributions

(1) Formulation of the task of generating a questionnaire consisting of pairwise preference questions that help to solve the new user cold-start problem; and an algorithm, called *static preference questionnaire* (SPQ), that solves the optimization problem with theoretical guarantees (Chapter 3).

(2) Formulation of the interaction between a recommender system and a user as a *Markov Decision Process* (MDP) (Chapter 4).

(3) Theoretical framework that combines short-term user preferences and long-term user preferences to increase user satisfaction with the session (Chapter 4).

(4) A theoretical analysis of existing metrics of disentanglement, in which we investigate whether the metrics give a high score to all disentangled representations and a low score to all entangled representations (Chapter 5).

(5) Proposal of a new metric of disentanglement with theoretical guarantees (Chapter 5).

1.2.2 Algorithmic contributions

(6) An algorithm that uses implicit signals to infer user profiles taking into account *personalization and ranks of the document and attractiveness* (PRA) (Chapter 2).

(7) A Reinforcement Learning method that is based on the actor-critic framework (ActorCriticRS) to find an optimal strategy to select documents to display on the result page (Chapter 4).

1.2.3 Empirical contributions

(8) An empirical comparison of PRA with other state-of-the-art methods for personalization that use implicit signals (Chapter 2).

(9) An empirical comparison of SPQ with other methods for creating questionnaires consisting of preference questions (Chapter 3).

(10) An empirical comparison of ActorCriticRS with other state-of-the-art strategies for selecting items to display on the result page (Chapter 4).

1.3 Thesis Overview

In this thesis, we investigate how to infer a user profile from her interactions with an interactive system and how to ask users informative and interpretable questions. In particular, in Chapter 2 we infer long-term user preferences using implicit signals.

Then, in Chapter 3 we propose a new method for creating informative questionnaires consisting of pairwise preference questions.

In Chapter 4 we propose a reinforcement learning method that combines implicit and explicit signals in order to select elements to display on the result page and to increase user satisfaction with the session.

In Chapter 5 we look deeper into the problem of creating preference questionnaires: we would like the questionnaire to be not only informative but also interpretable. To this end, we need a disentangled representation of the items and consequently a metric of disentanglement. That is why in Chapter 5 we analyze existing metrics of disentanglement and propose a new one.

Finally, in Chapter 6 we conclude the thesis and discuss limitations and future directions.

1.4 Origins

In this section, we list publications that form the basis of this thesis. Each research chapter is based on a paper. We provide references to these publications and explain the roles of the co-authors.

- Chapter 2 is based on the following paper:
  - Anna Sepliarskaia, Filip Radlinski, and Maarten de Rijke. "Simple Personalized Search Based on Long-term Behavioral Signals". In: *European Conference on Information Retrieval*. Springer, 2017, pp. 95–107.

  AS designed the algorithm, ran the experiments, and did most of the writing; FR and MdR contributed to the writing.

- Chapter 3 is based on the following paper:
  - Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. "Preference Elicitation as an Optimization Problem". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM. 2018, pp. 172–180.

  AS designed the algorithm, ran the experiments, and did most of the writing; JK, FR and MdR contributed to the writing.

- Chapter 4 is based on the following paper:
  - Anna Sepliarskaia, Sahika Genc, and Maarten de Rijke. "A Deep Reinforcement Learning-Based Approach to Query-Free Interactive Target Item Retrieval". In: *IEEE Transactions on Multimedia* (2019). Under review.

AS designed the algorithm, ran the experiments, and did most of the writing; SG helped with the algorithm design and contributed to the writing; MdR contributed to the writing.

- Chapter 5 is based on the following paper:
  - Anna Sepliarskaia, Kiseleva Julia, and Maarten de Rijke. "Evaluating Disentangled Representations". In: *Advances in Neural Information Processing Systems*. Under review. 2019.

  AS designed the metric and did most of the writing; JK and MdR contributed to the writing.

2

Simple Personalized Search Based on Long-Term Behavioral Signals

In this chapter, we address RQ1: how to infer a user's long-term preferences, using only their click interactions with the system. We present a new approach to incorporating click behavior into document ranking, using ideas from click models and learning to rank.

2.1 Introduction

Search engines today combine numerous types of features when producing a ranking for a given query. For instance, the learning to rank datasets made available in recent years have between 140 and 700 ranking features. They must provide ranked lists of results that are relevant (based on content), engaging (based on past user engagement), timely, and personally of interest to the user. These competing goals have led to a vast amount of work on each of them. Our focus is on personalization, which involves reranking documents on the search engine result page (SERP) so as to better satisfy a particular user's information need.

We present a novel approach to personalize search results with a model that is as effective as current state-of-the-art approaches, yet much simpler. By starting with a ranking produced by a commercial search engine, we know that the content of the top retrieved results is already likely to be of high relevance. However, we observe that usage still differentiates users and use this fact to rerank retrieval results based on implicitly collected usage. Consider, for instance, queries with only one intent but with a wide variety of relevant links such as "information retrieval conference." Links to SIGIR, ECIR, ICTIR, as well as links to general information on conferences are likely to be relevant. But each user has her own conference preference, which the system can infer from the user's past behavior—even if the user may be unable to formulate this preference directly in a query.

Previous research on personalizing search using behavioral data has found that to improve the ranking for a given user, information from the user's short-term and long-term behavior can be used [6, 13, 109]. Here, short-term behavior is information

This chapter was published as [106].

from the session in which the user is currently engaged; long-term behavior concerns information from all of the user's search history. We focus on the use of long-term behavior for personalizing search as long-term behavioral signals have led to larger improvements than short-term behavioral signals [112, 6]. Also, short-term features cannot be used for the first query of a session, and over 40% of all sessions are of this sort [109].

At a high level, our approach calculates document scores given a query issued by a user, for each document d in the SERP. The score is a simple function combining three components: how well the document matches the query, how likely the user is to engage with documents at a given position, and how likely a user is to engage with a particular document. Perhaps surprisingly, despite not relying on handcrafted rules or sophisticated feature engineering, we show that performance is competitive with state-of-the-art models. Thus our key contribution is to show that formulating the optimization problem in this way removes the necessity for previously published complexity. We anticipate that by learning a simpler model, personalize reranking becomes more generally applicable, less complex computationally, and less error prone.

2.2 Related work

There are several approaches to addressing personalized search, each with its own benefits and drawbacks. First, one needs to understand when reranking is needed. The distinction of queries in three types—navigational, informational and transactional—is well-known [9]. Users submitting navigational and transactional queries use search engines to retrieve easily findable and recognizable target results; for most navigational and transactional queries reranking is well understood [120, 80]. Teevan, Liebling, and Geetha [120] show an easy and low-risk Web search personalization approach for navigational queries. Their approach achieves more than 90% accuracy. However, it works on the small segment of queries that the same user has issued at least three times. Query ambiguity is one of the indicators to inform us about changing the order of documents. Features and measures to predict it are proposed in [119]. If multiple documents have a high probability of being clicked following the query, then there is a great potential to improve the ranker.

The second type of related work concerns click models. Click models use implicit feedback to predict the probability of clicks [19]. Clicks can be a good indicator of failure or success. Features from click models are very useful for ranking documents [55, 54]. However, few click models are personalized [108]. As click models use implicit feedback, manual assessment is not required nor is feature engineering. These models work well for improving the click through rate (CTR). However, to re-rank URLs the relative order of predicted relevance is more important than absolute CTR value [15]. The click model that achieves the best performance for predicting probability of click is the User Browsing Model (UBM) [26]. The main difference between UBM and other models is that UBM takes into account the distance from the current document to the last clicked document for determining the probability that the user continues browsing.

The third type of approach to behavior-based personalized search uses feature engineering to create behavior features and then learn a ranking function [79, 112, 125].

Work that follows this approach differs in the choice of machine learning algorithms used. LambdaMart [12] is used in [79]. Several learning to rank algorithms as well as regression models are used in [125]. Logistic regression is used in [112]. Cai, Liang, and Rijke [13] and Cai, Wang, and Rijke [14] use matrix factorization and restrict themselves to users with a sufficient volume of interactions. All of them devote significant attention to feature engineering. For example, Masurel, Lefévre-Hasegawa, Bourguignat, and Scordia [79] use the probability that the user skips, clicks or misses the documents. The winners of the 2014 Kaggle competition on personalized search use over 100 features [79].

2.3 Method

We begin by providing a general description of our personalized search method and the intuitions behind it. At a high level, our goal is to obtain a simple yet effective model. The simplicity is achieved by an easily interpretable function that scores documents. The document score reflects the probability that the document is relevant, which depends on three random variables: attractiveness of the document to the user, attractiveness of the document to the query and examination of the rank of the document. The uniqueness of our approach is that, in contrast to previous models, we do not optimize the log likelihood of click probability but explicitly fit the probability that one document is more relevant than another in the SERP.

Our method shares traits of learning to rank methods and click models. Inspired by approaches in non-personalized pairwise learning to rank, we explicitly model the probability that one document is more relevant than another one. As in click models, personalized reranking involves modeling the relevance of documents using historical personal interactions with them. Further, we propose to train our model using long-term behavioral signals, which can be compared with classical click models [26, 15, 70] in its simplicity and approach, but it is as effective as recent complex models.

In our algorithm, position bias is taken into account. We follow the position model [19], in which it is assumed that examination of URLs on a SERP is a function of their rank and does not depend on examinations and URLs at higher positions. However, we assume that examination also depends on the query. Moreover, we have a factor that reflects attractiveness of a document to a given query. None of these parameters are personalized, therefore, we introduce new ones that are user specific. We introduce only one type of user specific parameters in this chapter—attractiveness of a document to a specific user—but others could easily be integrated in a similar fashion.

We first introduce some notation:

(1) q denotes a query, r a rank, d a document, u a user;

(2) $e_{q,r}$ denotes the examination of a document at rank r in a SERP produced for q;

(3) $a_{q,d}$ is the attractiveness of document d for query q;

(4) $a_{u,d}$ is the attractiveness of d for user u.

We will use the sigmoid function

$$\sigma(x) = 1/(1 + \exp(-x))$$

and the indicator function

$$I(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{if } x \text{ is false.} \end{cases}$$

Given a query q submitted by user u, and a (non-personalized) SERP produced in response to q, our model re-ranks a document d that is originally placed at rank r in the SERP using the following scoring function:

$$score(q, d, u, r) = \sigma(a_{q,d}) \cdot \sigma(e_{q,r}) \cdot \sigma(a_{u,d}). \tag{2.1}$$

The learned parameters of the proposed model are $a_{q,d}, e_{q,r}, a_{u,d}$, which are single numbers. We use the sigmoid function to map these parameters to a probability.

We instantiate our model by training it based on implicit feedback from users. Given a query and user, we assume that the label of a given document is given by how the user interacts with it (clicks on it)—described specifically in Section 2.4. To achieve comparable results with the state-of-the-art model, we take inspiration from learning to rank methods and predict pairwise preferences of documents. More precisely, we map each document in the SERP to a number and the greater the difference between these numbers the higher the probability that one document is more relevant than another. Specifically, for a given tuple (query q and user u) each pair of URLs d_i and d_j in a SERP with different labels is chosen. For each such pair we compute the scores $s_i = score(q, d_i, u, i)$ and $s_j = score(q, d_j, u, j)$, by using the parameters $a_{q,d_i}, e_{q,r_i}, a_{u,d_i}, a_{q,d_j}, e_{q,r_j}, a_{u,d_j}$, that were received up to that step. Let $d_i \prec d_j$ denote the event that d_i should be ranked higher than d_j. The scores are mapped to a learned probability that d_i should be ranked higher than d_j via a sigmoid function:

$$p(d_i \prec d_j) = \sigma(s_i - s_j). \tag{2.2}$$

We use a gradient descent formulation to minimize the cross-entropy function for each pair of documents in the SERP:

$$C(d_i, d_j) = -I(d_i \prec d_j) \cdot \log(p(d_i \prec d_j)) - (1 - I(d_i \prec d_j)) \cdot \log(p(d_j \prec d_i)). \tag{2.3}$$

Our method consists of three phases: first it tunes $e_{q,r}$, then $a_{q,d}$, and finally $a_{u,d}$. At each step the training procedure uses stochastic gradient descent (SGD), sequentially scanning the list of SERPs, calculating the gradient of the loss function for a SERP as

$$C_{serp} = \sum_{d_i, d_j} C(d_i, d_j), \tag{2.4}$$

and updating parameters $p_{uqd_1}, \ldots, p_{uqd_{10}}$ according to the following equation:

$$p_{uqd_i} \mathrel{+}= \eta \cdot \frac{\partial C_{serp}}{\partial s_i} \cdot \frac{\partial s_i}{\partial p_{uqd_i}}, \tag{2.5}$$

where η is a SGD-step, and p_{uqd_i} is one of $e_{q,r}$, $a_{q,d}$, $a_{u,d}$, depending on the phase.

We refer to our reranking model as specified in this section as *personalized ranked attractiveness* (PRA).

2.4 Experiments

In this section, we compare PRA with state-of-the-art models for personalized reranking. For this purpose we use data from the Yandex Personalized Web Search challenge [133]. We begin by noting that this dataset is the only publicly available dataset that satisfies our experimental needs. It contains information about SERPs and historical interaction with all documents shown to users: documents with their ranks and clicks on them. It also provides information on which user issued the query and interacted with the SERP.

The Yandex Personalized Web Search challenge dataset is fully anonymized. There are only numeric IDs of users, queries, query terms, sessions, URLs and their domains. The dataset comes with a full description of the SERPs contained in it:

(a) the query for which the SERP was generated;

(b) the ID of the user who issued the query;

(c) URLs with their ranks and domains; and

(d) the user's interaction with documents on the SERP, that is, indicators of clicks on documents. In case of a click, the dwell time in time units is also included. The organizers of the challenge suggest that documents with a click and dwell times not shorter than 400 time units are highly relevant to the query [133].

The following preprocessing was performed on the dataset before release:

(a) queries and users are sampled from only one region (a large city);

(b) sessions containing queries with a commercial intent as detected with a proprietary classifier are removed;

(c) sessions with top-K most popular queries are removed; the number K is not disclosed.

Some key statistics of the dataset are:

(a) number of unique queries: 21,073,569;

(b) number of unique urls: 703,484,26;

(c) number of unique users: 5,736,333;

(d) number of sessions: 34,573,630; and

(e) number of clicks in the training data: 64,693,054.

Participants in the challenge are asked to rerank documents in SERPs according to the users' personal preferences.

We infer labels of URLs using a common approach [134]:

(1) a 0 (irrelevant) grade corresponds to documents with no clicks or clicks whose dwell time is less than 400 time units;

(2) a 1 (relevant) grade corresponds to documents that are clicked with a dwell time of more than 400 time units or clicked documents that have the lowest rank from all clicked documents in the SERP. A *satisfied* click is a click with a dwell time of at least 400 time units.

We use two popular binary evaluation metrics: Precision@1 (P@1) and MAP@10.

To assess the consistency of our results, we measure the performance of our algorithms on several days. The dataset covers a period of 27 days; we use the first 20 days for training and the last 7 days (days 21–27) for testing. For each test day, we train algorithms on all days prior to the test day, and evaluate on the data collected for the test day. We do this over seven days to verify that the day of the week does not affect performance.

2.4.1 Training PRA

Each time the algorithm scans a SERP, we call this a "step." We use several hyper parameters to train PRA:

(1) We make 5 steps for tuning each of parameters $a_{u,d}$, $a_{q,d}$, $e_{q,r}$: first, the algorithm makes 5 steps for tuning $a_{u,d}$, then 5 steps for tuning $a_{q,d}$, and finally 5 steps for tuning $e_{q,r}$.

(2) We learn PRA by SGD with decreasing learning rate. In each step the learning rate is equal to the reverse square root of the number of steps $learning\ rate = 1/\sqrt{step\ number}$.

(3) At the beginning we initialize all parameters $a_{u,d}$, $a_{q,d}$, $e_{q,r}$ to zero.

2.4.2 Baselines

We consider several experimental conditions (to be described below) and several baselines. Three baselines are considered for all experimental conditions:

(1) *ranker* (ORIG)—the default order that search results were retrieved by the Yandex search engine;

(2) *point-wise feature engineering* (PFE)—the winner of the Yandex Personalized Web Search challenge. The core of PFE [112] is feature engineering; it uses three types of feature. Some of the features reflect the basic ranker that feeds into the reranking: document rank, document id, query id, and so on. Another group of features describes the users' interactions with URLs: whether the user clicked, skipped or missed a document in the current session or the whole history. The third set of features are pairwise: they describe, for each pair of URLs in the SERP, which document has a higher rank. To train the PFE approach, Song [112] considers all queries and logistic regression as a classifier of satisfied clicks.

(3) *User Browsing Model* (UBM) [26]—a click model that performs the best for prediction probability of click [39].

For some of our experimental conditions we consider additional baselines:

Table 2.1: Distribution of SERPs depending on the rank of the lowest click.

Rank of the lowest click	1	2	3	4	5	6	7	8	9	10
Percentage of SERPs	54.5	13.8	8.4	5.7	4.3	3.3	2.6	2.3	2.2	2.5

(4) *past click on document* (PCLICK [120])—if the SERP contains a document that received a satisfied click from the user, then it is placed on the first rank;

(5) *document click through rate* (DCTR)—rerank documents according to CTR for document-query pairs.

2.4.3 Experimental conditions

In the literature, multiple experimental conditions have been considered for comparing approaches to personalized reranking. We consider the following:

(1) *all queries*;

(2) *rerank examined documents only*, where we consider all queries but with a truncated list of documents: documents below the lowest click are removed before running the evaluation;

(3) *repeated document subset*: SERPs with documents that a person clicked on in the past;

(4) *poor SERPs*; and

(5) *cold start*, where we group users depending on the richness of their histories.

We now describe those conditions in more detail.

(a) *All queries.* For comparability with PFE we report results on the full set of queries in the dataset and the exact same parameters as were mentioned by Song [112].

(b) *Rerank examined documents only.* To avoid falsely penalizing algorithms if they promote documents that are relevant but were not clicked simply because the user did not observe them, we also perform our experiments using all queries but with a truncated list of documents. Specifically, all documents below the lowest click are removed before running the evaluation. It is clear that SERPs with only the first retrieved document being clicked cannot be reranked in this condition, as all other documents are excluded for this particular analysis. To understand how the potential of algorithms to change the order of documents affects relative performance, we list the ranks of the lowest click in SERPs in the dataset in Table 2.1. In particular, note that after truncation, more than a half of the SERPs cannot be changed by any reranking algorithms. At the other end of the spectrum, for 2.5% of the SERPs, reranking algorithms can yield any permutation of the URLs in the originally retrieved list of results.

Table 2.2: Description of groups in the cold start problem.

Group number	1	2	3	4	5	6	7	8	9
Number of queries issued by users in the group	0	1–2	3–5	6–8	9–11	12–15	16–21	21–32	>32

(c) *Repeated document subset.* From previous studies [109, 120], we know that users' behavior on repeated queries is particularly predictable. People often try to re-find documents, which they have read before [118]. Therefore, we consider a third experimental condition: the set of SERPs with documents that a person clicked on in the past. More precisely, in order for a SERP to be included in this set it should contain one and only one previously clicked document, where a past click on the document may have been for a different query. This subset of SERPs contains 13.8% of the total. For this condition, we use PCLICK [118] as an additional baseline.

(d) *Poor SERPs.* From [119] we know that reranking is best applied selectively. Query ambiguity is one of the indicators to inform us about changing the order of documents and a good model should not rerank subsets of documents on which the ranker works well. For most queries, the top ranked document is clicked substantially more often than any of the other documents. However, for more ambiguous queries, or queries where the ranking is particularly poor, this is not the case. To evaluate such queries, in this subset we include queries for which the top ranked document is clicked less than twice as often as the second ranked document. A total of 48% of the SERPs in the dataset satisfy this condition. We also consider an additional baseline for this experimental condition: GCTR, the global clickthrough rate as defined in Section 2.4.1.

(e) *Cold start.* Naturally, there is the cold start problem: if a user or a query are new to the system, then it becomes more difficult to produce a proper ranking. To better understand the effectiveness of PRA we also provide information on the changes of algorithms' performance depending on the richness of users' histories. We divided users into nine groups depending on the number of sessions in their history in such a way that each group has about the same number of people, i.e., each group has roughly 11% of the users; see Table 2.2. The first group are the people that are new; group 2 contains users who issued one or two queries, etc. Below, we report experimental results per group.

2.5 Results

In this section we present our experimental results. We learned all models regardless of the experimental conditions. For each of the five experimental conditions defined above (all queries, examined documents only, repeated documents, query ambiguity and cold

Table 2.3: Results for the "all queries" condition, on each test day: days 21–27.

		21	22	23	24	25	26	27
P@1	ORIG	.597	.596	.588	.596	.594	.587	.581
	PFE	.607	.603	.602	.603	.604	.595	.594
	UBM	.603	.600	.596	.600	.600	.591	.587
	PRA	**.612**	**.610**	**.604**	**.611**	**.607**	**.600**	**.597**
MAP	ORIG	.719	.718	.713	.718	.714	.712	.709
	PFE	**.726**	.723	**.723**	.723	**.724**	**.718**	**.717**
	UBM	.724	.724	.719	.724	.722	.717	.713
	PRA	**.726**	**.725**	.720	**.725**	.723	**.718**	.716

start problem), we report on the performance of our proposed approach, PRA, and of the baselines listed in Section 2.4.1.

2.5.1 All queries

Table 2.3 lists the results for the "all queries" condition. We see that the performance of PRA and UBM is comparable to that of PFE, the state-of-the-art. In terms of Precision@1 PRA always outperforms PFE and PFE outperforms UBM although the difference is not significant (t-test, p-value > 0.1). In terms of MAP, the difference between PFE, UBM and PRA is at most 0.3%, in either direction. All of these three models, PFE, UBM and PRA, significantly outperform ORIG, the production ranker (t-test, p-value < 0.01). Also, surprisingly, UBM has comparable performance with PFE (t-test, p-value > 0.1).

2.5.2 Rerank examined documents only

We turn to the second experimental condition, where models rerank only examined documents. First, as this query set excludes documents below the lowest clicked position from reranking, all algorithms achieve higher scores, as we can see by contrasting the results in Table 2.4 with those in Table 2.3. The scores for PFE and PRA in this experimental condition are higher than in the "all queries" condition, both in terms of Precision@1 and MAP. Second, PRA outperforms PFE and UBM on both metrics. The difference in terms of Precision@1 exceeds 1.5% for each day, sometimes reaching 2.3%. Also, PRA performs significantly better than PFE, the state-of-the-art, in terms of MAP (t-test, p-value < 0.01). PFE and UBM have comparable performance.

Observing the performance differences between PRA, PFE and UBM relative to Table 2.3 more carefully, we note that the performance of PRA improved more due to the filtering of unobserved results. This tells us that on the complete dataset PRA promoted more documents that were not observed by the user than PFE or UBM. Thus, while the results in Table 2.3 are conservative (assuming all documents below the lowest actual click to be not relevant), the results in Table 2.4 are optimistic (restricted to documents for which we have more reliable evaluation labels). In both cases, we find that PRA outperforms PFE and UBM. We expect that results from an online evaluation

Table 2.4: Results for the "rerank examined documents only" condition, on each test day: days 21–27.

		21	22	23	24	25	26	27
P@1	ORIG	.597	.596	.588	.596	.594	.587	.581
	PFE	.610	.606	.608	.606	.608	.598	.599
	UBM	.610	.600	.606	.613	.610	.600	.597
	PRA	**.628**	**.627**	**.620**	**.629**	**.627**	**.621**	**.623**
MAP	ORIG	.719	.718	.713	.718	.717	.712	.709
	PFE	.734	.731	.733	.731	.733	.726	.726
	UBM	.730	.730	.728	.731	.732	.726	.723
	PRA	**.741**	**.740**	**.735**	**.741**	**.740**	**.736**	**.737**

would be somewhere between these two bounds.

2.5.3 Repeated document subset

In this experimental condition we only consider SERPs that contain exactly one previously clicked document. As this segment of queries was the specific target of the method proposed by Teevan, Adar, Jones, and Potts [118], we consider the additional baseline PCLICK. Table 2.5 lists the results for this condition. PRA achieves the best overall Precision@1 scores, followed by PCLICK, PFE, UBM and ORIG. Note that the difference in performance between PRA and the other approaches is more than 1% on every single test day. Surprisingly, PCLICK significantly outperforms PFE (t-test, p-value < 0.01), even though PFE is far more complicated and includes features that reflect user interactions with documents.

Although UBM and PFE achieve a similar performance in other experimental conditions, in this condition PFE achieves better results than UBM. This is a consequence of the fact that PFE is personalized and uses the whole history of a user to predict clicks. As expected, all approaches achieve better Precision@1 scores than ORIG.

Interestingly, the results for MAP show a different pattern. PCLICK and PRA work almost equally well: the difference between them is less than 0.3% and not statistically significant (t-test, p-value > 0.01). Both PCLICK and PRA perform significantly better than PFE (t-test, p-value < 0.01), which is better UBM, which, in turn, significantly outperforms *ORIG*.

2.5.4 Poor SERPs

Here we present results on ambiguous queries or queries where the ranking is particularly poor with the additional baseline DCTR; see Section 2.4.1 for a more precise definition. Table 2.6 shows the results on this subset for Precision@1 and MAP. For both metrics PRA outperforms other approaches, followed by PFE, UBM, DCTR, and then ORIG. The difference between PRA and the other approaches is significant (t-test, p-value < 0.01). ORIG performs significantly worse than the other approaches, while for most test days the differences between PFE, UBM and DCTR are not significant.

Table 2.5: Results for the "repeated document subset" condition on each test day: days 21–27.

		21	22	23	24	25	26	27
P@1	ORIG	.776	.776	.776	.773	.772	.755	.754
	PFE	.819	.801	.825	.800	.817	.782	.805
	UBM	.798	.800	.797	.796	.794	.778	.777
	PCLICK	.839	.838	.838	.836	.830	.817	.815
	PRA	**.851**	**.849**	**.848**	**.848**	**.842**	**.830**	**.831**
MAP	ORIG	.850	.850	.850	.849	.848	.836	.835
	PFE	.880	.868	.883	.866	.878	.855	.870
	UBM	.866	.867	.866	.865	.864	.853	.853
	PCLICK	**.894**	**.893**	**.893**	**.892**	**.888**	**.879**	**.880**
	PRA	.893	.891	.891	.891	.886	.877	**.880**

Table 2.6: Results for the "poor SERPs" condition on each test day: days 21–27.

		21	22	23	24	25	26	27
P@1	ORIG	.420	.415	.415	.415	.425	.424	.424
	PFE	.440	.434	.444	.433	.440	.442	.443
	UBM	.440	.435	.437	.437	.440	.444	.443
	DCTR	.450	.448	.433	.450	.446	.441	.443
	PRA	**.460**	**.458**	**.458**	**.458**	**.457**	**.456**	**.458**
MAP	ORIG	.617	.614	.611	.614	.618	.617	.618
	PFE	.628	.625	**.630**	.624	.623	.628	.630
	UBM	.627	.627	.623	.628	.624	.630	.630
	DCTR	.628	.626	.610	.627	.621	.617	.620
	PRA	**.635**	**.633**	**.630**	**.634**	**.632**	**.629**	**.633**

Also, all algorithms work much better on the subset where the condition of *Poor SERPs* is not satisfied. The performances of ORIG, PFE, UBM and PRA are similar and the precision@1 scores are over 78%. To conclude, the PFE, UBM and PRA methods improve ambiguous queries, but do not affect non-ambiguous ones.

2.5.5 Cold start problem

In the "cold start problem" condition we provide information on the algorithms' quality depending on the richness of users' history. This experiment has several results; see Table 2.7 for the results for both Precision@1 and MAP.

First, despite the fact that ORIG is not personalized it performs better for users with a long history. One of the explanations of this is that people who use the search engine a lot learn to submit high quality queries [85]. Second, the personalized models PFE and PRA benefit more from a user's history than ORIG and UBM. For users who issued more than 32 queries the difference between ORIG and these model is more than 2%

Table 2.7: Performance of algorithms depending on the number of queries issued by user.

		0	1–2	3–5	6–8	9–11	12–15	16–21	21–32	>32
P@1	ORIG	.584	.570	.572	.581	.585	.595	.605	.613	.652
	PFE	**.594**	**.579**	.581	.592	.597	.608	.620	.631	.684
	UBM	.593	.578	.581	.590	.595	.605	.617	.627	.673
	PRA	.588	.577	**.582**	**.594**	**.600**	**.613**	**.626**	**.640**	**.694**
MAP	ORIG	.710	.700	.700	.707	.710	.718	.726	.733	.762
	PFE	.714	.702	.704	.712	.717	.725	.734	.744	.783
	UBM	**.715**	**.705**	**.706**	**.714**	.717	.725	.734	.743	.777
	PRA	.710	.700	.703	.713	**.718**	**.727**	**.737**	**.749**	**.788**

for both metrics. Also, for users with a limited history, PFE and UBM benefits more than other algorithms. However, for users with a rich history UBM performs worse than PFE, which in turn performs worse than PRA, but still much better than ORIG.

2.6 Conclusion

As search engines often show ten documents as a result page, most users can find a relevant item among them. However, different users have different interests. Thus for some users the first document may be relevant, but for others not. This is why we study the problem of reranking documents according to user interest. We have proposed a new simple method for personalized search based on long-term behavioral signals that matches or outperforms the state-of-the-art for this task.

We note that current state-of-the-art solutions are effective, however they require extensive feature engineering. The most effective approaches have more than one hundred features. The second approach for this problem is manually creating rules, which is bound to work on a small segment of queries only. Another approach is click models. Click models are a very elegant solution for this problem, but in several experimental conditions work significantly worse than the state of the art. In contrast, our algorithm is applicable to all result sets, does not require feature engineering, but has comparable performance in all experimental conditions. We achieve this performance by incorporating click models with learning to rank algorithms.

We compared our proposed method with the state-of-the-art and with manually defined rules using a publicly available data set. We considered multiple experimental conditions. In all conditions we perform as least as well as the state-of-the-art and in several conditions we significantly outperform it according to both metrics used, despite the simplicity of our method.

Finally, we observe that our proposed approach only covers queries that have been seen previously, in the training data. In the future we plan to extend our approach to previously unseen queries by incorporating query similarity in our model. Also, we plan to incorporate different relevance signals from query results and behavioral facets (visited pages, eye movement, etc.)

In this chapter we answered RQ1 and proposed a new algorithm that infers a user's long-term preferences, using only their interaction with the system. In the next chapter, we take a different angle on the problem of understanding a user's preference and study how we can create an informative preference questionnaire to infer a cold user's preferences.

3 Preference Elicitation as an Optimization Problem

In this chapter we address RQ2 and propose a method to solve the user cold-start problem. The new user cold start problem arises when a recommender system does not yet have any information about a user. A common solution to it is to generate a profile by asking the user to rate a number of items. We propose a new elicitation method to generate a static preference questionnaire that poses relative preference questions to the user and answer RQ2: how to create an informative preference questionnaire to infer cold users' preferences.

3.1 Introduction

Millions of online e-commerce websites are built around personalized recommender systems (e.g., [27]). Together, they offer broad and varied items, ranging from books and accommodation to movies and dates. The main goal of recommender systems is to predict unknown ratings or preferences based on historical user interactions and information about items, such as past user ratings and item descriptions. Collaborative filtering (CF) is one of the most effective techniques to build personalized recommender systems. Approaches based on (CF) collect user item ratings and derive preference patterns. The main advantage of approaches based on CF is that they do not require domain knowledge and can easily be adapted to different recommender settings.

However, CF-based approaches only work well for users with a substantial amount of information about their preferences. When this information is not available, as for new users, the recommender system runs into the *new user cold-start problem*: it cannot produce reliable personalized recommendations until the "cold" user is "warmed-up" with enough information [36, 48].

Most well-known and effective approaches to the new user cold-start problem are questionnaire-based [48, 95]. These elicit information from new users via a questionnaire where users provide absolute ratings for some items [72, 33, 89]. However, obtaining such explicit ratings suffers from a calibration issue [56]: the same rating, of say three stars, may mean completely different things for different people. Moreover,

This chapter was published as [105].

users may change their opinion about an item after having seen other items. For instance, it has been shown that a user is likely to give a lower rating to an item if the preceding one deserved a very high rating [1, 61]. On top of that, recommender systems typically provide ranked lists of items, but for producing a good ranking pairwise preferences between items are preferable to learning absolute relevance scores [11].

Finally, we note that recent research has shifted towards predicting relative preferences rather than absolute ones [57, 107]. For such relative methods, it is natural to ask users to complete a questionnaire by answering relative questions rather than by providing absolute ratings. Therefore, *preference elicitation* questionnaires have been proposed [18, 95]. A preference elicitation method (PEM) thus asks questions about pairwise preferences between items.

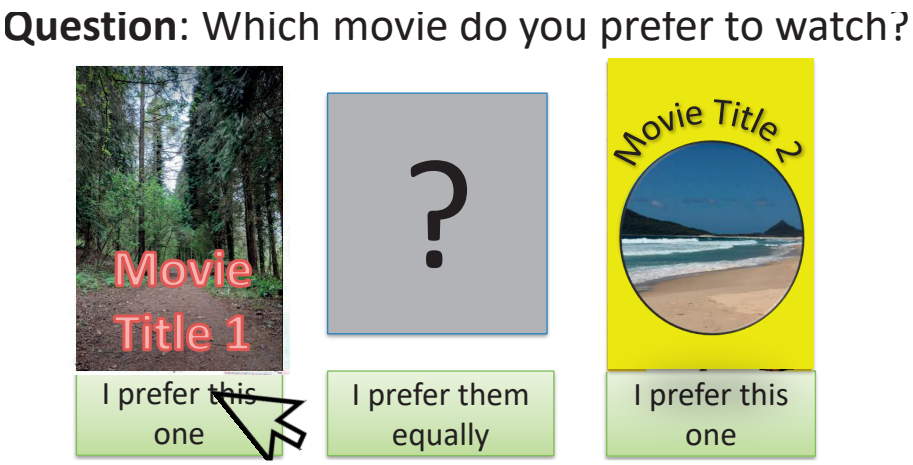


Figure 3.1: An example of a relative question in a questionnaire aimed at eliciting preferences.

An example preference elicitation question is shown in Figure 3.1 below, where a user is asked to indicate which of two movies she prefers. However, it is difficult to create these questionnaires, because it is unclear what criteria should be used to select items for questions. Moreover, a greedy approach, which is common in elicitation procedures, is computationally expensive in PEM, because the cardinality of the preference question space is proportional to the square of the number of items.

In this chapter, we address some of these challenges with the following **main research question**:

> *How to optimally generate a preference questionnaire, consisting of relative questions, for new users that will help to solve the new user cold-start problem?*

To answer this question, we develop a new preference elicitation method (PEM), called static preference questionnaire (SPQ), that can be used in several domains, such as book or movie recommendation. In particular, inspired by [2], we formulate the elicitation procedure in SPQ as an optimization problem. The advantage of static *preference* questionnaires, like the approaches of [18, 95], is that they work with relative rather than absolute questions. The main advantages of our SPQ method over previous work

is the absence of any constraint on the questions except that they should be informative, as was done in [18], and the fact that it optimizes the expectation of the loss function of the underlying LFM.

Thus, the proposed method SPQ is a preference elicitation method (PEM) that solves the following optimization problem: Select items for relative questions to predict with maximum accuracy how the user will respond to other relative questions.

3.2 Problem formulation

3.2.1 Assumptions

Suppose we have a system that contains a history of user feedback for items in a rating matrix $R \in \mathbb{R}^{n \times m}$, where n is the number of users and m is the number of items.[1] The value of entry r_{ui} describes the feedback of user u on item i. The feedback can be (a) explicit, e.g., a user rating (usually from 1 to 5) or a binary value, or (b) implicit feedback like purchases or clicks. Usually, R is sparse [83], since most users rate only a small portion of the items. The task for a recommender system is to predict missing ratings in R and recommend to a user the most attractive item.

An important aspect of our work, as well as previous work [2, 18], is the use of a latent factor model (LFM) [83] to solve this task. We assume that LFM is already trained and that each rating r_{ui} complies with the following noisy model:

$$r_{ui} = \mu + b_i + b_u + v_i^T v_u + \epsilon_{ui}, \tag{3.1}$$

where

(1) μ is a global bias,

(2) v_i, b_i are the latent factor vector and bias of item i,

(3) v_u, b_u are the latent factor vector and bias of user u, and

(4) ϵ_{ui} is small random noise.

The dimensions in the latent vectors represent item's characteristics. If the user likes a particular item's attributes, the corresponding dimensions in her latent factor vector are large. Large values in the item's latent vector correspond to the item's most valuable characteristics. Thus, if an item satisfies a user's taste, the value of $v_i^T v_u$ in Equation 3.1 is large. The *user bias* is the rating that a user predominantly provides. The intuition behind the item bias is its popularity. The LFM predicts that most users will like popular items more, despite their diverse preferences. The parameters listed above are assumed to be given to us; we refer to them further as *ground truth*.

The **main problem** that we address is to solve the new user cold-start problem, having ground truth parameters of all items and of all users who have provided some feedback already.

[1] The notation we use is presented in Table 3.1.

Table 3.1: Notation used in the chapter.

Symbol	Gloss
$\mathbb{I}$	Set of all items
$\mathbb{P}$	Set of all pairs of items: $\mathbb{I} \times \mathbb{I}$
$\mathbb{B}$	Seed set of questions
$\mathbb{F}$	Subset of questions from $\mathbb{B}$
i, j	Items
u	User
q	Question
$\mathbb{V}_{\mathbb{S}}$	Latent presentation of questions from set $\mathbb{S}$
v_i	Latent factor vector of item i
$v_{(i,j)}$	Latent presentation of the question
v_u^*	True latent factor vector of user u
$\hat{v}_u$	Predicted latent factor vector of user u
R	Matrix of ratings provided by users
$\hat{r}_{uij}$	Predicted preference of user u of item i over item j
r_{uij}^*	True preference of user u of item i over item j

3.2.2 Problem definition

To address the new user cold-start problem, we propose a preference elicitation method (PEM) that produces a questionnaire with relative questions; an example such question is in Figure 3.1. We call our method static preference questionnaire (SPQ). The main goal of SPQ is to minimize the number of questions N that is required to derive a ranked list of personalized recommendations. SPQ minimizes N by generating a list of questions that maximize the gain that a recommender system observes after adding answers for q.

The LFM uses SPQ as follows:

(1) using the completed questionnaire, LFM gets $\hat{v}_u$ while predicting the ground truth parameters v_u^*; then

(2) LFM uses $\hat{v}_u$ to predict users' relative preferences about all pairs of the items.

Thus, the performance of SPQ is measured in terms of the binary classification quality of item pairs. For a given user u and a pair of items (i, j) the classifier estimates a user u's preference of item i over item j. The quality of the classifier is defined as the deviation of the predicted preference value $\hat{r}_{uij}$ from the true preference value r_{uij}^*. We assume that r_{uij}^* can take the values -1, 1, and 0: -1 and 1 reflect that the user prefers one item over another, while 0 corresponds to the situation when she treats them equally (again, for an example, see Figure 3.1). Similar to Anava, Golan, Golbandi, Karnin, Lempel, Rokhlenko, and Somekh [2], we define the preference elicitation task as an optimization problem. Given a budget N, SPQ chooses a seed set $\mathbb{B} \subseteq \mathbb{P}$, consisting of

N pairs of items, that minimizes the expected loss of the corresponding binary classifier:

$$\arg\min_{\mathbb{B}} \mathbb{E}\left(\sum_{(i,j)\in\mathbb{P}} (\hat{r}_{uij} - r^*_{uij})^2\right). \tag{3.2}$$

The intuition behind this optimization problem is to force a model to provide a high-quality list of recommendations, asking a predefined number of questions. To provide a high-quality ranked list, a model should correctly classify pairwise preferences between items [11]. That it is exactly what Equation 3.2 expresses. In Section 3.5, we will prove how minimizing Equation 3.2 corresponds to improving the quality of the final list of recommendations.

To summarize, we have introduced assumptions for our model and formulated the task of finding a seed set of relative questions as an optimization problem. Next, we will describe an iterative greedy procedure to obtain a near-optimal solution for this problem.

3.3 Method

Equation 3.2 is difficult to optimize directly, so we will reformulate it in Section 3.3.2. To this end, we need two preliminary procedures:

(1) one that predicts users' answers to preference questions, and

(2) one for modeling a user's latent vector representation after receiving their completed questionnaire.

We provide these next.

3.3.1 Preliminaries

Predicting users' answers

We aim to estimate a user u's preference for item i over item j, given her predicted latent representation $\hat{v}_u$, using the equation:

$$\hat{r}_{uij} = \hat{v}_u \cdot v_{(i,j)} + b_{(i,j)}, \tag{3.3}$$

where $v_{(i,j)}$ is the latent representation of the question, $v_i - v_j$, and $b_{(i,j)} = b_i - b_j$ is a bias of the question. We rewrite this in matrix form. The predicted answers to a set of preference questions $\mathbb{S}$ are:

$$\hat{r}_{u\mathbb{S}} = \hat{v}_u \cdot \mathbb{V}_{\mathbb{S}} + b_{\mathbb{S}}, \tag{3.4}$$

where the columns of matrix $\mathbb{V}_{\mathbb{S}}$ are the latent vector representations of the questions from $\mathbb{S}$ and $b_{\mathbb{S}}$ is a column that consists of the biases of the questions.

Modeling users' latent vector presentations

To model the user parameters after receiving a completed questionnaire, which consists of a seed set $\mathbb{B}$, we solve the following problem:

$$\arg\min_{v_u} \sum_{(i,j)\in\mathbb{B}} (v_u \cdot v_{(i,j)} + b_{(i,j)} - r^*_{uij})^2, \tag{3.5}$$

where r^*_{uij} is an answer given by the user while completing the questionnaire. This problem can be solved using linear regression.

3.3.2 Static Preference Questionnaire (SPQ)

Anava, Golan, Golbandi, Karnin, Lempel, Rokhlenko, and Somekh [2] show that if the procedure for predicting a user's answers and modeling their latent presentation is exactly like we describe above in Equation 3.4 and Equation 3.5, then the optimization problem Equation 3.2 can be formulated as a simpler yet equivalent task:

$$\arg\min_{\mathbb{B}} \operatorname{tr}[(\mathbb{V}_{\mathbb{B}}\mathbb{V}_{\mathbb{B}}^T + \epsilon \cdot \mathbb{E})^{-1}]. \tag{3.6}$$

Here, $\operatorname{tr}(M)$ denotes the trace of matrix M and $\mathbb{E}$ is the identity matrix. We propose a greedy iterative algorithm that chooses the next question by minimizing Equation 3.6 in each iteration. More precisely, given the subset $\mathbb{F}$ of $\mathbb{B}$ consisting of questions that have already been chosen for our questionnaire, the procedure selects the next question q that minimizes the following:

$$\operatorname{tr}[(\mathbb{V}_{\mathbb{F}\cup\{q\}}\mathbb{V}_{\mathbb{F}\cup\{q\}}^T + \epsilon * \mathbb{E})^{-1}]. \tag{3.7}$$

The proposed optimization problem can be reduced to a simpler one, which is stated and proven in Theorem 1 below.

Before presenting Theorem 1, we need to introduce some additional notation used in Theorem 1 and our algorithm. Define $\mathbb{A} = (\mathbb{V}_{\mathbb{F}}{\mathbb{V}_{\mathbb{F}}}^T + \epsilon * \mathbb{E})^{-1}$. Also, $e_1, e_2, \ldots, e_n$ are orthonormal eigenvectors of $\mathbb{A}$ with eigenvalues: $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_i > \epsilon^{-1}$. We assume that the number of questions in the questionnaire is less than the dimension of the latent space; as a consequence, the subspace of eigenvectors with eigenvalues ϵ^{-1} is non-empty; we denote it as E_ϵ.

Theorem 1. *Let $\epsilon > 0$ be arbitrarily small. The optimization problem* (3.7) *is equivalent to finding a question q with latent representation $v_q = \sum a_i e_i + e_\epsilon$ that minimizes*

$$\frac{\lambda_1 a_1^2 + \cdots + \lambda_n a_n^2 + 1}{\mid e_\epsilon \mid^2}, \tag{3.8}$$

where $e_\epsilon \in E_\epsilon$ and a_i is a coordinate of v in the basis $\langle e_1, \cdots, e_n, e_\epsilon\rangle$.

Proof. To begin, note that $\mathbb{V}_{\mathbb{F}\cup\{q\}}\mathbb{V}_{\mathbb{F}\cup\{q\}}^T + \epsilon * \mathbb{E} =$

$$\mathbb{V}_{\mathbb{F}}\mathbb{V}_{\mathbb{F}}^T + v_q * v_q^T + \epsilon * \mathbb{E} = \mathbb{A}^{-1} + v_q * v_q^T. \tag{3.9}$$

Thus minimizing (3.7) is equivalent to maximizing the following:

$$\begin{aligned}
&\max\left(\frac{\sum_{i=1}^{n}\lambda_i^2 a_i^2+\epsilon^{-2}\mid e_\epsilon\mid^2}{1+\sum_{i=1}^{n}\lambda_i a_i^2+\epsilon^{-1}\mid e_\epsilon\mid^2}\right)\\
&\sim \max\left(\epsilon^{-1}+\frac{\sum_{i=1}^{n}\lambda_i(\lambda_i-\epsilon^{-1})a_i^2-\epsilon^{-1}}{1+\sum_{i=1}^{n}\lambda_i a_i^2+\epsilon^{-1}\mid e_\epsilon\mid^2}\right)\\
&\sim \max\left(\epsilon^{-1}+\frac{\sum_{i=1}^{n}\lambda_i(\lambda_i\cdot\epsilon-1)a_i^2-1}{\epsilon(1+\sum_{i=1}^{n}\lambda_i a_i^2)+\mid e_\epsilon\mid^2}\right)\\
&\sim \max\left(\frac{\epsilon\cdot(\sum_{i=1}^{n}\lambda_i^2 a_i^2)-(\sum_{i=1}^{n}\lambda_i a_i^2+1)}{\epsilon(1+\sum_{i=1}^{n}\lambda_i a_i^2)+\mid e_\epsilon\mid^2}\right)\\
&\sim^{\epsilon\to 0}\max\left(\frac{-(\sum_{i=1}^{n}\lambda_i a_i^2+1)}{\mid e_\epsilon\mid^2}\right)\sim\min\left(\frac{(\sum_{i=1}^{n}\lambda_i a_i^2+1)}{\mid e_\epsilon\mid^2}\right).
\end{aligned}$$ □

To understand the intuition behind optimizing Equation 3.8, we need to understand the connection between the eigenvectors of $\mathbb{A}$ and the questions in $\mathbb{F}$, which we address next.

Lemma 1. *If the questions in $\mathbb{F}$ are represented by linearly independent vectors $v_{(i_1,j_1)},\ldots,v_{(i_m,j_m)}$, then $e_1,\ e_2,\ \ldots,e_n$ are vectors from a subspace $U=\langle v_{(i_1,j_1)},\ldots v_{(i_m,j_m)}\rangle$. Also, the number of eigenvalues that are different from ϵ is n. That is, $n=m$.*

Proof. Eigenvectors are the same for $\mathbb{A}$ and $\mathbb{A}^{-1}$, so we prove this lemma for $\mathbb{A}^{-1}$. To prove the first part of the lemma it is enough to show that U and $U^{\perp}$ are invariant subspaces for $\mathbb{A}^{-1}$ and $U^{\perp}\subseteq E_\epsilon$, where $U^{\perp}$ is the subspace that consists of vectors orthogonal to U. Note that $U^{\perp}$ is a subspace of E_ϵ. Indeed, if $v\in U^{\perp}$, then

$$\begin{aligned}
\mathbb{A}^{-1}v &= \left(\mathbb{V}_{\mathbb{F}}{\mathbb{V}_{\mathbb{F}}}^T+\epsilon\cdot\mathbb{E}\right)v=\left(\sum_l v_{(i_l,j_l)}v_{(i_l,j_l)}^T+\epsilon\cdot\mathbb{E}\right)v\\
&= \sum_l v_{(i_l,j_l)}v_{(i_l,j_l)}^T v+\epsilon\cdot v=\sum_l\langle v_{(i_l,j_l)},v\rangle v_{(i_l,j_l)}+\epsilon\cdot v\\
&= \epsilon\cdot v. \qquad (3.10)
\end{aligned}$$

Also, U is an invariant subspace of A^{-1}. To see this, assume that $u=\sum_l x_l v_{(i_l,j_l)}$ is a vector from U. Then $A^{-1}=$

$$\left(\sum_l v_{(i_l,j_l)}v_{(i_l,j_l)}^T+\epsilon*\mathbb{E}\right)u=\sum_l v_{(i_l,j_l)}\langle v_{(i_l,j_l)},u\rangle+\epsilon u\in U. \qquad (3.11)$$

To prove the last part of the lemma, it is enough to show that $\mathbb{A}^{-1}|_U$ is non-degenerate. $\mathbb{A}^{-1}|_U: U\to U$ is an operator that changes the vectors from U in the same way as $\mathbb{A}^{-1}$: $\mathbb{A}^{-1}|_U\cdot u=\mathbb{A}^{-1}\cdot u$ for any vector $u\in U$. From Equation 3.11 it follows that $\mathbb{A}^{-1}|_U$ is a Gram matrix [102] of the vectors $v_{(i_1,j_1)},\ldots,v_{(i_n,j_n)}$. But $v_{(i_1,j_1)},\ldots,v_{(i_n,j_n)}$ are independent vectors and consequently their Gram matrix is non-degenerate. □

One implication of Lemma 1 is that E_ϵ is equal to $U^\perp$. Also, if we choose the basis of U to be $\{\sqrt{\lambda_1}e_1, \ldots, \sqrt{\lambda_n}e_n\} = \{e'_1, \ldots, e'_n\}$ and decompose the latent presentation of the new question by this basis $v_q = \sum_l a_l e'_l + e_\epsilon$, then the numerator of Equation 3.8 will simply be

$$\sum_l {a_l}^2 + 1, \tag{3.12}$$

which is the length of the part of v_q that is parallel to U in this basis, increased by one.

To understand the intuition behind the denominator of Equation 3.8 we need another piece of notation: we denote the projection of vector v on E_ϵ as v_ϵ. Suppose question q is about the preference of i over j, then the denominator of Equation 3.8 equals

$$\mid v_{q,\epsilon} \mid^2 = \mid (v_i - v_j)_\epsilon \mid^2 = \mid v_{i,\epsilon} - v_{j,\epsilon} \mid^2 . \tag{3.13}$$

That means that the denominator of Equation 3.8 is a Euclidean distance between the projection of v_i and v_j or a length of $v_{q,\epsilon}$. Consequently, in order to add one more question to $\mathbb{F}$, we should find a vector with a large component from $U^\perp$ and with a small component from U. Thus, if the questions from $\mathbb{F}$ are linearly independent, then the question that will be added to $\mathbb{F}$ should be linearly independent of all previous questions, and the extended questionnaire will consist of linearly independent questions.

Now we are ready to present Algorithm 1, SPQ, for generating preference questionnaires.

Algorithm 1 Static Preference Questionnaire (SPQ).

1: **Input:** $\{e'_1, \ldots, e'_n\}$ basis of U; $\mathbb{E}$ the set of all items and $r > 0$
2: **for** $i \in I$ **do**
3: $\quad v_i = v'_i + v_{i,\epsilon}$, where $v'_i \in U$, $v_{i,\epsilon} \in U^\perp$
4: $\mathbb{S} = \{i \in \mathbb{E} :\mid v'_i \mid < r\}$
5: $\mathbb{S}_\perp = \{v_{i,\epsilon}$ for $i \in \mathbb{S}\}$
6: Find the farthest points $(i, j) \in \mathbb{S}_\perp$ by Euclidean distance.
7: **return** $v_{(i,j)}$

To minimize the numerator of Equation 3.8, we pick a small number r ($r = 1$ in our experiments) and choose the set of items $U_{<r}$ on which Equation 3.12 is smaller than it. To maximize the denominator of Equation 3.8, the algorithm chooses among $U_{<r}$ the question q with $\max \mid v_{q,\epsilon} \mid^2$. As pointed out above, this problem is equivalent to finding the diameter of the projections of points from $U_{<r}$ to E_ϵ. The diameter finding problem does not have an exact solution in linear time [32], thus we use a well-known approximation to find it; see Algorithm 2 [29].

3.4 Experimental Setup

3.4.1 Research questions

We assume that a good questionnaire consists of a small number of questions in order not to bother the user too much. At the same time, the information received after

Algorithm 2 Computing the diameter of a set of points [29].

1: **Input:** The set of points $\mathbb{S}$
2: Choose a random point $p \in \mathbb{S}$
3: Find the farthest point from p: p_1 by Euclidean distance
4: Find the farthest point from p_1: p_2 by Euclidean distance
5: **return** p_1, p_2

completing the questionnaire should be enough to provide good recommendations and to understand the user's preferences. Thus, our research questions are the following:

RQ2.1 The final goal of SPQ is to provide enough information to build a high-quality list of personalized recommendations for a new "cold" user. Does minimizing Equation 3.2 correspond to a better quality of the recommendations?

RQ2.2 What is the impact of the preference elicitation procedure used on the final quality of recommendations?

3.4.2 Experimental methodology

Datasets

We conduct experiments on the MovieLens dataset, with explicit ratings, and an Amazon book dataset, with implicit feedback. As a ground-truth answer r^*_{uij} to the question of the preference between two elements i and j, that have received different ratings $r_{ui} \neq r_{uj}$, we use 1 if $r_{ui} < r_{uj}$ and -1 otherwise. We processed the datasets subject to the following constraints:

(1) in the dataset there is a sufficient amount of information in order for LFM to be reliable;

(2) personalization really improves the quality of the recommendation of the items in the data set; and

(3) users in the dataset tend to evaluate items they like and items they do not like.

MovieLens. The original MovieLens dataset [42] contains 27,000 movies, 20 million ratings, that are scored by 138,000 users. These 5-star ratings were collected between January 9, 1995 and March 31, 2015. The dataset is processed as follows. First, following [72] we are interested in the ability of algorithms to distinguish good recommendations from bad ones. Therefore, we binarize the ratings as in [72]: ratings that are scored by 4 stars and above become positive, others become negative. Then, for the constructed dataset to satisfy the first constraint mentioned above, we only retain movies that have been rated by at least five users. In addition, for the created dataset to satisfy the second constraint, we keep only movies for which the ratio between the number of negative ratings and the positive ratings received by these movies is less than three. Finally, we keep only users for whom the ratio between the number of negative

and positive ratings given by each of these users is less than three. Thereby the dataset satisfies the third constraint.

The final dataset consists of 10 million ratings, produced by 89,169 users, for 11,785 movies.

Amazon books. This dataset contains ratings for books as part of users' reviews [81]. One drawback of collecting ratings this way is that users rate and write reviews only for books that they bought and read. So, the descriptions and genres of the negatively rated books can potentially satisfy user's preferences. Therefore, we create a new dataset using the reviews from Amazon. In the constructed dataset a rating is positive if a user wrote a review about the book, and consequently, bought and read it.

Negative ratings are created by uniformly sampling from books that the user did not read. For each user, the number of positive ratings equals the number of negative ratings. In order for the data set to satisfy the first constraint (that LFM is reliable), we only include the 30,000 most popular books. We only keep users who rated from 20 and 1,000 books. This leaves us with a dataset with 3,200,000 ratings, from 33,000 users for 30,000 books; all books included in the dataset received at least 5 reviews. The dataset satisfies the second and third constraint because users and items have diverse ratings as the data was obtained by negative sampling.

Baselines

We are the first to propose a preference elicitation method that creates a *static* questionnaire containing relative questions about user preferences. To the best of our knowledge, there are no direct baselines to compare with. Therefore, we compare SPQ with three baselines designed for solving a similar but not identical task. However, they all optimize the Presicion @k measure, which we also use to compare the methods. Moreover we use them in the setting for which they were designed. As baselines we use two state-of-the-art methods that create a preference questionnaire, and one method that creates an absolute questionnaire:

- Bandits [18] and Pair-Wise Decision Trees (PWDT) [95] that create a dynamic questionnaire by asking relative questions;
- Forward Greedy [2] that creates a static questionnaire but contains absolute questions.

In addition, we compare SPQ with a random baseline that selects relative questions randomly.

Importantly, the main goal of this chapter is to create a *preference questionnaire* in an optimal way, thereby avoiding problems from which absolute questionnaires suffer. Thus, the most appropriate baselines for SPQ are methods that create preference questionnaires.

Bandits. Christakopoulou, Radlinski, and Hofmann [18] present a bandit-based method for creating a dynamic questionnaire by asking relative questions in an online setting. They propose the following algorithm for picking two items for asking relative questions:

(1) select the first item that the recommender algorithm predicts to be the best for the user in the current stage;

(2) virtual observation and update: assume that the user does not like the first item and virtually update the model including this knowledge;

(3) select a comparative item that the algorithm, using the virtually updated model, predicted to be the best for the user in the current stage.

PWDT. Rokach and Kisilevich [95] propose the PWDT algorithm, which solves the same problem as Bandits, i.e., it creates a dynamic questionnaire asking relative questions. PWDT minimizes the weighted generalized variance in each step. To this end, the algorithm needs to brute force all pairs of items. To avoid this, the authors suggest to first cluster items, using the k-means algorithm with cosine similarity. Then they create artificial items, which are the centroids of these clusters, and use only these artificial items to choose the next question. The PWDT algorithm is dynamic, which means that it must perform every step fast; it remembers all answers and questions that it has received and asked earlier, using a lazy decision tree [34].

Forward Greedy. Anava, Golan, Golbandi, Karnin, Lempel, Rokhlenko, and Somekh [2] propose a Forward Greedy (FG) approach that creates a static questionnaire consisting of absolute questions. Users are asked to rate one item per question. FG has the same assumptions as SPQ (listed in Section 3.2.1) and it also solves optimization problem which similar to our formulation in (3.2). Anava, Golan, Golbandi, Karnin, Lempel, Rokhlenko, and Somekh [2] prove an upper bound on the error of the Backward Greedy (BG) algorithm and show that FG and BG achieve similar results on realistic datasets. Therefore, we only use FG for comparison. Our goal is to provide a method for creating a static *preference questionnaire* in an optimal way, thereby avoiding problems from which absolute questionnaires suffer. However, it is useful to compare the proposed method with a reasonable absolute baseline to understand if the quality of the preference questionnaire is comparable to the quality of absolute questionnaires in terms of the final list of recommendations.

Random. Finally, we compare SPQ with a baseline that randomly picks pairs of items for questions. We include this comparison to better understand the importance of carefully selecting questions.

Training latent factor models (LFMs)

We stress that training LFM is not the task of this chapter; instead, we just use its parameters: μ, v_i, b_i, v_u, b_u (presented in Equation 3.1) for 1. the ground truth and 2. our recommender system. We factorize users' ratings matrix R using LFM (Equation 3.1), which is trained to minimize the Root Mean Squared Error (RMSE) using Stochastic Gradient Descent (SGD). We first randomly pick one thousand users as "cold start users". Then we train LFM on all ratings by all other users, who are "warm" users. To choose a combination of hyper-parameters, namely the latent dimension and regularization, we create a validation set that consists of ten random ratings for each "warm" user who has rated at least twenty items. We set the hyper-parameters to the values that achieve the best accuracy on the validation set.

In order to achieve statistically meaningful results, we perform this procedure ten times. I.e., the following is repeated ten times:

(1) choose the subset of cold-start users randomly;

(2) train LFM using the ratings obtained from all other users;

(3) create a questionnaire; and

(4) measure its quality.

3.4.3 Experimental conditions

We randomly divide all items into two equal subsets:

(1) the training items that are used to select questions;

(2) the test items that are used to measure performance.

For all experimental conditions we use a standard procedure to measure the performance [33], as described in Algorithm 3:

(1) we receive a completed questionnaire for each user;

(2) we receive predictions of the users' latent presentation using Equation 3.3 and ground truth presentation of the question ($v_{(i,j)}$ and $b_{(i,j)}$); and

(3) we use this presentation to predict the user's ratings on the chosen subset of the test items.

Algorithm 3 Evaluation of an elicitation procedure.

Input: Set of items $\mathbb{I}$, the set of cold users $\mathbb{U}_{cold}$
$Loss = 0$
divide $\mathbb{I}$ into two subsets of equal size randomly: $\mathbb{I}_{test}$, $\mathbb{I}_{train}$
for each $u \in \mathbb{U}_{cold}$ **do**
    create a questionnaire, consisting of questions about items from $\mathbb{I}_{train}$
    receive u's completed questionnaire
    predict v_u: $\hat{v}_u$
    choose test questions about items from $\mathbb{I}_{test}$
    using $\hat{v}_u$ predict answers on test questions: $Answers_u$
    increase the loss $Loss \rightarrow Loss + Loss(Answers_u)$
return $Loss$

We have two experimental conditions, *real* (Section 3.4.3) and *simulated* (Section 3.4.3), which differ in how they receive users' answers and choose test questions.

Real condition

As described above, we first split all items into two equal-sized subsets randomly:

(1) training items that are used to select questions; and

(2) test items that are used to measure the quality.

In this condition, for each user, we create questions only about items that she has rated. We also measure the quality of the elicitation procedure using the subset of the test items that the user has rated. Consequently, we can infer users' answers using the ratings that are available in the dataset. Thus, in this setup, we do not simulate answers and do not need any assumptions to receive users' answers. However, optimal items may not be available. Also, despite the fact that we use the same elicitation procedure for all users, the questionnaires will be different for different users because they rate different subset of items. To avoid this problem and ask all users the same questions, we also perform a simulated experiment.

Simulated condition

In order to ask all users the same questions and to choose the questions in an optimal way we require the mechanism for obtaining answers to the questions. However, we cannot obtain this information directly from ratings in the datasets that we use, because users rate different subsets of items and only a small fraction of all items. Thus, following [18], in the *simulated condition* we simulate users' answers using Equation 3.3. This way of obtaining users' answers is motivated by the following assumption: all ratings comply with the LFM model that is given to us. We use this simulation procedure to get answers for both sets of questions from (1) the questionnaire; and (2) the subset of test questions. It is important to point out that, like SPQ, Bandits, PWDT, and FG also rely on the assumptions about pretrained LFM described in Section 3.2.1. Thus, using the simulation condition does not favor any method, baseline or otherwise.

Algorithm 4 provides detailed information on how to choose a subset from the test items to generate the test questions. First, for a given user we divide all test items into 50 groups according to her preference. Then we pick one item per group. This procedure picks only 50 items instead of thousands of test items; moreover, the chosen 50 items are very diverse.

Algorithm 4 Choosing a subset from the test items in the Simulated condition.

1: **Input:** Set of test items $\mathbb{I}_{test}$, user u's ground truth vector v_u
2: initialize subset from the test items $\mathbb{I}_{testSubset} = \emptyset$
3: using (3.1) receive all ratings by u to $\mathbb{I}_{test}$
4: get the extended recommendation list $L = \{i_1, \ldots, i_n\}$ by sorting items according to their ratings
5: divide items into 50 groups $S_1, \ldots, S_{50}$:
$S_k = \{i_{(k-1)*(n/50)+1}, \ldots, i_{(k-1)*(n/50)+n/50}\}$
6: **for each** group S in $S_1, \ldots, S_{50}$ **do**
7: choose the random item $i \in S$
8: add i to $\mathbb{I}_{testSubset}$
9: **return** $\mathbb{I}_{testSubset}$

3.4.4 Evaluation methodology

To answer our research questions we use different metrics.

RQ2.1

To understand whether it is right to optimize Equation 3.2, we should provide two measures:

(1) one that reflects how well Equation 3.2 is optimized; and

(2) another one that reflects the quality of the final personalized list of recommendations.

We choose

(1) Precision of the classification of all preference questions (PCPF) between any two test items: PCPF is equal to the fraction of correctly classified pairwise preferences between items; and

(2) Precision@10 of the recommendation list (P@10).

We would like to observe if the algorithm with the lowest PCPF has the highest P@10.

RQ2.2

The goal of a recommender system is to provide to a user the most relevant recommendations. That is why we use ranking measures to evaluate all methods, specifically P@10, to compare the quality of the final personalized lists. Precision @k is the fraction of relevant items among the top-k recommendations. It is one of the most popular measure for the recommender systems. Our final evaluation measures were computed by averaging Precision@k over all users in the test set.

3.5 Results

Our experimental results are shown in Figure 3.2 and 3.3 for the MovieLens movie dataset and the Amazon book dataset, respectively.

Importantly, the maximum Precision@10 score depends on the dataset and LFM: for some users, there are fewer than 10 positive ratings. We chose the range of the plots for Precision@10 measure from 1. its minimum value for the non-personalized recommendation; 2. until its maximum value, the value achieved by calculating the Precision@10 for the ideal ranking in which items that a user likes are all on the top. The maximum value for Precision@10 in the real condition for the MovieLens and Amazon Books datasets are 0.71 and 0.90, respectively. And the maximum value for Precision@10 in the simulated condition for the MovieLens and Amazon Books datasets are 0.75 and 0.89, respectively.

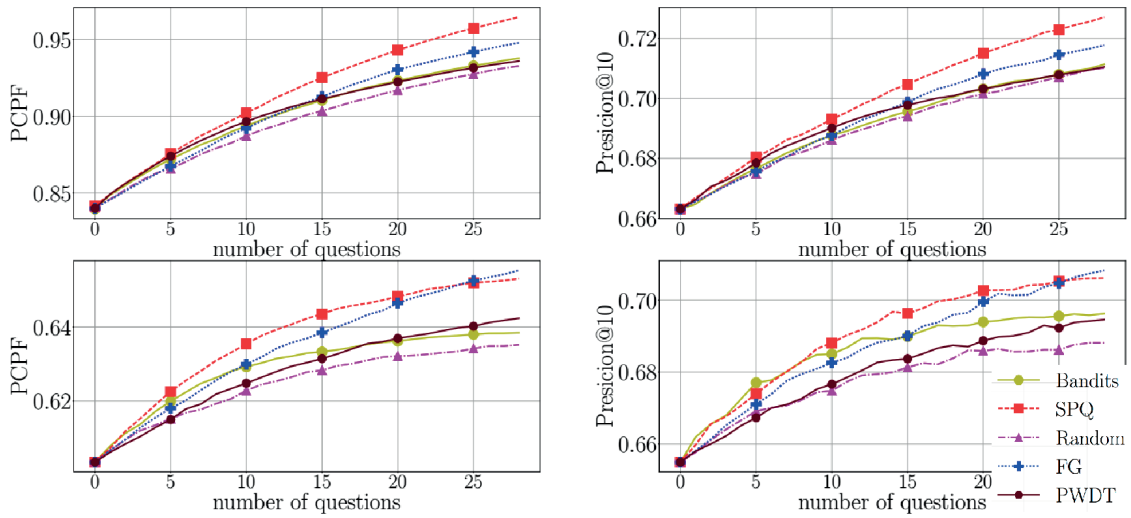


Figure 3.2: Results on the MovieLens movie dataset. (Top): Simulated condition. (Bottom): Real condition. (Left): PCPF. (Right): P@10.

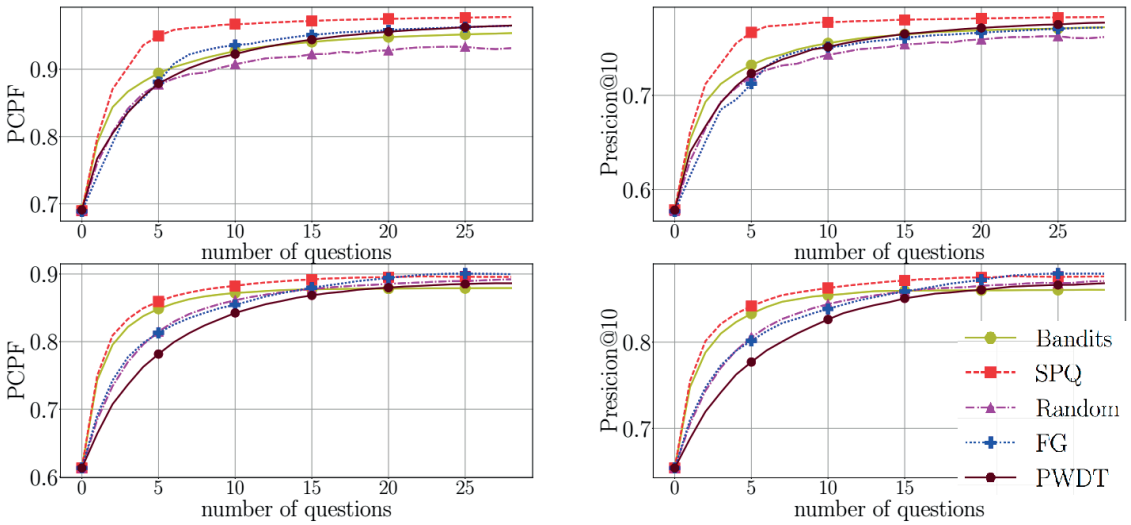


Figure 3.3: Results on the Amazon books dataset. (Top): Simulated condition. (Bottom): Real condition. (Left): PCPF. (Right): P@10.

3.5.1 RQ2.1

To answer RQ2.1 we should understand if the property listed in Section 3.4.4 holds for every pair of algorithms and all conditions. For pragmatic reasons we answer RQ2.1 by comparing methods that achieve state-of-the-art results in terms of Precision @10. Let's turn to the plots shown in the left and center columns of Figure 3.2 and 3.3. We observe that for every pair of methods A and B, and every budget size N (≤ 29), if method A significantly outperforms B in terms of PCPF, then method B does not significantly outperform A in terms of P@10. Hence, we were correct in optimizing Equation 3.2 to produce better lists of recommendations for a user after receiving her completed questionnaire.

3.5.2 RQ2.2

To answer RQ2.2 we turn to Figure 3.2 and 3.3 (right-hand side).

MovieLens movie dataset

All algorithms provide the same performance for questionnaires with a length less than 5: the differences in performance in terms of P@10 between the elicitation methods are not statistically significant ($p > 0.1$) for any pair of methods. In all cases except one, SPQ works significantly better than other methods if the budget is large (P@10; $p < 0.01$); only FG achieves the same performance as SPQ in the Real condition for a large budget ($N > 20$). In the Real condition with a medium size budget SPQ significantly outperforms all other algorithms.

Amazon book dataset in the Real condition

SPQ achieves near-perfect performance only with budgets that are larger than 15; for the smaller budget it has a similar but statistically significantly better performance than Bandits; for budgets larger than 15, SPQ has a similar performance as FG. FG is statistically better than SPQ when the budget is more than 25, but the difference with the quality that is achieved by this algorithm in terms of Precision@10 is less than 0.5%. This finding can be explained by the fact that for long questionnaires (with $N > 15$), in the Real condition, the following property holds: a completed absolute questionnaire provides the same information that is contained in the dataset about a user. Hence, for a long questionnaire the FG method achieves the best result for this experimental condition. But this does not imply that for real users it would lead to the same top performing results.

To conclude our answer to RQ2.2, the importance of the elicitation method depends on the dataset. We distinguish two cases. The first case is datasets like the MovieLens movie dataset, where a non-personalized list of recommendations based on item popularity achieves good results. In that case for a small budget, such as $N = 5$, it is not important to carefully choose the questions. But for a larger budget ($5 < N \leq 15$), it is important to choose a method other than Random. For long questionnaires ($N > 15$), SPQ is the best choice. The second case is for datasets such as the Amazon book dataset, for which a non-personalized recommendation list has far

worse quality than a personalized one. In this case, it is always beneficial to use SPQ to create a questionnaire.

There are two important properties of the elicitation method that should be considered when choosing a method. First, the function the elicitation method minimizes to select a question. We concluded that if the method directly optimizes the loss function, then it works better than others. Moreover, the difference becomes noticeable if the questionnaire is long enough: SPQ and FG achieve better performance than PWDT, despite the fact that they are static, while PWDT is dynamic. A possible explanation is that PWDT optimizes a function that is different from the loss function, namely weighted generalized variance. Second, constraints on selecting items to create a question. The Bandits method selects items that, as it predicts, will be liked by the user. This constraint affects performance, especially when the questionnaire is long ($N > 15$) because at this time Bandits already predicts some user preferences (but not all) and starts to ask questions about the elements that satisfy these preferences. However, the problem that we solve in this chapter has no limitations on the choice of items. Instead, methods that do not have this limitation still ask very informative questions and, therefore, achieve better performance: SPQ selects informative questions and achieves better performance than Bandits, while being static.

Amazon book dataset in the Simulated condition

SPQ is statistically significantly better than the other methods for a budget that is larger than 3 ($p < 0.01$). Moreover, it achieves a high performance after asking only 5 questions, while other algorithms achieve similar results only for a budget that is larger than 15. Thus, compared to other elicitation methods, the length of the questionnaire can be reduced by a factor of three for the same recommendation performance. In all other cases, LFM achieves a near perfect result (on the validation set, RMSE < 0.45).

3.6 Related work

3.6.1 Cold start problem

In Collaborative Filtering (CF), ratings and implicit information are used to provide recommendations [99]. Unlike content-based methods, CF does not require feature engineering. Two popular approaches are clustering [123] and LFM [83]. In LFM, items and users are represented as vectors in a latent space that is automatically inferred from observed data patterns. The dimensions of the space represent information about user preferences and properties of items. The recommendation algorithm predicts ratings using the representations. An influential realization of this approach is MF [83]. MF, like all CF methods, cannot provide reliable ratings for users with limited history and we run into the new user cold-start problem [59, 7].

Previous research suggests different approaches to the new user cold-start problem. Li, Chu, Langford, and Schapire [66] suggest to model representative vectors, not only recommending the best items (exploit), but also getting information about user preferences (explore). Such approaches are usually used for recommender domains that are dynamic: items appear very fast, and usually, new items are relevant (for example,

news and ad recommendation). For a broader range of domains, a popular method for solving the new user cold-start problem involves questionnaire-based[2] approaches where new users are asked a seed set of questions [18, 89, 90, 95, 36].

3.6.2 Questionnaire-based approaches to the new user cold-start problem

Static questionnaire

The standard procedure for creating a static questionnaire chooses seed items independently of new users, and then users rate these items [89]. The underlying algorithmic problem is how to build a seed set that can yield enough preference information to build a good recommender system. One approach is to formulate it as an optimization problem; Anava, Golan, Golbandi, Karnin, Lempel, Rokhlenko, and Somekh [2] find a mathematical expression for the expectation of the number of wrong predictions of ratings after a user has answered a questionnaire. Rashid, Albert, Cosley, Lam, McNee, Konstan, and Riedl [89]'s method selects questions that are not only informative but also satisfy the constraint that users know the items in the questions and are therefore able to make fast decisions. The set of questions can be chosen based on diversity and coverage of the set [33, 72]. That is, latent vectors of the selected questions should have the largest length yet be as orthogonal as possible to each other, i.e., the parallelepiped spanned by these latent vectors should have maximum volume. Fonarev, Mikhalev, Serdyukov, Gusev, and Oseledets [33] and Liu, Meng, Liu, and Yang [72] propose to use Maxvol [38] to find such items.

In line with [2], we propose a method that creates a static questionnaire by solving an optimization problem. But SPQ is the first one to do it for relative questions of the kind shown in Figure 3.1.

Dynamic questionnaire

In a dynamic[3] questionnaire the next question depends on previous questions and answers. A popular way to adapt questions to users' answers is to build a decision tree [35, 48]. In each node of the tree, there is a question. The children of the node are the next questions depending on the answer to the question in the current node. Thus, each node has as many children as there are answers to the current question. Consequently, dynamic methods can be memory inefficient, but usually, they provide better results than static ones. The method presented in this chapter, while static and memory efficient, achieves better results than dynamic baselines. We leave a dynamic variant as future work.

[2] In the literature, such methods are also called "interview-based." To avoid ambiguity, we stick with "questionnaire-based."

[3] In the literature, such methods are also called "interactive." To avoid ambiguity, we stick with the term "dynamic."

Asking relative questions

Another way to solve the new user cold-start problem is to ask relative questions. As stated in [56], this approach has all the advantages of pairwise preference approaches: it is faster and easier for a user to answer relative questions, and users' relative preferences are more stable over time. Jones, Brun, and Boyer [56] also show that relative questions can deal with the rating calibration issue. However, the number of possible relative questions is proportional to the square of the number of items and therefore it is computationally expensive to select optimal preference questions. Thus, only a few papers have so far used this approach [95, 18].

There are several solutions to avoid brute-forcing all pairs of items. Rokach and Kisilevich [95] suggest to cluster items before selecting questions. However, valuable information about items may be lost in the clustering. Moreover, this algorithm can be memory inefficient due to the need to memorize all answers of all users. Christakopoulou, Radlinski, and Hofmann [18] present several bandit-based algorithms for online recommendations, including asking relative questions in an online setting.

To summarize, we propose a method that creates a static questionnaire consisting of relative questions, which allows us to build a personalized list of recommendations for a new user. Similarly to [2], we formulate our task as an optimization problem, but we ask relative questions [18, 95] instead of absolute ones. Therefore, we cannot perform the kind of brute-force search of all available questions that is performed in [2]. Hence, we propose a new solution for the optimization task defined in [2]. Our method differs from [95], as we offer a procedure that selects the most informative questions without clustering items, thus avoiding losing any information. Moreover, our method solves the optimization problem that minimizes the expectation of misclassified personalized preferences of one item over another, while the method in [95] minimizes the weighted generalized variance, which is not directly related to the loss that we minimize. The main advantage of SPQ over [18] is that we optimize informativeness of questions without constraints.

3.7 Conclusion and Future Work

We have proposed a static preference questionnaire generation method, called SPQ, that extends earlier work [2, 18, 36, 89, 90, 95] on approaching the new user cold-start problem for recommender systems by asking a set of seed questions. Our main research question is *How to optimally generate a preference questionnaire, consisting of relative questions, for new users that will help to solve the new user cold-start problem?* Generating a short questionnaire that is sufficiently informative so as to derive a high-quality list of personalized recommendations, is crucial for a recommender system to be able to engage new "cold" users. We are the first to address the problem of generating a list of relative questions as an optimization problem. The use of relative questions is beneficial [56] because:

(1) users find it easier to provide feedback through relative questions than through absolute ratings;

(2) absolute ratings suffer from calibration issues; and

(3) answers on relative questions are stable over time.

We have demonstrated theoretically how to solve the optimization problem and how to avoid brute-forcing all possible questions as is implemented in [2]. Also, we have shown experimentally that minimizing the proposed SPQ objectives leads to a high-quality list of personalized recommendations. We have performed experiments on two datasets: the MovieLens movie dataset and the Amazon book dataset. These datasets differ in the type of user feedback and in the diversity of items. Also, we have considered two experimental conditions to evaluate elicitation procedures:

(1) experiments in *Simulated* shows results in the ideal situation when users behave according to the LFM; and

(2) experiments in the *Real* condition do not rely on any user rating model.

We have compared SPQ with multiple state-of-the-art baselines [2, 18, 95] that solve similar but different tasks. SPQ outperforms all baselines on both datasets independently of the length of the questionnaire, and it achieves a statistically better performance than the baselines for questionnaires whose length exceeds 5. On the Amazon book dataset in the real condition, SPQ was able to reduce the length of the questionnaire by a factor of three. Moreover, SPQ, a static method, has demonstrated better results than the dynamic baselines [18, 95]. SPQ outperforms the dynamic methods because it optimizes the loss function and has no constraints on the questions, while dynamic baselines either optimize a substitute function instead of the LFM loss function or have constraints on the questions that are not necessary for the problem solved in this chapter.

In future work, we plan to work on the following important aspect to improve a questionnaire. SPQ may ask very informative questions about items that a user might not like. To overcome this problem, we plan to make SPQ dynamic to make sure that we ask attractive questions for the user while receiving almost the same amount of information.

In summary, to answer research question RQ2, we formulated the problem of creating an informative preference questionnaire as an optimization problem and proposed an algorithm that solves this optimization problem. Next, we will focus on the different criteria of preference questionnaires and study how to create a preference questionnaire in which a user likes the displayed documents.

4

A Deep Reinforcement Learning-Based Approach to Query-Free Interactive Target Item Retrieval

In this chapter, our aim is to answer RQ3: how to find the optimal strategy for selecting documents to display, which helps to increase user satisfaction with an ongoing session, combining short-term and long-term preferences. In particular, we propose a method for finding an optimal strategy for a recommender system that increases user satisfaction with an ongoing session for the task of query-free interactive target item retrieval. In this task, a user has a concept or category of items in mind and the system's task is to find the right item that falls within the category. Traditional recommender systems tend to recommend items greedily, i.e., they display very similar elements that have the highest estimated relevance. However, early in a session, the degree of uncertainty about the category of items that is currently of interest to the user is high. We propose an approach based on deep reinforcement learning in which we combine a reinforcement learning-based recommendation agent with a relevance feedback step. Specifically, we introduce an actor-critic-based framework to iteratively select sets of items based on real-time feedback from users and their purchase history, thereby maximizing user satisfaction within the entire session.

4.1 Introduction

Over the last decade, e-commerce sites have become ubiquitous [27]. Customers use them every day for a range of purposes: from exploring available items to purchasing a specific item [111, 113]. A large portion of purchases in e-commerce sites is focused on fashion products [17], where images of a product may play an important role. Several methods have been introduced to extract clothing attributes from images have been proposed, with successful take-up in industry [68, 127]. By automatically identifying attributes of fashion items, systems enable users to complement traditional search based on keyword descriptions [68, 127] . However, usually, the attributes are predefined, and not all users are familiar with the vocabulary being used.

This chapter is under submission as [103].

In this paper, we focus on the task of fashion item retrieval in e-commerce sites. We consider the scenario in which a user wants to find and obtain an item from a particular category, which can be represented by various "mental pictures" that may vary from a particular image to the user's impression. In this context, the task for an e-commerce site is to find the right item that fits the desired category and in parallel show result pages that contain fashion items that are close to the target category. We address this problem, to which we refer as the task of *query-free interactive target item retrieval for experienced users*.

In e-commerce sites, a fashion item retrieval can be done by combining signals from a user's history of purchases and the signals from the user's current interaction with a system. To utilize the user's historical interaction with the system, the traditional Recommender Systems (RSs) can be used, which predicts user preferences based only on past interactions between users and the system [132, 44, 81, 92, 83].

The interaction between a user and a RS consists of two stages:

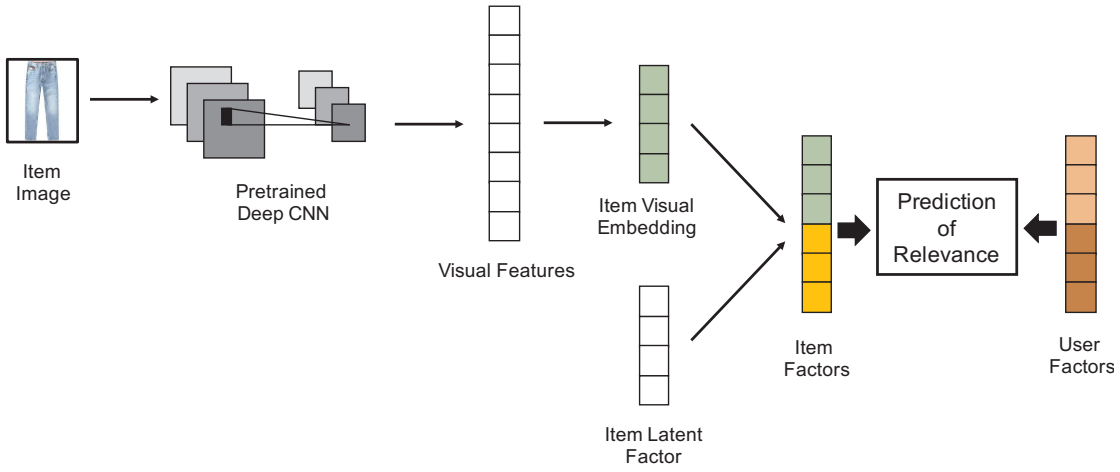
(1) the RS selects items to be displayed on the result page, and

(2) the user views the result page and provides some level of feedback, such as clicks on some items or adding items to a cart.

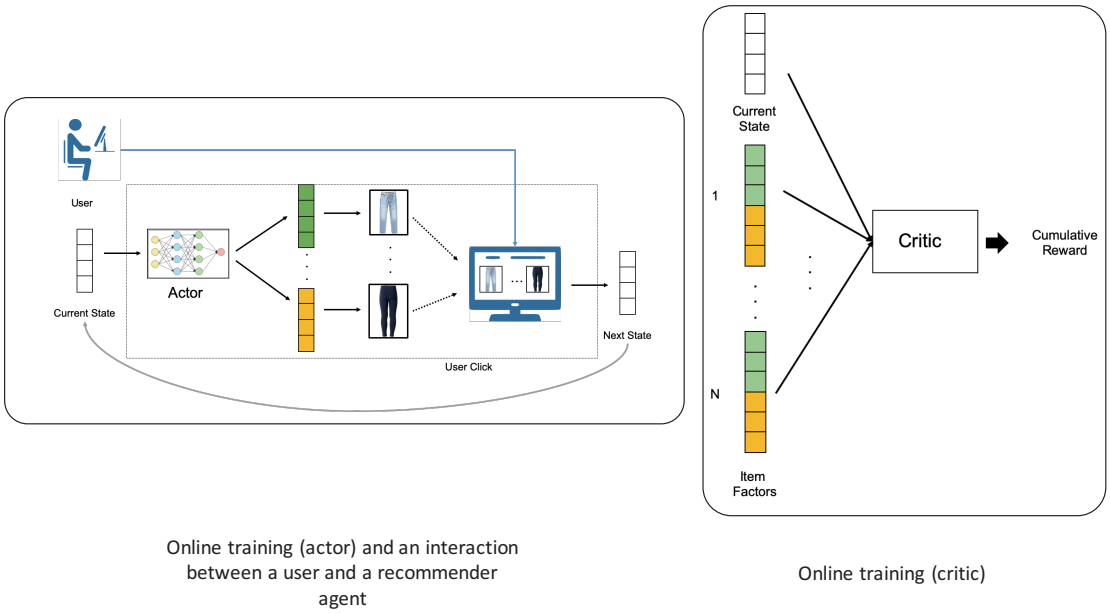
This type of interfaction does not fit well at the start of a new session, which is a stage of uncertainty. In the task of *query-free interactive target item retrieval* the target category is unknown for the system. Previous work has proposed to use Relevance Feedback (RF) mechanisms [8, 30, 31, 126, 131] to minimize the RS's uncertainty about a user's target category. In RF, a system shows a subset of items to a user and the user interacts with the system by clicking on the item that is (supposedly) most similar to the target item. Systems that use an RF mechanism to understand a user's current preference aim to minimize the length of the session, by displaying diverse items on the result page. Consequently, not all the shown items satisfy the user's preference and a user can be unsatisfied with the results shown.

We propose a RL approach to finding the optimal strategy for selecting items to display on the result page in which the selected items are shown to a user in a grid. Our approach, called ActorCriticRS, maximizes the expected long-term cumulative reward per user, where the cumulative reward corresponds to user satisfaction during a session. Moreover, ActorCriticRS adapts its strategy for different approximations of user satisfaction with the result page. In the context of item search, user satisfaction depends on a variety of factors and the relation between a user's perceived search satisfaction can be complex and remain unclear [113]. ActorCriticRS can automatically trade-off between greedy exploitation of learned past interests and exploration aimed at uncovering current interests. ActorCriticRS is built upon a Reinforcement Learning framework, called Actor-Critic. The use of a RL method is challengnig for a RS, since the state and action spaces are very large, and recommendations must be given quickly. We overcome these problems by using a model-free Deep Reinforcement Learning based algorithm. The overall pipeline is shown in Figure 4.1.

To summarize, in this chapter, we propose a deep RL-based framework to help users find an item from a target category, using her history of purchases and information gleaned from the session in progress. Our contribution is three-fold:



(a) Obtaining users' and items' embedding from the users' historical interactions.



(b) An interaction between a user and a recommender agent and training using interactions in the current session.

Figure 4.1: Overall recommendation framework.

(1) We propose a deep reinforcement learning-based theoretical framework for target item retrieval.

(2) We propose simple implementations of each component of the framework.

(3) We analyze the quality of the framework depending on three aspects:

   (a) the number of items displayed on the result page,

   (b) the duration of the session measured by the number of result pages shown during the session, and

   (c) our approximation of user satisfaction with the result page.

4.2 Problem Description

In this chapter, we consider a task that is similar to "mental image retrieval" [17]. In our task, a user wants to find an item from a target category. However, the user does not know the category name and, therefore, does not provide a query. For example, a user wants to buy leggings that she saw before, but she does not know what kind of leggings or may consider tights instead. Unlike for "mental image retrieval," in the task of *query-free interactive target item retrieval for experienced users* the user is known to the system. More formally, we follow the protocol detailed in Section 4.2.1 below.

4.2.1 The recommendation task

Consider a user with a history of purchases who initiates a new session on an e-commerce site. When she starts a session, she has a concept or category in mind, but cannot formulate it and does not submit a query to the system. The RS and the user interact to uncover the implicit need. Interactions between user and recommender agent consist of the following steps:

(1) user u starts a search session in which she wants to find an item from the target category c;

(2) recommender agent shows her n items from its dataset $I = \{i\}_1^n$;

(3) user u provides feedback by clicking on the best item from I;

(4) recommender agent obtains a reward r equal to the user satisfaction of the result page;

(5) recommender agent processes the feedback and creates a new list of items; and

(6) the search session ends when the session is too long.

The task of the recommender agent is to maximize the cumulative reward, mathematically:

$$\sum_{t=0}^{t=\infty} \gamma^t \cdot r_t, \tag{4.1}$$

where r_t is the user satisfaction obtained on t-th result page of the session; r_t is equal to 0 if the user leaves the recommender system, examining less than t result pages. We propose a Reinforcement Learning (RL) approach to finding a strategy that a recommender agent should follow in order to maximize the expected cumulative reward.

Next, we formalize the concepts that are necessary to use RL.

4.2.2 Recommendation as a Markov decision process

We formulate the process of interaction between a user and a recommender agent as a Markov decision process and use RL to automatically learn an optimal strategy for selecting items to display on the result page. The optimal strategy is obtained by maximizing the long-term cumulative reward (see Eq. 4.1) during the session. Now we formally define the five necessary components of an MDP (s, A, r, p, γ).

(1) A *state* $s \in S$ is defined as an approximation of the current user preference. At the beginning of the session, the initial state s_0 is generated using the user's purchase history:

$$s_0 = Purchases = \{i_1, \ldots, i_k\}, \tag{4.2}$$

where $i_1, \ldots, i_k$ are the items that a user bought before. On the t-th result page a user provides her feedback by clicking on one of the shown items:

$$F_t = (i^t_{clicked}, \{i^t_{1,non\text{-}clicked}, \ldots, i^t_{n-1,non\text{-}clicked}\}), \tag{4.3}$$

where $i^{t+1}_{clicked}$ is a clicked item, $i^{t+1}_{j,non\text{-}clicked}$ is a non-clicked item. After the t-th round of interaction between the user and the recommending agent, the state s_t is generated based on the user's purchase history and feedback that has already been received from her during the session:

$$s_t = (Purchases, F_1, \ldots, F_t)\,. \tag{4.4}$$

(2) The *action space* A is defined as the set of all subsets containing n items:

$$\{i_1, \ldots, i_n\} \in A. \tag{4.5}$$

The chosen items are displayed on the result page.

(3) The user examines the result page, and the recommender agent obtains an *immediate reward* r_t, equal to the user's satisfaction with the current result page.

(4) The *transition probability* $p(s_{t+1}|s_t, a_t)$ defines the state transition from s_t given an action a_t.

(5) $\gamma \in [0, 1]$ determines the discount factor for future rewards. If $\gamma = 0$, the recommender agent becomes a greedy agent and considers only immediate rewards. If $\gamma = 1$, the recommender agent considers all future rewards without discount.

We model the recommendation task as an MDP defined by (S, A, r, p, γ). Here, the recommender system is an agent that interacts with a user during a sequence of time steps. At each time step the agent chooses an action by displaying a set of items and receives a reward from the user, that is, her satisfaction with the result page. A user clicks on the item closest to the target category. The environment updates its state. The agent's goal is to find a policy $\pi : S \to A$ that maximizes the cumulative reward (see Eq. 4.1) over a session.

4.3 Theoretical Framework

In this section, we describe our framework for solving the task of query-free interactive target item retrieval for experienced users. It consists of three components (see Figure 4.2):

(1) an *item and user embedder*;

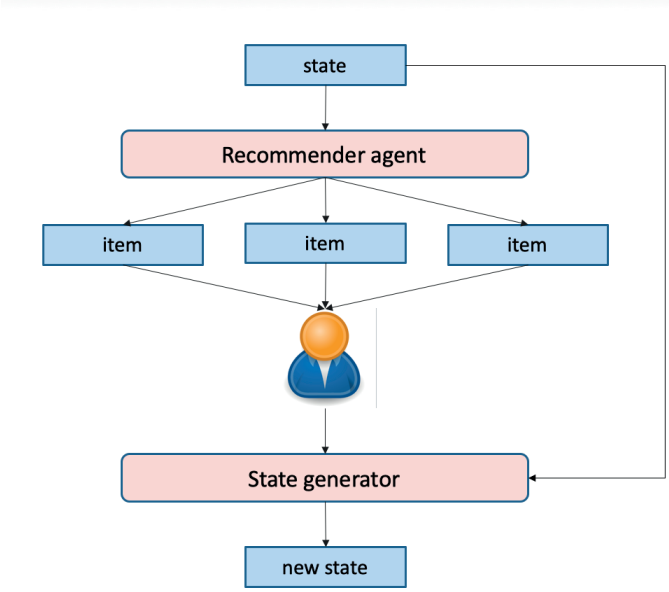


Figure 4.2: Framework for query-free interactive target item retrieval for experienced users.

(2) a *state generator*, and

(3) a *recommender agent*.

Next, we describe the desired properties of each component; their specific implementation is given in Section 4.4.

4.3.1 Item and user embedder

The item and user embedder uses the history of users' purchases to find a low-dimensional representation of users and items in the same latent space. The representation of users is used as an initial state when a new query-free session starts, and should be an approximation of users' preferences. Dimensions in the latent representation of items should reflect the most valuable characteristics of the items. Therefore, the dot product between a user's representation and an item's representation should reflect the user's preference of this item. The problem of finding such a low-dimensional representation using the history of users' feedback for items, can be solved by a recommendation algorithm.

4.3.2 State generator

A state generator generates a low-dimensional representation of the current state. As a vector representation of the initial state $\mathbf{s}_0$ we use the user representation generated by the *items and users embedder*. Therefore, the representation of the initial state $\mathbf{s}_0$ reflects the user's preferences, using all the data that is available about the user at the beginning of the session, namely her purchase history. While the user interacts with the system, the representation of the state changes to reflect information gleaned from the session in progress. More precisely, the representation of the state $\mathbf{s_t}$ should be changed

to reflect the user's clicks on the $(t+1)$-st result page of the session: $\mathbf{s_{t+1}}$ should be closer to the clicked item on this page than to all non-clicked items on the same page.

The constraints on the *state generator* can be rephrased in the following way. Using s_0, which can be considered to be the initial query q_0, the system returns the initial result page to a user. The user marks the most relevant item by clicking on it. The received feedback together with the initial query are used to estimate a new state that can be considered as a new query q_t. The new query q_t should maximize the similarity with relevant items while minimizing the similarity with non-relevant items. Below, we solve this task by methods that utilize an RF mechanism to improve the relevance of items on a result page.

4.3.3 Recommender agent

The recommender agent uses a low-dimensional representation of the state to produce a result page of items. After displaying the page, the recommender agent obtains a reward, which is the user satisfaction for the page. The task for the agent is to maximize the expected cumulative reward (see Eq. 4.1) obtained during the session. This task can be solved by a RL approach. However, not every RL method can be used as the recommender agent. The action space is very large: the number of possible result pages is even larger than the number of elements, which in ordinary e-commerce platforms equals 100K or more. Moreover, the space of states is continuous. Therefore, a recommender agent should be implemented using an RL method that works within the constraints described above: a continuous state space and a very large or continuous action space.

4.4 Method

In this section, we describe our implementation of the three components of the framework introduced in Section 4.3: an items and users embedder, a state generator, and a recommender agent.

4.4.1 Item and user embedder

As we discussed in Section 4.3.1, any matrix factorization model can be used as a *item and user embedder*. We use a state-of-the-art MF model that is able to utilize not only a user's purchase history but also images of items, viz. VBPR [44]. VBPR is a preference predictor that is built on top of MF; it embeds all items and users in the same low-dimensional latent space. He and McAuley [44] partition dimensions in the latent space: some dimensions represent visual preferences, others non-visual factors. The structure of the VBPR predictor is shown in Figure 4.1a. The VBPR predictor takes the following form

$$r_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u^T \cdot \gamma_i + \theta_u^T \cdot \theta_i, \tag{4.6}$$

where α is a global offset, β_u and β_i are user and item bias terms, γ_u and γ_i are latent representations of a user u and item i in non-visual subspace, while θ_u and θ_i are latent representations of a user u and item i in visual latent space. The visual latent

representation of items θ_i is obtained by projection of features of items' images to the latent space:

$$\theta_i = \mathbb{E} f_i, \tag{4.7}$$

where $\mathbb{E}$ is a matrix that embeds Deep CNN features [62] of images of item i into the visual space. As vector representations of users and items, we use a concatenation of their embedding into visual and non-visual low-dimensional latent spaces: $\mathbf{u} = \langle \gamma_u, \theta_u \rangle$, $\mathbf{i} = \langle \gamma_i, \theta_i \rangle$.

4.4.2 State generator

As a state generator, we use the Rocchio algorithm [94], which is a classic algorithm that utilizes information obtained using a RF mechanism. The Rocchio algorithm is a simple but efficient method that is used in many approaches that incorporate RF information into the vector space [8, 30, 126, 131]. The Rocchio algorithm changes the representation of the state $\mathbf{s}_t$, bringing it closer to the representation of the relevant items and further from the representation of non-relevant items. In the problem of query-free interactive item retrieval, the Rocchio algorithm moves the state $\mathbf{s}_t$ in the direction of the clicked item and in a direction opposite to non-clicked items:

$$\mathbf{s}_{t+1} = a \cdot \mathbf{s}_t + (1 - a) \cdot \frac{1}{n-1} \cdot \sum \left(\mathbf{i}_{clicked} - \mathbf{i}_{non\text{-}clicked} \right), \tag{4.8}$$

where a is a coefficient for balancing the influence of new information on information gained earlier, and n is the number of elements displayed on the page. To reflect the fact that feedback from the user becomes less informative once we have received a user's relevance for several result pages we set a to be a function of t:

$$a(t) = 1 - 0.8^t \cdot a', \tag{4.9}$$

where a' is a parameter of the *state generator*. The final formula for updating states is the following:

$$\mathbf{s}_{t+1} = a(t) \cdot \mathbf{s}_t + (1 - a(t)) \cdot \frac{1}{n-1} \sum \left(\mathbf{i}_{clicked} - \mathbf{i}_{non\text{-}clicked} \right). \tag{4.10}$$

4.4.3 Recommender agent

An RL algorithm that can be used as the *recommender agent* should be able to deal with state and action spaces that are continuous high-dimensional spaces (see Section 4.3.3). Conventional RL methods such as Q-learning [128] and POMDP [43] become infeasible in this case because it is impossible to estimate the transition probabilities and to store the Q-value table. Thus, we use Deep Reinforcement Learning [69, 84], which does not estimate the transition probability and does not store Q-values in a table.

The Actor-Critic framework [69] is able to select actions efficiently when the action space is continuous. It consists of two parts: an Actor and a Critic (see Figure 4.3). The Actor generates an action using the current state. The Critic predicts the expectation of cumulative reward, noted as $Q(\mathbf{s}_t, \mathbf{a}_t)$, given that an action $\mathbf{a}_t$ is taken in state $\mathbf{s}_t$.

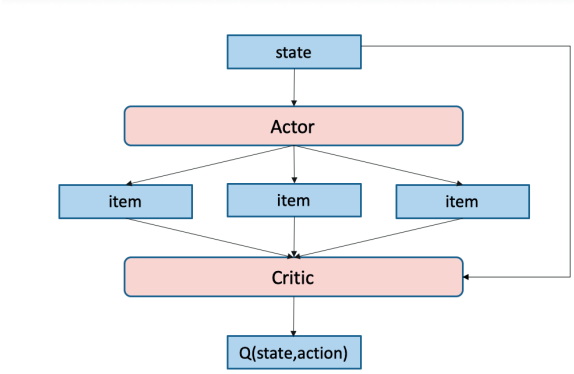


Figure 4.3: Actor Critic Framework.

According to the judgment from the Critic, the Actor updates its parameters to choose an action $\mathbf{a}_t$ that maximizes $Q(\mathbf{s}_t, \mathbf{a}_t)$ in the state $\mathbf{s}_t$. In this way, the Actor outputs better actions in future iterations. Below we describe the architecture of the Actor and Critic in detail.

Actor

The Actor chooses n items to display on a result page, depending on the current state $\mathbf{s}_t$. The Actor first generates a pseudo-action, consisting of n vectors in the latent space $\{\hat{\mathbf{i}}_1, \ldots, \hat{\mathbf{i}}_n\}$. Then, for each generated vector $\hat{\mathbf{i}}_j$, the most similar item is selected that was not shown in the session:

$$i_j = \arg\max_{i_j}(\mathbf{i}_j^T \cdot \hat{\mathbf{i}}_j). \tag{4.11}$$

This procedure of selecting items, using vectors generated by the Actor, is exactly the same as the Mapping Algorithm described in [138]. The details of the Mapping Algorithm are presented in Algorithm 5.

Algorithm 5 Mapping from vector generated by the Actor to items

1: **Input:** Set of items $\mathbb{I}$ that have not been displayed before, n vectors in the latent space $\{\hat{\mathbf{i}}_1, \ldots, \hat{\mathbf{i}}_n\}$ that are generated by the Actor.
2: **Output:** Set of items $I_{current} = \{i_1, \ldots, i_n\}$, that will be shown to a user.
3: **for** $j = 1, \ldots, n$ **do**
4: Select the most similar item $i_j \in \mathbb{I}$ according to Eq. 4.11
5: Add i_j to $I_{current}$
6: Remove item i_j from $\mathbb{I}$

To understand the implementation of the Actor, it is important to analyze desiderata for its behavior. The Actor should change its behavior over time. Indeed, with more feedback received from a user, the state $\mathbf{s}_t$ moves closer to the embedding of the target item. This means that the vectors generated by the Actor should get closer to the state as the time step increases. To achieve this, we design an Actor consisting of two components:

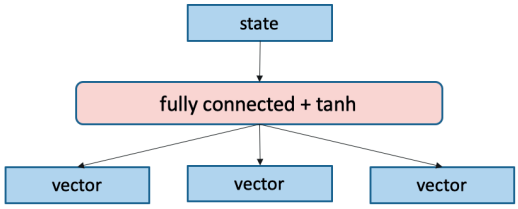


Figure 4.4: Trainable part of the Actor.

(1) one is trained, and

(2) the other is greedy.

The trained component $A_{trained}$ generates n vectors $\{\widetilde{\mathbf{i}}_1, \ldots, \widetilde{\mathbf{i}}_n\}$ in the latent space and has the following architecture: it is a neural network with one fully connected layer and a $\tanh$ non-linearity (see Figure 4.4). Vectors generated by the Actor are a linear combination of the current state $\mathbf{s}_t$ and vectors generated by $A_{trained}$:

$$\hat{\mathbf{i}}_j = (1 - \epsilon_t^{actor})\,\widetilde{\mathbf{i}}_j + \epsilon_t^{actor}\mathbf{s}_t, \tag{4.12}$$

where ϵ_t^{actor} is a coefficient between 0 and 1, monotonically increasing with t. More precisely, ϵ_t^{actor} is obtained by the following formula:

$$\epsilon_t^{actor} = \frac{1}{1 + \exp(-a_{actor} \cdot t + b_{actor})}, \tag{4.13}$$

where a_{actor} and b_{actor} are trainable parameters of the Actor.

Critic

The Critic predicts the $Q(\mathbf{s}_t, \mathbf{a}_t)$-value in the current state $\mathbf{s}_t$, if the action taken is a_t. In RS, the state and action spaces are very large, therefore, estimating the action-value function $Q(s_t, a_t)$ for each state-action pair is infeasible. Thus, we use an approximation function to estimate the Q-value. In particular, we use a neural network as the Critic, which has a similar architecture as the Actor. The Critic also consists of two parts. The first part is a neural network with one fully connected layer and a $\tanh$ non-linearity (see Figure 4.5). The second part is an addition ϵ_t^{critic} that is dependent on the step t:

$$\epsilon_t^{critic} = \frac{1}{1 + \exp(-a_{critic} \cdot t + b_{critic})}, \tag{4.14}$$

where a_{critic} and b_{critic} are trainable parameters of the Critic.

4.5 Experiments

4.5.1 Research questions

In this chapter, we consider the task of query-free interactive target item retrieval for experienced users. To address it, we have proposed a framework with three components:

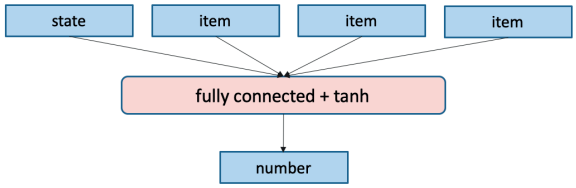


Figure 4.5: Neural network part of the Critic.

an item and user embedder, a state generator, and a recommender agent. We mainly focus on the implementation of the recommender agent, while using simple but efficient solutions for the other components. We address the following research questions:

(RQ3.1) Does user satisfaction within a session increase when items are displayed to a user according to the strategy obtained by an RL-based method instead of the traditional greedy strategy?

(RQ3.2) How does the number of items displayed on a result page affect the performance of the recommender agent?

(RQ3.3) How does the performance of a recommender agent depend on the way user satisfaction is measured?

4.5.2 Dataset

Dataset description

As a dataset, we use a large crawl of item reviews from Amazon created by McAuley, Targett, Shi, and Van Den Hengel [81]. The decision to choose the best item displayed on the result page becomes easier when the decision can be made according to images of the items. That is why we only use a category of items for which visual features have already been demonstrated to be important, namely *Women's Clothing* [44]. Following [44], we take users' review histories as implicit feedback (if the user wrote a review that is considered positive, otherwise it is negative), and process the dataset so that each user has at least 5 reviews. The final dataset contains ∼100K users, ∼330K items, ∼850K reviews. The reviews in this dataset contain not only user id, item id and review text, but also timestamp, which helps to build the purchase histories of the users.

Visual features

As visual information about items, we use the image features extracted in [44]. More precisely, one image was collected for each item. Then the Caffe reference model [24], which implements the CNN architecture proposed by [62] was used. The model architecture has 5 convolutional layers followed by 3 fully-connected layers and has been pre-trained on 1.2 million ImageNet (ILSVRC2010) images. Finally, the output of the second fully connected layer (i.e., FC7) was used to obtain a 4,096 dimensional vector of visual features.

Metadata

The dataset contains a taxonomy of the items. There are about 2,700 different categories in the taxonomy. We create a *Categorical space* by a hard-code embedding of the items: each item contains 0 and 1 as coordinates, where the coordinate equals 1 if an item belongs to the corresponding category in the taxonomy. We use the *Categorical space* to simulate user clicks in a session.

4.5.3 Simulation

To evaluate our strategy of selecting items to show to a user we should be able to obtain users' feedback on the selected items. However, this information is not directly available from the historical logs to which we have access. That is why we simulate users feedback. We simulate users clicks using the *Categorical space*: in the simulation, a user always clicks on the item that is most similar to the target one in the *Categorical space*. In the case when several items are equally good, a simulated user clicks on one of them randomly.

In our simulation for each user, we choose the last purchased item as the target item. All other purchases are used to estimate the user's preferences. Importantly, the taxonomy is *not* used by any of our recommendation methods: all methods use only the users' history of purchases and features from images of the items.

4.5.4 Baselines

Below we describe different strategies for selecting items to show to a user, which we use as our baselines. The *item and user embedder* and the *state generator* remain the same for all the methods that we compare.

Greedy

A common strategy for selecting items to show to a user is a greedy strategy [8, 44, 83, 92, 126, 131]. The greedy strategy selects items with the highest predicted relevance scores. In this chapter, the predicted relevance scores are proportional to the scalar product between low-dimensional representations of items received from *Item and user embedder* and low-dimensional representations of the current state.

Methods that use a greedy strategy to select items maximize immediate user satisfaction with the result page, rather than long-term user satisfaction with the session. However, it is a common practice in methods that use RF to use a greedy strategy as a *recommender agent* and the Rocchio algorithm as a *state generator* [8, 126, 131].

It is also worth noting that the initial state is obtained by training *VBPR* model. Consequently, the greedy recommender agent can be considered as an interactive version of the *VBPR* model.

Random

The *Random* baseline randomly selects items to display on the result page.

IRF

A very efficient algorithm *IRF* for query-free image retrieval using interactive relevance feedback was proposed in [31]. *IRF* achieves good quality and is widely used in content-based image retrieval with relevance feedback [4, 114, 115]. The algorithm selects items by growing subsequent Voronoi cells [25] based on the item similarity distances and their current score of relevance, denoted as $score_t(i_k)$. The mass of each Voronoi cell $score_t(cell)$ is a fraction of the total mass:

$$score_t(cell) = \frac{1}{n} \cdot \sum_{i_k \in I} score_t(i_k). \tag{4.15}$$

More specifically, the first item selected by *IRF* is the one with the highest relevance score:

$$i_{t,1} = \arg\max_{i_k} score_t(i_k). \tag{4.16}$$

Then the Voronoi cell C_1 is grown by including items one by one in the order of their similarity with $i_{t,1}$ until the mass of C_1 is less than $score_t(cell)$. The k-th item is selected among the items from outside the chosen Voronoi cells:

$$i_{t,k} = \underset{i_k \in I \setminus C_1 \cup C_2 \cup \cdots \cup C_{k-1}}{\arg\max} p_t(i_k). \tag{4.17}$$

The Voronoi cell C_k is grown by including items one by one among the items from outside the chosen Voronoi cells in the order of their similarity with $i_{t,k}$ until the mass of C_k is less than $score_t(cell)$.

Since *IRF* does not scale for a large collection of items, we first select 500 items with the highest predicted relevance scores, and then use *IRF* only for these items. As a predicted score of relevance we use the scalar product of the latent embedding of the state and the latent representation of items:

$$score(i_k) = \mathbf{s}_t \cdot \mathbf{i}_k. \tag{4.18}$$

As similarity between items, we use the scalar product of their latent representations.

MGD

An algorithm called MGD has been proposed in [101]. MGD is a reinforcement learning based algorithm that can be used as a recommender agent. MGD has been designed for online learning to rank problems that can be reformulated as a query-free target item retrieval. More precisely, at each timestep t MGD selects n rankers. Then, using user clicks, the best ranker of the n selected rankers is inferred. Finally, MGD adjusts the prediction of the vector representation of the globally best ranker. For our task of query-free item retrieval, the individual items play the role of rankers.

MGD is based on the DBGD [136] algorithm. The key difference between MGD and DBGD is that MGD selects n rankers, while DBGD selects 2. DBGD is a bandit-based algorithm that optimizes a cumulative reward, performs a trade-off between exploration and exploitation, and has theoretical guarantees.

4.5.5 Evaluation methodology

To answer the research questions listed in Section 4.5.1 we perform experiments with a standard configuration for RS, changing one parameter of the standard configuration at a time. In the standard configuration, we display 10 items on the result page and use r_{mean} as an estimation of user satisfaction with the result page:

$$r_{mean} = \frac{1}{n} \cdot \sum_{i_k} relevance(i_k). \tag{4.19}$$

Answering RQ3.1

To obtain an answer to RQ3.1 (Section 4.5.1), we compare the r_{mean} score obtained by MGD and ActorCriticRS with the r_{mean} score obtained by the *Greedy* appraoch.

Answering RQ3.2

To understand the impact of the number of items shown to a user on the result page, we compare r_{mean} obtained by *Greedy*, *Random*, *IRF*, *MGD* and ActorCriticRS in the standard configuration and vary the number of items shown to a user. More precisely, we run experiments when 3, 5, 7 and 10 items are shown to a user on the result page.

Answering RQ3.3

To obtain an answer to RQ3.3 (Section 4.5.1), we compare rewards obtained by *Greedy*, *Random*, *IRF*, *MGD* and ActorCriticRS in the standard configuration and vary the function that we use to approximate user satisfaction with the result page. Specifically, we choose 2 reward functions: one that is obtained by Eq. 4.19 and another that is obtained by:

$$r_{max} = \max_{i_k} relevance(i_k). \tag{4.20}$$

The intuition behind r_{max} is that a user is satisfied if at least one item satisfies her preference.

4.5.6 Training models

Item and user embedder

To obtain visual-semantic representations of users and items, we use the code from [44]. In [44], all purchases are divided into three subsets: training, validation and testing. We use the last purchases as a test set. The validation set is created by selecting for each user u a random item v_u from her purchases that are not in the test set. All remaining data is used for training. We train the VBPR model with 32 dimensions of the non-visual latent representation and 32 dimensions of the visual latent representation.

State generator

The proposed state generator has only one hyperparameter, namely a' (see Eq. 4.9). We chose a' to be 0.5. We tried other values of a' (0.3, 0.5, 0.8), but the cumulative reward obtained by *Greedy*, *IRF* and MGD was the largest when a' is set to 0.5.

Recommender agent

For all configurations, that is, different numbers of items displayed on the result page and different ways to measure user satisfaction with a page, we use the same hyperparameters of ActorCriticRS. Specifically, the batch size is equal to 128, the learning rate of the Actor is $1e-6$, the learning rate of the Critic is $1e-4$.

To train the Actor-Critic framework we use DDPG [69]. More precisely, the Critic is trained to minimize the following loss function:

$$L(\theta_{critic}) = E_{s,a,r,s'}\left[(r + \gamma \cdot Q(s', a') - Q_{\theta_{critic}}(s, a))^2\right], \tag{4.21}$$

where θ_{critic} are the trainable parameters of the Critic, and s, a, $Q(s', a')$, r are sampled from the replay buffer [84].

The Actor is trained to maximize the Q value predicted by the Critic:

$$R(\theta_{actor}) = \max E_s\left[Q_{\theta_{critic}}(s, Actor_{\theta_{actor}}(s))\right], \tag{4.22}$$

where θ_{actor} are the trainable parameters of the Actor.

Following the standard procedure of training DDPG, the training algorithm has 2 stages in each iteration:

(1) a transition stage, and

(2) parameter update stage.

In the transition stage, new data is added into the replace buffer. In the update stage, the parameters of the Actor and Critic are updated using transitions that are sampled from the replace buffer.

To achieve a statistically significant result, we perform 10-fold cross-validation.

4.6 Results

4.6.1 Answer to RQ3.1

To understand whether a reinforcement learning approach helps to improve user satisfaction with a session, we turn to Tables[1]–4.1, 4.2, 4.3, 4.4. These tables show the reward r_{mean} depending on the result page number in the session: $r_{mean}(p_1)$ is the reward for the first result page in the session, $r_{mean}(p_2)$ for the second result page, and so on. We compare the *Greedy* method with two reinforcement learning based methods: MGD and ActorCriticRS.

[1] All results that are marked as $*$ in Tables 4.1 - 4.5 are statistically significant (t-test, p-value < 0.01)

Table 4.1: Rewards obtained on the first, second, . . . , fifth result page of a session, when 3 items are displayed

Method	$r_{mean}(p_1)$	$r_{mean}(p_2)$	$r_{mean}(p_3)$	$r_{mean}(p_4)$	$r_{mean}(p_5)$
Random	0.39	0.39	0.39	0.39	0.39
IRF	0.46	0.49	0.50	0.51	0.52
Greedy	**0.47**	0.48	0.49	0.49	0.50
MGD	**0.47**	0.49	0.50	0.51	**0.52**
ActorCriticRS	0.46	**0.50***	**0.51***	**0.52***	**0.52**

Table 4.2: Rewards obtained on the first, second, . . . , fifth result page of a session, when 5 items are displayed.

Method	$r_{mean}(p_1)$	$r_{mean}(p_2)$	$r_{mean}(p_3)$	$r_{mean}(p_4)$	$r_{mean}(p_5)$
Random	0.39	0.39	0.39	0.39	0.39
IRF	0.46	0.50	0.51	0.52	0.53
Greedy	**0.47**	0.49	0.50	0.51	0.51
MGD	**0.47**	0.50	0.51	0.53	0.53
ActorCriticRS	0.45	**0.51***	**0.53***	**0.54***	**0.55***

We expect that *Greedy* obtains the highest reward for the first result page. However, it learns user preferences very slowly, because it does not do enough exploration and displays very similar items on subsequent result pages. Therefore, it should work worse than RL-based methods for subsequent result pages in the session. *Greedy*, *MGD*, and ActorCriticRS have the expected behavior regardless of the number of items displayed on the result page. More precisely, for the first result page in the session ActorCriticRS and *MGD* perform a little worse than *Greedy*. For the other result pages shown in the session, ActorCriticRS and *MGD* outperform *Greedy*.

The results prove that for the task of query-free interactive target item retrieval for experienced users an RL-based approach is useful and outperforms traditional RS.

4.6.2 Answer to RQ3.2

To understand how the number of items displayed on a result page affects the performance, we turn to Tables 4.1, 4.2, 4.3, and 4.4. We compare the quality of the algorithms depending on the number of items displayed on the result page. More precisely, for each method we compare its results depending on the number of displayed items. *Random* obtains the same r_{mean} regardless of the number of displayed items. Other methods obtain larger rewards when the number of items displayed on the result page increases. This can be explained by the fact that when a user compares more items, the feedback obtained from her becomes more informative. Indeed, when a user examines the result page and clicks on the closest item to the target item, we get $n - 1$ preferences: the selected items is better than the others displayed on the result page. As the number of items displayed on the result page increases, the number of preferences received from

Table 4.3: Rewards obtained on the first, second, . . . , fifth result page of a session, when 7 items are displayed.

Method	$r_{mean}(p_1)$	$r_{mean}(p_2)$	$r_{mean}(p_3)$	$r_{mean}(p_4)$	$r_{mean}(p_5)$
Random	0.39	0.39	0.39	0.39	0.39
IRF	0.46	0.50	0.52	0.53	0.54
Greedy	**0.47**	0.49	0.50	0.51	0.52
MGD	0.46	0.50	0.52	0.54	**0.55**
ActorCriticRS	0.46	**0.51***	**0.53***	**0.55***	**0.55**

Table 4.4: Rewards obtained on the first, second, . . . , fifth result page of a session, when 10 items are displayed.s

Method	$r_{mean}(p_1)$	$r_{mean}(p_2)$	$r_{mean}(p_3)$	$r_{mean}(p_4)$	$r_{mean}(p_5)$
Random	0.39	0.39	0.39	0.39	0.39
IRF	0.46	0.51	0.52	0.53	0.54
Greedy	**0.47***	0.50	0.51	0.52	0.53
MGD	0.46	0.50	0.52	0.54	**0.56**
ActorCriticRS	0.45	**0.52***	**0.54***	**0.55***	**0.56**

showing a result page also increases.

4.6.3 Answer to RQ3.3

To understand how the way user satisfaction is measured affects the performance of algorithms, we turn to Tables 4.4 and 4.5. ActorCriticRS obtains the highest reward starting from the second result page of the session independently of the reward function. MGD and *IRF* obtain comparable r_{max}, but MGD obtains larger r_{mean} for later result pages in the session. This can be explained by the fact that during the training the cumulative reward of MGD uses r_{mean} as an immediate reward. *Greedy* obtains the largest r_{mean} for the first result page, but performs worse than the other methods for later result pages in the session. *Random* has the worst performance in all cases.

4.7 Related Work

The work related to this chapter comes in three groups: traditional recommender systems, reinforcement learning approaches to recommender systems, and methods that utilize relevance feedback.

4.7.1 Traditional recommender systems

CF is one of the most effective techniques to build personalized recommender systems [93]. CF-based methods collect user-item ratings and derive preference patterns. They

Table 4.5: Rewards obtained on the first, second, . . . , fifth result page of a session, when 10 items are displayed, using an alternative satisfaction function.

Method	$r_{max}(p_1)$	$r_{max}(p_2)$	$r_{max}(p_3)$	$r_{max}(p_4)$	$r_{mean}(p_5)$
Random	0.63	0.63	0.63	0.63	0.63
IRF	0.64	0.68	0.70	0.72	0.73
Greedy	0.61	0.64	0.65	0.67	0.68
MGD	0.64	0.68	0.70	0.72	0.73
ActorCriticRS	$\mathbf{0.65}$	$\mathbf{0.69}^*$	$\mathbf{0.72}^*$	$\mathbf{0.73}^*$	$\mathbf{0.74}$

do not use information about the sequence in which the ratings were given nor contextual information about the items. The main advantage of approaches based on CF is that they do not require domain knowledge and can easily be adapted to different recommender settings [83, 92]. Context-aware CF methods use both the contextual item features as well as ratings [44, 81]. Recently proposed CF-based methods use all the information available in the data: a sequence of the given ratings, ratings and product specific features [65, 73, 98]

Traditional recommender systems select items to display on the result page according to the relevance score of the items, thus the items on the result page are very similar. Unlike traditional recommender systems, the items displayed on the result page by ActorCriticRS satisfy the user's taste and they are diverse too.

4.7.2 Reinforcement learning for recommender systems

Methods that use RL for recommendation consider recommendation as a sequential process. Each episode consists of interactions between a user and a RS, ordered by time. In traditional Recommender System, items displayed on the page are predicted to satisfy users' preference the most, whereas in RL-based approaches the system maximizes long-term user satisfaction with recommendations.

Multi-armed bandits are a successful RL technique used for recommendation [3, 66, 117]. In the original bandit problem a gambler on a number of slot machines must decide which machines to play, how many times to play each machine, and in what order to play them [78]. Each machine gives a random reward from the probability distribution specific to that machine. The goal of the player is to maximize the reward received during the sequence of games. In bandit-based Recommender Systems, items play the role of machines. A common reward used in Recommender Systems is the click-through rate [66, 117].

Another RL-technique that is widely used for recommendation is Deep Reinforcement Learning (DeepRL) [69, 84, 139, 138, 140]. DeepRL does not estimate the transition probability and does not store Q-values in a table. Instead, a neural network estimates cumulative rewards (see Eq. 4.1). Thus, DeepRL chooses actions quickly and can be used in Recommender Systems where the action and state spaces are continuous and/or high-dimensional.

Differently from RL-based recommendations, ActorCriticRS utilizes a RF-mechanism

to obtain information from interactions between user and RS. RF allows us to obtain a lot of information from each interaction, and, thus, adapt to users' preferences fast.

4.7.3 Relevance feedback

The idea of RF is to involve a user in the search process so as to improve the recommended results. In RF, a user interacts with a system, marking some items as relevant, some as irrelevant, or noting that some items are more relevant than others [141]. Using users' feedback, the system revises its results and displays items that better satisfy users' tastes. RF is used when it is difficult to formulate a good query, but it is easy to judge a particular set of items. The main problem that arises in RF is which items to show. There are two main approaches to this problem. In the first, a user initiates a session with a query [8, 30, 31, 126, 131]. Then, all items and the query are embedded in a high-dimensional latent space. After each iteration, the vector representation of the query is refined, by bringing it closer to vector representations of relevant items and farther from irrelevant items. Finally, items that are closest to the new representation of the query are displayed. In the second approach, instead of estimating the representation of a query, the probability of items being relevant is modeled [4, 17, 114]. It is modeled as a conditional probability depending on the given relevance feedback. The displayed strategies in the two RF approaches are different. In the second approach, the algorithm selects the items to display in order to minimize the uncertainty or, in other words, the entropy of the target item.

Methods that use a RF mechanism are not aware of the history of user interaction with RS prior to the current session. That is why the first result pages shown by methods that use a RF mechanism are not personalized. As a consequence, these methods use several iterations to understand a user's long-term preferences. On the other hand, the method proposed in this chapter, ActorCriticRS, is aware of the history of a user's purchases. As a result, ActorCriticRS has a pretty accurate estimation of a user's long-term preferences from the start of a session, and during the session ActorCriticRS should only infer a user's current interests. This allows to make the session shorter.

4.8 Conclusion

In this chapter, we have considered the task of query-free interactive target item retrieval for experienced users. We do not assume that a user knows how to formulate a query. Consequently, she is unlikely to find an item that satisfies her preferences on conventional e-commerce sites where users interact by providing a query.

To solve the task of query-free interactive target item retrieval for experienced users we have proposed a three-part framework:

(1) an *item and user embedder*,

(2) a *state generator*, and

(3) a *recommender agent*.

The item and user embedder uses a Recommender System approach to infer users' long-term preferences using their purchase history. The state generator uses information received from a Relevance Feedback mechanism to generate low dimensional states that reflect users' preferences. The recommender agent selects items to display on the result page, given a state.

We have focused on the implementation of the recommender agent while using simple and efficient solutions for the other components. Specifically, we have proposed a model-free deep Reinforcement Learning-based algorithm, ActorCriticRS, that selects a set of items to display on the result page. We find an optimal strategy by maximizing the expected user satisfaction with the whole session. Thus, we can infer information about the target category, while simultaneously showing result pages that satisfy users' preferences.

We have compared ActorCriticRS with various approaches. A traditional RS-based approach suffices to generate the first result page in a session but not for subsequent pages of the session as user satisfaction hardly increases. In contrast, the first result pages generated using a RF-based approach contain items that do not satisfy user's preferences, but users like later result pages in the session. ActorCriticRS is able to show result pages that maximize user satisfaction with the entire session. In the future we want to investigate how different implementations of the components of the framework affect user satisfaction with a session.

In this chapter, we proposed to use a RL-based method to find an optimal strategy for selecting documents to display on a result page, and hereby we answered the RQ3. In the next chapter, we will study another aspect of preferences questionnaires: the interpretability of questions.

5 Evaluating Disentangled Representations

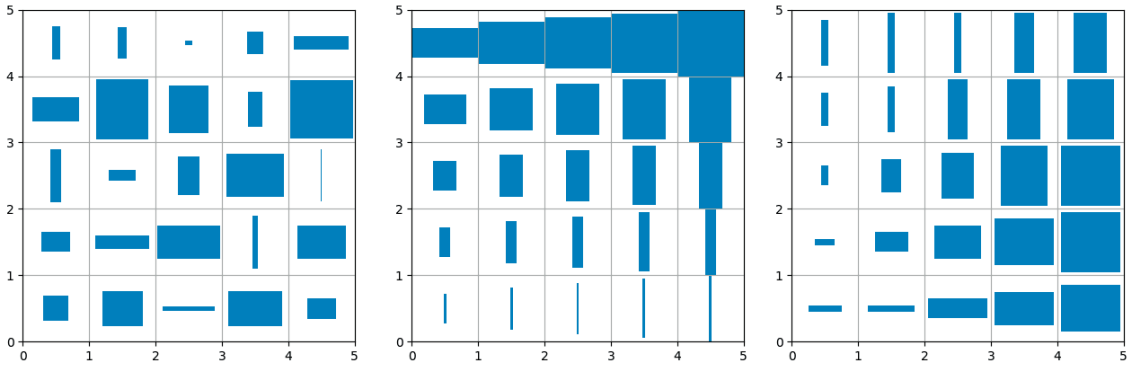
In this chapter, we study one more aspect of preferences questionnaires: the interpretability of questions. An interpretable and informative questionnaire can be created using disentangled representations, where a representation is called disentangled if it captures and separates factors of variations of data. In this chapter, we focus on the metric of disentangled representation and answer RQ4: how to measure the disentanglement of latent representations.

5.1 Introduction

Algorithms for learning representations are crucial for a variety of machine learning tasks, including image classification [47, 124] and image generation [37, 76]. These algorithms build a low-dimensional representation for each sample in a dataset, which is called a *latent representation*. Interpretable factors that describe every sample from the dataset are called *generative factors* of the dataset. While there is no single formalized notion of disentangled representation, the key intuition is that a disentangled representation should capture and separate the generative factors [5, 45]. For example, we show a dataset containing rectangles of different shapes (see Figure. 5.1a). There are two generative factors for this dataset: the length and width of the rectangles. In the disentangled latent representation of this dataset we can choose two latent factors. One of these factors is an invertible function of the length of the rectangles. Another factor is an invertible function of the width of the rectangles. In such a representation a change of one latent factor leads to a change only in one generative factor (see Figure. 5.1b). While in an entangled representation a change in one latent factor may lead to a change the length and width of the rectangles (see Figure. 5.1b).

Learning a disentangled representation is an important step towards better representation learning because a disentangled representation contains information about elements in a dataset in an interpretable and compact structure [5, 45]. Interpretability of the representation helps in tasks where users interact with a system, as they understand how it works and can provide informative feedback. Moreover, learning a disentangled representation helps for tasks where state-of-the-art machine learning-based approaches still struggle but where humans excel. Such scenarios include learning with knowledge

This chapter is under submission as [104].



(a) Example of a dataset containing rectangles with width w and height h from the uniform distribution $U(0, 1)$

(b) Change in the elements of the dataset caused by a change in the disentangled latent representation. Horizontally, the change is caused by a change in the first hidden factor; vertically, the change is caused by another hidden factor.

(c) Change in the elements of the dataset caused by a change in the entangled latent representation. Horizontally, the change is caused by a change in the first hidden factor; vertically, the change is caused by another hidden factor.

Figure 5.1: Example of a dataset and disentangled and entangled representations.

transfer [50, 88, 122], zero-shot inference [64, 96] and supervised learning [86, 116]. The possible reason why people successfully solve these tasks is that they have a mental model that captures explanatory factors about the world. These factors are generalized and used in a new way when a person solves a new task. Algorithms that can obtain the same representation of the samples in a collection as humans will have a similar ability to generalize.

Recently, several important steps have been taken towards the formal evaluation of disentangled representations. For example, there is a recent comparison of existing metrics of disentanglement through an experimental study on several datasets [74]. An important step is the intoduction of a framework for the evaluation of disentangled representations [28]. One more step is the creation of a formal definition of disentangled representation using group theory [45]. In this chapter we continue this important line of research by providing an analysis of theoretical properties of disentanglement metrics.

To summarize, our key technical contributions are:

- We review existing metrics of disentanglement and discuss their properties.
- We propose a new metric of disentanglement with theoretical guarantees, and establish its properties.[1]

[1] A comparison of methods for *learning* a disentangled representation is beyond the scope of this chapter.

5.2 Background

5.2.1 Representation learning

Usually, a representation learning algorithm consists of two parts: an encoder and a decoder. An encoder is a function:

$$f_e : \mathbb{R}^d \to \mathbb{R}^N,\ \mathbf{c} = f_e(\mathbf{x}), \tag{5.1}$$

where $\mathbf{c}$ is a latent representation of the data sample $\mathbf{x}$. Typically, the dimension of the latent representation is much smaller than the dimension of the data. A decoder is a function:

$$f_d : \mathbb{R}^N \to \mathbb{R}^d,\ f_d(f_e(\mathbf{x})) \sim \mathbf{x}, \tag{5.2}$$

where $f_d(f_e(\mathbf{x}))$ should be close to $\mathbf{x}$. Thus, the latent representation should contain almost all the information that is contained in the original data.

5.2.2 Ground truth generative factors

Generative factors of a dataset are interpretable factors that describe the difference between any two samples from X. Consider, for example, a dataset containing rectangles of different shapes presented, as illustrated in Figure. 5.1a. The generative factors for this dataset are the length and width of the rectangles. Formally, the definition of the generative factor can be formulated as follows:

$$\exists g : \mathbb{R}^K \to \mathbb{R}^d \text{ such that } \forall \mathbf{x} \in X \exists! \ \mathbf{z} : \mathbf{x} = g(\mathbf{z}), \tag{5.3}$$

where g is a generative process. The *ground truth* generative factors are generative factors that are given for a collection. This means that for each dataset sample $\mathbf{x} \in X$, the values of the generative factors $\mathbf{z} \in \mathbb{R}^K$ are known.

5.3 Metrics of Disentanglement of Representations

In this section, we analyze the properties of the metrics of disentangled representations proposed in [16, 28, 46, 58, 63]. To this end, we first repeat the definitions of disentanglement of representations that the proposed metrics reflect.

Definition 1 ([28, 46, 58])**.** A *disentangled representation* is a representation where a change in one latent dimension corresponds to a change in one generative factor while being relatively invariant to changes in other generative factors.

Definition 2 ([63, 74])**.** A *disentangled representation* is a representation where a change in a single generative factor leads to a change in a single factor in the learned representation.[2]

[2]This property of representations is also called *completeness* [28].

A representation is *entangled* if it is not disentangled. Below we analyze the existing disentanglement metrics using these definitions: for each metric, we use a definition of disentanglement depending on which definition it should reflect according to the authors of the metric. In particular, we analyze if the existing metrics satisfy the following properties:

Property 1. *A metric gives a high score to all almost perfectly disentangled representations.*

Property 2. *A metric gives a low score for all entangled representations.*

5.3.1 BetaVAE and FactorVAE

BetaVAE [46] and *FactorVAE* [58] are metrics that rely on the following property of disentangled representations:

Property 3. *If one generative factor is fixed, while other generative factors are arbitrarily changed, then in the corresponding disentangled representation one latent factor should vary less than the others.*

Definition of BetaVAE

The algorithm that calculates *BetaVAE* [46] consists of the following steps:

(1) Choose a generative factor z_k.

(2) Generate a batch of pairs of vectors for which the value of z_k within the pair is equal, while other generative factors are chosen randomly:

$$(\mathbf{p}_1 = \langle z_{1,1}, \ldots, z_{1,K} \rangle, \mathbf{p}_2 = \langle z_{2,1}, \ldots, z_{2,K} \rangle),\ z_{1,k} = z_{2,k}$$

(3) Calculate the latent code of the generated pairs: $(\mathbf{c}_1 = f_e(g(\mathbf{p}_1)), \mathbf{c}_2 = f_e(g(\mathbf{p}_2)))$

(4) Calculate the absolute value of the pairwise differences of these representations:

$$\mathbf{e} = \langle |c_{1,1} - c_{2,1}|, \ldots, |c_{1,N} - c_{2,N}| \rangle$$

(5) The mean of these differences across the examples in the batch gives one training point for the linear regressor that predicts which generative factor was fixed.

(6) BetaVAE is the accuracy of the linear regressor.

Definition of FactorVAE

The idea behind FactorVAE [58] is very similar to BetaVAE. The main difference between them concerns how a batch of examples is generated to obtain a variation of latent variables when one generative factor is fixed. Another difference is the classifier that predicts which generative factor was fixed using the variation of latent variables. *FactorVAE* can be calculated by performing the following steps:

(1) Choose a generative factor z_k.

(2) Generate a batch of vectors for which the value of z_k within the batch is fixed, while other generative factors are chosen randomly.

(3) Calculate latent codes of vectors from one batch.

(4) Normalize each dimension in the latent representation by its empirical standard deviation over the full data.

(5) Take the empirical variance in each dimension of these normalized representations.

(6) The index of the dimension with the lowest variance and the target index k provides one training point for the classifier.

(7) FactorVAE is the accuracy of the classifier.

Facts about BetaVAE and FactorVAE

Fact 1. BetaVAE *and* FactorVAE *do not satisfy Property 1 when the disentangled representation is defined using Definition 1.*

Proof. By Definition 1, in a perfectly disentangled representation there may be several latent factors that correspond to changes in the same generative factor. Consequently, these latent factors have variation 0 when the corresponding generative factor is fixed. That is why the classifier cannot distinguish between these latent factors and its accuracy is less than 1. Consequently, the BetaVAE and FactorVAE metrics return scores of less than 1 for a perfectly disentangled representation. □

Fact 2. BetaVAE *does not satisfy Property 2 when the disentangled representation is defined using Definition 1.*

Proof. As a proof, we give a counterexample. Let us consider all training points for a linear classifier with a fixed label. The classifier can learn to map some regularity in the values of features to the right class. However, there do not exist any constraints on this regularity. The classifier can learn to map samples with a value of 0 of some feature to the correct class. But the classifier can also learn to map samples with other patterns in the feature values to the correct class. Given this intuition, let us consider the following example. Suppose there are 3 generative factors from a uniform distribution and the dimension of the latent representation is 3. Assume that the latent variables are equal to the generative factors with the following probabilities:

$$p_1 = (0.5, 0.5, 0),\ p_2 = (0, 0.5, 0.5),\ p_3 = (0.5, 0, 0.5).$$

We generate 10,000 training points with a batch size of 128. The accuracy of the linear classifier is equal to 0.9967 in this case, but the latent representation is totally entangled. This shows that BetaVAE can assign high scores to entangled representations. □

Fact 3. FactorVAE *does not satisfy Property 2 when the disentangled representation is defined using Definition 1.*

Proof. First, let us analyze the algorithm that calculates the FactorVAE score. Suppose that for each generative factor z_j there is a latent variable c_{i_j} that correlates with z_j more than other variables in the latent code $\mathbf{c}$. In this case, when z_j is fixed and the batch size is large enough, the variation in the latent factor c_{i_j} will be smaller than the variation in other latent factors. Consequently, the classifier will have high accuracy. Given this intuition, let us consider the following example. Suppose there are 3 generative factors from a Gaussian distribution with $\mu = 0, \sigma = 1$, and each latent variable is a weighted sum of the generative factors:

$$c_1 = 0.5 \cdot z_1 + 0.4 \cdot z_2 + 0.5 \cdot z_3,$$
$$c_2 = 0.4 \cdot z_1 + 0.5 \cdot z_2 + 0.5 \cdot z_3,$$
$$c_3 = 0.4 \cdot z_1 + 0.4 \cdot z_2 + 0.6 \cdot z_3.$$

We generate 10,000 training points with a batch size of 128. The FactVAE disentanglement score is equal to 1 in this case, but the representation is entangled. This shows that FactoVAE can assign high scores to entangled representations. □

5.3.2 The DCI, MIG and SAP metrics

In this section, we describe metrics of disentangled representations that are calculated using a notion of informativeness between latent variables and generative factors.

DCI: Disentanglement, Completeness and Informativeness

Eastwood and Williams [28] propose to use a metric of disentangled representations, which we call DCI, that is calculated as follows:

(1) First, the *informativeness* between c_i and z_j is calculated. To determine the informativeness between c_i and z_j, Eastwood and Williams [28] suggest training K regressors. Each regressor f_j predicts z_j given $\mathbf{c}$ ($\hat{z_j} = f_j(\mathbf{c})$) and can provide an importance score $P_{i,j}$ for each c_i. The normalized importance score obtained by regressor f_j for variable c_i is used as the informativeness between c_i and z_j:

$$I_{i,j} = \frac{P_{i,j}}{\sum_{k=0}^{k=K} P_{i,k}}.$$

(2) For each latent variable its score of disentanglement is calculated as follows:

$$H_K(I_i) = 1 + \sum_{k=1}^{K} I_{i,k} \log_K I_{i,k}.$$

(3) The weighted sum of the obtained scores of disentanglement for the latent variables is DCI:

$$\text{DCI}(\mathbf{c}, \mathbf{z}) = \sum_i \left(\rho_i \cdot H_K(I_i)\right), \text{ where } \rho_i = \frac{\sum_j P_{i,j}}{\sum_{ij} P_{i,j}}. \tag{5.4}$$

MIG: Mutual Information Gap

Chen, Li, Grosse, and Duvenaud [16] propose a disentanglement metric, Mutual Information Gap (MIG), that uses mutual information between the j-th generative factor and the i-th latent variable as a notion of informativeness between them. The *mutual information* between two variables c and z is defined as

$$I(c; z) = H(z) - H(z|c),$$

where $H(z)$ is the entropy of the variable z. Mutual information measures how much knowing one variable reduces uncertainty about the other. A useful property of mutual information is that it is always non-negative $I(c; z) > 0$. Moreover, $I(c; z)$ is equal to 0 if and only if c and z are independent. Also, mutual information achieves its maximum if there exists an invertible relationship between c and z. The following algorithm calculates the MIG score:

(1) Compute a *matrix of informativeness* $I_{i,j}$, in which the ij-th entry is the mutual information between the j-th generative factor and the i-th latent variable.

(2) For each column of the score matrix $I_{i,j}$, which corresponds to a generative factor, calculate the difference between the top two entries, and normalize it by dividing by the entropy of the corresponding generative factor. The average of these normalized differences is the MIG score:

$$\text{MIG}(\mathbf{c}, \mathbf{z}) = \frac{1}{K} \sum_k \frac{I_{i_k,k} - \max_{l \neq i_k} I_{l,k}}{H(z_k)}, \; i_k = \arg\max_i I_{i,k}.$$

SAP: Separated Attribute Predictability

Kumar, Sattigeri, and Balakrishnan [63] provide a metric of disentanglement that is calculated as follows:

(1) Compute a *matrix of informativeness* $I_{i,j}$, in which the ij-th entry is the linear regression or classification score of predicting the j-th generative factor using only the i-th variable in the latent representation.

(2) For each column in the matrix of informativeness $I_{i,j}$, which corresponds to a generative factor, calculate the difference between the top two entries (corresponding to the top two most predictive latent factors). The average of these differences is the final score, which is called the SAP, for Separated Attribute Predictability:

$$\text{SAP}(\mathbf{c}, \mathbf{z}) = \frac{1}{K} \sum_k \left(I_{i_k,k} - \max_{l \neq i_k} I_{l,k} \right), \; i_k = \arg\max_i I_{i,k}.$$

Facts about DCI, MIG and SAP

Fact 4. *DCI does not satisfy Property 1 when the disentangled representation is defined using Definition 1.*

Proof. We argue that using entropy as a score of disentanglement of one latent variable is not correct. Indeed, a score of disentanglement of c_i should be high when c_i reflects one generative factor well, while it reflects other generative factors equally poorly. However, since the distribution may be close to uniform for these generative factors, the entropy is large. Let us provide an example that is built on this observation. Suppose there are 11 generative factors, and 11 is the dimension of the latent representation. Each latent factor c_i captures primarily a generative factor z_i:

$$P_{i,i} = 0.8,\ P_{i,k} = 0.02,\ k \neq i.$$

Then, the DCI score is 0.6, so the DCI assigns a small score to a disentangled representation. □

Fact 5. *MIG satisfies Property 1 when the disentangled representation is defined using Definition 2.*

Proof. Indeed, in a disentangled representation each generative factor is primarily captured in only one latent dimension. This means that for each generative factor z_j, there is exactly one latent factor c_{i_j} for which z_j is a function of c_{i_j}: $z_j \sim f(c_{i_j})$. Therefore,

$$I_{i_j,j} = H(z_j) - H(z_j|c_{i,j}) \sim H(z_j),$$

whereas for other latent variables $I_{k,j} = I(c_k, z_j) \sim 0$. Consequently, according to MIG, the score of disentanglement of each generation factor is close to 1:

$$\frac{I_{i_j,j} - \max_{k \neq i_j} I_{k,j}}{H(z_j)} \sim 1. \tag{5.5}$$

Therefore, the average of these scores is also close to 1. This shows that MIG score always assigns a high score to disentangled representations. □

Fact 6. *SAP does not satisfy Property 1 when the disentangled representation is defined using Definition 2.*

Proof. We think that it is incorrect to use the R^2 score of linear regression as informativeness between latent variables and generative factors. Indeed, a linear regression cannot capture non-linear dependencies. Thus, informativeness, which is calculated using the R^2 score of a linear regression, may be low if each generative factor is a non-linear function of some latent variable. Let us give an example that is built on this observation. Suppose there are 2 generative factors from the uniform distribution $U([-1, 1])$ and the dimension of the latent representation is 2. Let us assume the latent variables are obtained from the generative factors according to the following equations:

$$c_1 = z_1^{15},\ c_2 = z_2^{15}$$

For this perfectly disentangled representation, we generate 10,000 examples and obtain a SAP score equal to 0.32. This proves that SAP can assign a low score to disentangled representations. □

Fact 7. *DCI does not satisfy Property 2 when the disentangled representation is defined using Definition 1.*

Proof. We give a counterexample, which is built on the fact that the weighted sum in Eq. 5.4 can be large if only one latent variable is disentangled, while the other latent variables are entangled and do not capture any information about generative factors. Suppose there are 2 generative factors and the dimension of the latent representation is 2, and the matrix of informativeness is the following:

$$P_{0,0} = 1, P_{0,1} = 0, P_{1,1} = 0.09, P_{1,0} = 0.01.$$

In this case, the DCI score is 0.957. This counterexample shows that the DCI score can be close to 1 for entangled representations. □

Fact 8. *MIG satisfies Property 2 when the disentangled representation is defined using Definition 2.*

Proof. A high MIG score indicates that the majority of generative factors is captured in only one latent dimension. Consequently, a change in one of the generative factors entails a change primarily in only one latent dimension. □

Fact 9. *SAP does not satisfy Property 2 when disentangled representation is defined using Definition 2.*

Proof. A high SAP score indicates that the majority of generative factors is captured linearly in only one latent dimension. However, the SAP metric does not penalize the existence of several latent factors that capture the same generative factor non-linearly. Let us consider the following example. Suppose there are 2 generative factors from the uniform distribution $U([-1, 1])$, and the dimension of the latent representation is 3. Let us assume that the latent factors are obtained from the generative factors according to the following equations:

$$c_1 = z_1,\ c_2 = z_1^{25} + z_2^{25},\ c_3 = z_2.$$

For this latent representation, a change in each generative factor leads to a change in several latent factors, but the SAP score is equal to 0.98. This shows that the SAP score can be close to 1 for entangled representations. □

A summary of the results of our analysis is given in Table 5.1.

Table 5.1: Summary of facts about proposed metrics of disentangled representations.

Metric	Satisfies Property 1	Satisfies Property 2	Definition used
BetaVAE	No	No	Definition 1
FactorVAE	No	No	Definition 1
DCI	No	No	Definition 1
SAP	No	No	Definition 2
MIG	Yes	Yes	Definition 2

5.4 A New Metric of Disentanglement, DCIMIG

While previous metrics are either based on Definition 1 or on Definition 2, we believe that both of these definitions should be satisfied simultaneously for a disentangled representation. This means that we should consider the following definition of disentangled representation:

Definition 3. A *disentangled representation* is a representation satisfying the following properties. In a disentangled representation, we can choose a subset of latent variables: $c' = \{c_{i_1}, \ldots, c_{i_K}\}$, which satisfy Definition 1 and Definition 2. Moreover, a disentangled representation should contain nearly all information about generative factors, i.e., it should have a high degree of *informativeness* [28].

Based on Definition 3, we propose a new metric of disentanglement of representation, called DCIMIG. The previously introduced MIG metric is a good starting point because it is the only metric that gives high scores for disentangled representations and low scores for entangled representations when using Definition 2. However, the MIG metric has some drawbacks. The MIG metric does not penalize latent representations in which latent factors capture several generative factors. Consequently, the MIG metric can assign large scores to representations that are entangled according to Definition 1. Moreover, the MIG metric does not capture the informativeness of latent representation as it equally penalizes latent representations for not capturing informative generative factors and for not capturing non-informative generative factors. That is why we propose a new metric, *DCIMIG*, that captures the Disentanglement, Completeness (see footnote 2) and Informativeness of a representation using the Mutual Information Gap.

5.4.1 Definition of DCIMIG

Following MIG, we create a matrix of informativeness $I_{i,j}$, in which the ij-th entry is the mutual information between the j-th generative factor and the i-th variable in the latent representation. The following steps for calculating DCIMIG differ from the steps suggested in MIG:

(1) For each latent variable c_i, find the generative factor z_{j_i} that it reflects the most: $j_i = \arg\max_j I_{i,j}$.

(2) Calculate the disentanglement for each latent variable: $D_i = I_{i,j_i} - \max_{k \neq j_i} I_{i,k}$.

(3) For each generative factor z_j, find the most disentangled latent factor c_{k_j}, that reflects z_j: $k_j = \arg\max_{l \in \mathbb{I}_j} D_l$, where $\mathbb{I}_j = \{i : z_{j_i} = z_j\}$.

(4) For each generative factor z_j, calculate the disentanglement score D_j^z, which is equal to D_{k_j} if there is at least one latent factor, that captures z_j, otherwise, it is 0.

(5) Finally, the *disentanglement score* of a latent representation according to DCIMIG is the normalized sum of D_j^z:

$$\text{DCIMIG}(\mathbf{c}, \mathbf{z}) = \frac{\sum_{j=1}^{K} D_j^z}{\sum_{j=1}^{K} H(z_j)},$$

where $H(z_j)$ is the entropy of z_j.

5.4.2 Facts about DCIMIG

Fact 10. *DCIMIG satisfies Property 1 when the disentangled representation is defined using Definition 3.*

Proof. Indeed, in a disentangled representation according to Definition 3, there is a subset c' of latent variables, in which each latent variable is sensitive to changes in one generative factor only. Moreover, for each generative factor z_j there is only one latent variable $c_{i_j} \in c'$ that captures the changes in z_j. Consequently, c_{i_j} is a function of z_j: $c_{i_j} = f_j(z_j)$, while the other latent factors are invariant to changes in z_j. This means that, $D_{i_j} = I_{i_j,j} - \max_{k \neq j} I_{i_j,k} = I_{i_j,j}$. Also, the disentangled representation should have a high degree of *informativeness*. Consequently, the latent variables in c' should capture all the information contained in z_j. But only $c_{j,i}$ contains some information about z_j. Therefore, $I_{i_j,j} = H(z_j)$, and $D_j^z = H(z_j)$. Consequently, DCIMIG is equal to 1 in this case. □

Fact 11. *DCIMIG satisfies Property 2 when the disentangled representation is defined using Definition 3.*

Proof. When a representation is entangled for the majority of informative generative factors z', we cannot find a factor in the latent representation that reflects only this factor. There are 2 cases for the generative factors from z'. In the first case, there is no latent factor that captures the generative factor $z_j \in z'$. In that case, D_j^z is equal to 0. The second case is characterized by the fact that there is a latent factor that captures a generative factor, but this latent factor also captures other generative factors. In that case the disentangled score of this latent factor D_{i_j} is small, and consequently, D_j^z is small. □

5.4.3 Distinctions between DCIMIG, DCI and MIG

DCIMIGis similar to DCI and MIG, but has important distinctions from them. We describe the differences between these metrics below. First, we list the differences between DCIMIGand DCI:

(1) DCIMIGpenalizes a latent representation if there is a generative factor that is not captured by any latent variable, while DCI does not penalize such a representation.

(2) DCIMIGgives a high score to a representation in which, in addition to the entangled latent variables, there is a subset of the disentangled latent variables that capture all the generative factors. On the other hand, DCI penalizes representations that in addition to disentangled latent variables contain entangled latent variables.

The main differences between DCIMIGand MIG are:

(1) DCIMIGpenalizes representations in which all the latent factors capturing some generating factor are entangled. Our measure penalizes representations in which all hidden factors capturing some generative factor are entangled MIG score may be close to 1 in this case.

(2) DCIMIGgives a lower score to representations that do not capture informative generativr factors than to representations that do not capture non-informative generative factors. MIG does not distinguish between these representations.

To support the claim that DCIMIG, DCI and MIG give different scores we provide experimental results in Appendix A.

5.5 Related work

This chapter is relevant to two research directions: the formulation of a notion of disentangled representation and the analysis of differences between proposed metrics of disentangled representations.

A definition of disentangled representation is presented by Higgins, Amos, Pfau, Racaniere, Matthey, Rezende, and Lerchner [45], who proposed to call a representation disentangled if it is consistent with transformations that characterize the dataset. In particular, Higgins, Amos, Pfau, Racaniere, Matthey, Rezende, and Lerchner [45] suggested that transformations that change only some properties of elements in the dataset, while leaving other properties unchanged, given the structure of a dataset. Desirable properties of a disentanglement metric were formulated by Eastwood and Williams [28], namely disentanglement, completeness, and informativeness. Eastwood and Williams [28] claimed that a good representation should satisfy all of these properties, namely (1) if a representation is good, then change in one latent factor should lead to change in one generative factor, (2) a change in one generative factor should lead to a change in one latent factor, and (3) a latent representation should contain all information about the generative factors. Therefore, Eastwood and Williams [28] proposed three metrics to satisfy each of the properties listed. However, the proposed metrics were not analyzed — a gap that we fill.

Several papers analyze the differences between metrics of disentanglement through experimental studies [16, 74]. For example, Locatello, Bauer, Lucic, Gelly, Schölkopf, and Bachem [74] trained 12,000 models that cover the most prominent methods and evaluate these models using existing metrics of disentanglement. The study showed that the metrics are correlated, but the degree of correlation depends on the dataset. It is important to note that their experimental results are consistent with our theoretical findings: the BetaVAE [46] and FactorVAE [58] metrics are strongly correlated with each other; and the SAP [63], MIG [16], DCI [28] scores are also strongly correlated. Locatello, Bauer, Lucic, Gelly, Schölkopf, and Bachem [74] made an important step towards the evaluation of methods to create disentangled representations, however, the properties of the metrics were not analyzed theoretically. Chen, Li, Grosse, and Duvenaud [16] took a step in this direction, but only analyzed the BetaVAE, FactorVAE and MIG metrics. Chen, Li, Grosse, and Duvenaud [16] compared metrics by analyzing their robustness to the choice of the hyperparameters during experiments. The experimental findings are

quite similar to ours: BetaVAE is a very optimistic metric and assigns high scores to entangled representations.

To summarize, the key distinctions of our work compared to previous efforts are: (1) an in-depth analysis of previously proposed metrics of disentanglement, and (2) a proposal of a single metric of disentanglement that reflects all properties of previously proposed ones and has theoretical guarantees.

5.6 Conclusion

In recent years, several models have been developed to obtain disentangled representations [22, 49, 58, 135]. The importance of developing a reliable metric of disentanglement has been clearly stated by Kim and Mnih [58]. In this chapter, we analyzed whether existing metrics of disentanglement are close to 1 when a representation is disentangled and whether the metrics are close to 0 when a representation is entangled. As the definition of disentanglement varies from paper to paper, for our analysis we use definitions of disentanglement used by the authors of the corresponding metrics. Surprisingly, we found that most of the existing metrics can either give a low score to a disentangled representation or a high score to an entangled representation.

In some papers, disentangled representations are defined as representations in which a change in one generative factor should lead to a change in a single latent factor. While in other papers, disentangled representations are defined as representations in which a change in one latent factor should lead to a change in one generative factor. We argued that both properties should be satisfied by a disentangled representation. Based on this richer definition of a disentangled representation, we propose a new metric of disentanglement, DCIMIG, that captures the Disentanglement, Completeness and Informativeness of a representation using the Mutual Information Gap. We prove that DCIMIG assigns high scores to disentangled representation and low scores to entangled representations.

In future work, we plan to extend DCIMIG to the case where some generative factors form a subspace and a disentangled representation should align with these subspaces instead of single generative factors.

This chapter was the last research chapter in this thesis. We studied metrics of disentanglement of representations. Next, we will conclude the thesis and elaborate on the possible future work.

Appendix

A Experiments

We experimentally assess metrics by considering examples and understanding score differences. We will include outcomes of experiments using [74]'s library, with the dimensionality reduction method UMAP [82] on the cars3d dataset [91] (with different hyperparameter settings for the number of neighbors). DCIMIG, DCI and MIG give different results; see the tables below.

First, DCIMIG decreases when $UMAP_{100}$ is used instead of $UMAP_{20}$, but MIG, DCI scores increase (see left-most able). To see why, consider the informativeness matrices (IMs) of $UMAP_{20}$ and $UMAP_{100}$ (center and right-most tables). MIG increases because a latent variable of $UMAP_{100}$ starts capturing z_0: $I_{2,0} = 0.2$; this does not increase DCIMIG as c_2 captures other generative factors: $I_{2,1} = 0.8$. DCI increases because c_3 in $UMAP_{20}$ is entangled, but has a large weight; c_3 in $UMAP_{100}$ is still entangled but has lower weight. DCIMIG does not consider c_3 since c_3 captures mostly z_1, but the most disentangled variable that captures z_1 is c_0.

	MIG	DCI	DCIMIG
$UMAP_3$	0.005	0.033	0.015
$UMAP_4$	0.006	0.035	0.021
$UMAP_5$	0.016	0.031	0.041
$UMAP_{10}$	0.015	0.038	0.1
$UMAP_{20}$	0.017	0.07	0.12
$UMAP_{100}$	0.04	0.12	0.1

Disentanglement scores on car3D

	z_0	z_1	z_2
c_0	0.025	1.2	0.25
c_1	0.05	1.12	0.37
c_2	0.063	0.81	0.63
c_3	0.08	1.07	0.57
c_4	0.04	0.53	0.77
c_5	0.015	0.62	0.68

IM of $UMAP_{20}$

	z_0	z_1	z_2
c_0	0.09	1.11	0.41
c_1	0.03	1.04	0.32
c_2	0.2	0.8	0.29
c_3	0.1	0.71	0.64
c_4	0.028	0.47	0.73
c_5	0.01	0.85	0.85

IM of $UMAP_{100}$

6 Conclusions

In this chapter, we first revisit our research questions introduced in Chapter 1 and summarize the main findings and implications of our research in Section 6.1. Then, in Section 6.2, we describe the main limitations of our work and possible future directions.

6.1 Main Findings

6.1.1 Inferring long-term preferences from history of users' clicks

We started with the task of inferring user profiles using click signals from her history log and asked:

RQ1 How to infer a user's long-term preferences, using only their click interactions with the system?

To answer this question, we proposed a model that calculates the probability of a click on a document depending on the document position and the relevance of the document to the current query. But, differently from general click models, in our model, called personalized ranking of document and attractiveness (PRA), the probability of a click also depends on the preferences of users. Moreover, inspired by learning to rank algorithms, in contrast to click models, we do not optimize the log likelihood of a click, but explicitly fit the probability that one document is more relevant than another on the result page.

We empirically evaluated the proposed model for the personalized reranking, where the goal is to rerank 10 documents displayed on the result page, depending on the user's preference. We compared PRA with the state of the art methods for this task. We found that PRA achieves comparable or better results than the state of the art methods. To investigate the properties of PRA more deeply and eliminate bias from our evaluation, we also assessed the quality of PRA for a subset of the data in which the documents were seen by users with a high probability. The advantage of using PRA becomes even more obvious on this subset. Thus, our model achieves good results while being very simple and interpretable: for each 4-tuple (query, user, document, position of the document) it just calculates the probability that a user will explore the position of the document, the document's relevance to the query, and the user's preference for the document.

6.1.2 Inferring new users' preferences

We were interested in determining the preferences of new users for whom there is no interaction history and answered the second research question:

RQ2 How to create an informative preference questionnaire to infer cold users' preferences?

It is almost impossible to create an informative questionnaire without assumptions about the method that utilizes the answers the the questionnaire. That is why we focus only on one type of method, namely Collaborative Filtering (CF) based methods. We choose to answer the research question RQ2 for CF based methods for two reasons:

(1) they are widely used in recommendations due to their effectiveness and independence from the domain;

(2) and these methods utilize information about past interactions between users and a system and, therefore, the user cold start problem arises very sharply for them.

Many of the previous methods for producing a preference questionnaire use heuristics and are inefficient: many algorithms perform a brute force search of all possible preference questions. In contrast, we formalized the notion of informativeness of questionnaires and reformulated the task of finding informative questionnaires as solving an optimization problem. In addition to theoretical conclusions, we proposed an efficient algorithm that solves the optimization problem.

We empirically evaluated the proposed method by comparing it with state of the art methods for creating preference questionnaires. We found that the proposed method SPQ achieves the best performance. We concluded that in order to achieve good quality, a method must solve the correct optimization problem. In addition, we explored how a dataset affects the importance of choosing a questionnaire. We found two cases:

(1) datasets for which recommendations based on the popularity of items has good performance;

(2) datasets for which a non-personalized list of recommendations has far worse quality than a personalized one.

We concluded that for the first type of datasets if the questionnaire consists of a small number of questions the choice of elicitation procedure is not so important, however in the second case the method for creating questionnaires should be chosen carefully.

6.1.3 A reinforcement learning approach for inferring users' preferences

Our third research question was:

RQ3 How to find the optimal strategy for selecting documents to display, which helps to increase user satisfaction with an ongoing session, combining short-term and long-term preferences?

To answer this question, we used a CF based method to derive user's long term preferences from a log of their historical interactions. Different from standard recommender systems, we proposed to collect *relevance feedback* from users. That is, we ask users to specify their preferences for each shown page, choosing the best displayed document. One more distinction of the proposed method is the use of Reinforcement Learning (RL) to simultaneously select all documents for display on the result page, while most of the methods display items with the highest scores. Moreover, our RL-based approach optimizes long-term reward, which is users' satisfaction with the entire session. In addition, various standard methods have been developed to maximize some specific function that aims to approximate users' satisfaction. But the problem of understanding which functions correctly approximate users' satisfaction is still unsolved. The advantage of ActorCriticRS, the proposed method, is that in order to find an optimal strategy for a new approximation function of users' satisfaction, there is no need to design a new algorithm: only the reward function should be changed.

We empirically evaluated the proposed method, comparing it with state of the art methods that select items to recommend to users. In particular, we first evaluated it depending on the number of pages shown in the session. We conclude that ActorCriticRS works a little worse than the state of the art method for recommending items when a session consists of only one page, but for longer session ActorCriticRS outperforms other models. Then, we evaluated the performance of ActorCriticRS depending on the number of items shown on the page. We found that ActorCriticRS outperforms other methods regardless of the number of shown documents. Moreover, we showed that ActorCriticRS can adapt to various reward functions that are designed to reflect users' satisfaction with a session.

6.1.4 Understanding disentanglement of representations

Finally, we took a step towards interpretable, but informative questionnaires and explored a definition of a metric of disentangled representations, answering the following research question:

RQ4 How to measure the disentanglement of latent representations?

To answer this question, we explored the metrics proposed so far in the literature [16, 28, 46, 58, 63]. Most of the metrics were created to evaluate a new method of representation learning that aims to build disentangled representations. As a consequence, the metrics were not the focus of previously published papers and were proposed since there is no generally accepted metric of disentanglement. That is why the properties of these metrics were not analyzed in depth so far. However, some papers partially filled this gap in the understanding of a metric of disentanglement by extensive experiments. In contrast to previous work, we analyzed the theoretical properties of the metrics. In particular, we analyzed whether the metrics satisfy two properties:

(1) give a high score to all almost perfectly disentangled representations; and

(2) give a low score to all entangled representations.

We found that almost all metrics do not satisfy these basic properties. Moreover, the definition of disentanglement varies between different papers. We proposed a new metric of disentanglement and proved that these properties hold for this metric.

6.2 Future work

To conclude the thesis, we list possible directions for future work grouped by our main research questions.

6.2.1 Inferring long-term user preferences from history of users' clicks

PRA, the proposed approach for personalized reranking, covers only those queries that have been seen in the training data. However, most queries are tail queries, that is, queries that are unique and have been seen only once. An important direction is the understanding of similarity between queries.

Moreover, users change their behavior depending on the query: for some queries, users look at a lot of displayed documents, while for other queries, users look only at the top ranked results. For example, if a user is studying some new scientific area, she will probably examine more documents on the result page than when she is looking for some specific movie. We think that a good future direction is modeling position bias as a function of query embedding.

Also, it will be interesting to identify topics that reflect interests of users. This can be done by embedding the context of the document in a hidden space and deducing the interests of users from their clicks.

6.2.2 Constructing a questionnaire for inferring a user's preferences

To answer RQ2, we proposed a method SPQ, that has some drawbacks. The proposed method SPQ relies heavily on two assumptions:

(1) An MF based method is used for recommendation; and

(2) a latent representation of items has a good quality.

However, these assumptions do not always hold. Recently, many CF methods have been proposed that are built on a neural network [137]. These methods usually take into account a variety of signals, both behavioral and contextual. For example, using knowledge of users' reviews, descriptions of the items, images of items, categories of items, users purchases, submitted queries, etc., helps to improve the quality of recommendations. However, the theory developed behind SPQ is not applicable to deep learning methods since they use a non-linear function to predict the scores of items. An interesting future direction is to understand which questions about items provide the most information when the scoring function is nonlinear.

Moreover, SPQ selects items based only on their latent representation. Nevertheless, for some items, there are many ratings, but for others only a few. As a result, for

some items, their latent representations are reliable, while for others, they are not. The inference from the preferences of items for which the latent representation is unreliable and may be incorrect.

6.2.3 A reinforcement learning approach for inferring a user's preferences

In Chapter 4 we studied the task of target item retrieval. For finding a target item we proposed a framework consisting of three components:

(1) an *item and user embedder*;

(2) a *state generator*, and

(3) a *recommender agent*.

We chose the simplest implementation for each component, except for the recommender agent. In future work, we plan to explore possible implementations of each component. In particular, the component called *item and user embedder*, embeds items and users into latent space reflecting users' long term preferences. As an implementation of this component, we used the *VBPR* model [44]. However, VBPR is quite simple, the method does not take into account relations between items. For example, knowledge that items were purchased simultaneously can improve the quality of recommendations. Also, VBPR only uses images as contextual features, however, several other contextual features can be very useful.

Another component, called *state generator*, changes the state of the system depending on user's feedback. As a state generator, we chose a simple model that changes the state linearly. We can choose as a state generator a different model. One option is to combine a state generator with a recommendation agent in one function and train them jointly.

Another direction of future work is to understand how the system works when the answers of users are noisy. In our experimental setup, users always answer correctly: they never make mistakes and choose the item closest to the target item. But this assumption is not always true, especially when displayed items are similar.

Moreover, the recommendation agent in the ActorCriticRS approach is not aware of the number of a user's purchases in the past, and how diverse her choices were. But this information can be very useful. For example, if a user used the system many times and only bought a specific type of jeans, then she likely wants to buy jeans again. On the other hand, if a user used the system only a few times and bought jeans, a dress, and tights, then at the beginning of a new session the system does not know what the user wants to buy now. For the user in the first example, the items shown should not be so diverse, while for the user in the second example the diversity of items is a necessary condition for effective search. One of the possible research directions in the future is to study the balance between exploitation and exploration of a recommender agent, depending on a user's historical log.

6.2.4 Disentangled representations

In Chapter 5 we studied properties of the metrics of disentangled representations and proposed a new metric of disentangled representations. However, the proposed metric has some limitations. In particular, the proposed metric gives intuitively correct scores only in the case when the ground truth factors of variation are known, and it is important to have a one-to-one correspondence between latent variables and generative factors. First, in real life, the ground truth factors of variation are not given, but we can infer whether the representation is disentangled by looking at its properties. For example, suppose we ask people to indicate which of the shown items is the most similar to the target one, and the shown products have the same latent representations, except for the value of one latent variable. Then the values of this latent variable of the target product and the selected should be equal.

Moreover, the proposed metric penalizes a representation if there is no one-to-one mapping between the latent representation and factors of variation. However, in some cases, factors of variation form a subspace. For example, a variation factor such as "color" has three dimensions. A metric should not penalize a latent representation that contains latent factors that correspond to "purple, red, green" instead of "blue, red, green." Consequently, a future direction of defining a metric of disentanglement is to create a metric that provides intuitive results in the case when there is no need to have a one-to-one mapping.

Bibliography

[1] Xavier Amatriain, Josep M Pujol, and Nuria Oliver. “I like it... I like it not: Evaluating user ratings noise in recommender systems”. In: *International Conference on User Modeling, Adaptation, and Personalization*. Springer. 2009, pp. 247–258.

[2] Oren Anava, Shahar Golan, Nadav Golbandi, Zohar Karnin, Ronny Lempel, Oleg Rokhlenko, and Oren Somekh. “Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design”. In: *Proceedings of the 24th International Conference on World Wide Web*. ACM. 2015, pp. 45–54.

[3] Avinash Balakrishnan, Djallel Bouneffouf, Nicholas Mattei, and Francesca Rossi. “Using Contextual Bandits with Behavioral Constraints for Constrained Online Movie Recommendation.” In: *IJCAI*. 2018, pp. 5802–5804.

[4] Björn Barz, Christoph Käding, and Joachim Denzler. “Information-Theoretic Active Learning for Content-Based Image Retrieval”. In: *arXiv preprint arXiv:1809.02337* (2018).

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828.

[6] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. “Modeling the Impact of Short- and Long-Term Behavior on Search Personalization”. In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 2012, pp. 185–194.

[7] Lucas Bernardi, Jaap Kamps, Julia Kiseleva, and Melanie J. I. Müller. “The continuous cold start problem in e-commerce recommender systems”. In: *Proceedings of the Workshop on New Trends on Content-Based Recommender Systems co-located with ACM Conference on Recommender Systems*. 2015, pp. 30–33.

[8] Keping Bi, Qingyao Ai, and W Bruce Croft. “Revisiting Iterative Relevance Feedback for Document and Passage Retrieval”. In: *arXiv preprint arXiv:1812.05731* (2018).

[9] Andrei Broder. “A taxonomy of web search”. In: *Proceedings of the 25th international ACM SIGIR conference on Research and development in information retrieval* 36.2 (2002), pp. 3–10.

[10] Peter Brusilovski, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web: methods and strategies of web personalization*. Vol. 4321. Springer Science & Business Media, 2007.

[11] Christopher JC Burges. “From RankNet to LambdaRank to LambdaMART: An overview”. In: *Learning* 11.23-581 (2010), p. 81.

[12] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. “Learning to rank using gradient descent”. In: *ICML*. 2005, pp. 89–96.

[13] Fei Cai, Shangsong Liang, and Maarten de Rijke. “Personalized document re-ranking based on Bayesian probabilistic matrix factorization”. In: *Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2014.

[14] Fei Cai, Shuaiqiang Wang, and Maarten de Rijke. “Behavior-based personalization in web search”. In: *J. Assoc. for Inform. Sci. and Techn.* (2017). To appear.

[15] O. Chapelle and Y. Zhang. “A Dynamic Bayesian Network Click Model for Web Search Ranking”. In: *Proceedings of the 18th international conference on World wide web*. 2009, pp. 1–10.

[16] Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. “Isolating Sources of Disentanglement in Variational Autoencoders”. In: *arXiv preprint arXiv:1802.04942* (2018).

[17] Zhuoxiang Chen, Zhe Xu, Ya Zhang, and Xiao Gu. “Query-Free Clothing Retrieval via Implicit Relevance Feedback”. In: *IEEE Transactions on Multimedia* 20.8 (2018), pp. 2126–2137.

[18] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. “Towards conversational recommender systems”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 815–824.

[19] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. “Click models for web search”. In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7.3 (2015), pp. 1–115.

[20] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. “An experimental comparison of click position-bias models”. In: *Proceedings of the 2008 international conference on web search and data mining*. ACM. 2008, pp. 87–94.

[21] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. “Google news personalization: scalable online collaborative filtering”. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 271–280.

[22] Emily L Denton et al. “Unsupervised learning of disentangled representations from video”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4414–4423.

[23] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. “Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 193–202.

[24] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *International conference on machine learning*. 2014, pp. 647–655.

[25] Qiang Du, Vance Faber, and Max Gunzburger. “Centroidal Voronoi tessellations: Applications and algorithms”. In: *SIAM review* 41.4 (1999), pp. 637–676.

[26] G. Dupret and B. Piwowarski. “User browsing model to predict search engine click data from past observations”. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 2008, pp. 331–338.

[27] *E-Stats: Measuring the electronic economy*. https://www.census.gov/econ/estats/. 2015.

[28] Cian Eastwood and Christopher KI Williams. “A framework for the quantitative evaluation of disentangled representations”. In: *ICLR*. 2018.

[29] Ömer Eğecioğlu and Bahman Kalantari. “Approximating the diameter of a set of points in the Euclidean space”. In: *Information Processing Letters* 32.4 (1989), pp. 205–211.

[30] Bahaeddin Eravci and Hakan Ferhatosmanoglu. “Diverse relevance feedback for time series with autoencoder based summarizations”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.12 (2018), pp. 2298–2311.

[31] Marin Ferecatu and Donald Geman. “A statistical framework for image category search from a mental picture”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.6 (2009), pp. 1087–1101.

[32] Daniele V. Finocchiaro and Marco Pellegrini. “On computing the diameter of a point set in high dimensional Euclidean space”. In: *Theoretical Computer Science* 287.2 (2002), pp. 501–514.

[33] Alexander Fonarev, Alexander Mikhalev, Pavel Serdyukov, Gleb Gusev, and Ivan Oseledets. *Efficient rectangular maximal-volume algorithm for rating elicitation in collaborative filtering*. arXiv preprint arXiv:1610.04850. 2016.

[34] Jerome H. Friedman, Ron Kohavi, and Yeogirl Yun. “Lazy decision trees”. In: *AAAI/IAAI, Vol. 1*. 1996, pp. 717–724.

[35] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. “Adaptive bootstrapping of recommender systems using decision trees”. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM. 2011, pp. 595–604.

[36] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. “On bootstrapping recommender systems”. In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 1805–1808.

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[38] Sergei A. Goreinov, Ivan V. Oseledets, Dimitry V. Savostyanov, Eugene E. Tyrtyshnikov, and Nikolay L. Zamarashkin. “How to find a good submatrix”. In: *Matrix Methods: Theory, Algorithms, Applications*. Ed. by Vadim Olshevsky and Eugene E. Tyrtyshnikov. Hackensack, NY: World Scientific, 2010, pp. 247–256.

[39] Artem Grotov, Aleksandr Chuklin, Markov. Ilya, Luka Stout, Finde Xumara, and Maarten de Rijke. “A comparative study of click models for web search”. In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer International Publishing, 2015.

[40] Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, Gerald Tesauro, and Rogerio Feris. “Dialog-based interactive image retrieval”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 678–688.

[41] Jutta Haider and Olof Sundin. *Invisible Search and Online Search Engines: The ubiquity of search in everyday life*. Routledge, 2019.

[42] F. Maxwell Harper and Joseph A. Konstan. “The movielens datasets: History and context”. In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (2016), p. 19.

[43] Milos Hauskrecht. “Incremental methods for computing bounds in partially observable Markov decision processes”. In: *AAAI/IAAI*. Citeseer. 1997, pp. 734–739.

[44] Ruining He and Julian McAuley. “VBPR: visual bayesian personalized ranking from implicit feedback”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[45] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. “Towards a Definition of Disentangled Representations”. In: *arXiv preprint arXiv:1812.02230* (2018).

[46] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. “beta-vae: Learning basic visual concepts with a constrained variational framework”. In: *ICLR*. 2017.

[47] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[48] Fangwei Hu and Yong Yu. “Interview process learning for top-n recommendation”. In: *Proceedings of the 7th ACM conference on Recommender systems*. ACM. 2013, pp. 331–334.

[49] Qiyang Hu, Attila Szabó, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. “Disentangling Factors of Variation by Mixing Them”. In: *arXiv preprint arXiv:1711.07410* (2017).

[50] De-An Huang and Yu-Chiang Frank Wang. “Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2496–2503.

[51] Bernard J Jansen and Amanda Spink. “How are we searching the World Wide Web? A comparison of nine search engine transaction logs”. In: *Information processing & management* 42.1 (2006), pp. 248–263.

[52] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. “Comparison of implicit and explicit feedback from an online music recommendation service”. In: *Proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*. ACM. 2010, pp. 47–51.

[53] Wei Jiang, Damien Jose, and Gargi Ghosh. “User-SERP Interaction Prediction through Deep Multi-task Learning”. In: (2018).

[54] T. Joachims. “Evaluating retrieval peformance using clickthrough data”. In: *Text Mining*. 2003.

[55] T. Joachims. “Optimizing search engines using clickthrough data”. In: *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 133–142.

[56] Nicolas Jones, Armelle Brun, and Anne Boyer. “Comparisons instead of ratings: Towards more stable preferences”. In: *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. Vol. 1. IEEE. 2011, pp. 451–456.

[57] Saikishore Kalloori, Francesco Ricci, and Marko Tkalcic. “Pairwise preferences based matrix factorization and nearest neighbor recommendation techniques”. In: *Proceedings of the 10th ACM conference on Recommender systems*. ACM. 2016, pp. 143–146.

[58] Hyunjik Kim and Andriy Mnih. “Disentangling by factorising”. In: *arXiv preprint arXiv:1802.05983* (2018).

[59] Julia Kiseleva, Alexander Tuzhilin, Jaap Kamps, Melanie J. I. Muller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, and Djoerd Hiemstra. “Beyond movie recommendations: Solving the continuous cold start problem in e-commerce recommendations”. In: *CoRR* 1607.07904 (2016).

[60] Michal Kompan, Ondrej Kassak, and Maria Bielikova. “Beyond User Preferences: The Short-Term Behaviour Modelling.” In: *RecTemp@ RecSys*. 2017, pp. 1–3.

[61] Yehuda Koren. “Collaborative filtering with temporal dynamics”. In: *Communications of the ACM* 53.4 (2010), pp. 89–97.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[63] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. “Variational inference of disentangled latent concepts from unlabeled observations”. In: *arXiv preprint arXiv:1711.00848* (2017).

[64] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. “Learning to detect unseen object classes by between-class attribute transfer”. In: *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*. 2009, pp. 951–958.

[65] Jing Li, Wentao Xu, Wenbo Wan, and Jiande Sun. “Movie recommendation based on bridging movie feature and user interest”. In: *Journal of computational science* 26 (2018), pp. 128–134.

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

[67] Seth Siyuan Li and Elena Karahanna. “Online recommendation systems in a B2C E-commerce context: a review and future directions”. In: *Journal of the Association for Information Systems* 16.2 (2015), p. 72.

[68] Xiaodan Liang, Liang Lin, Wei Yang, Ping Luo, Junshi Huang, and Shuicheng Yan. “Clothes co-parsing via joint image segmentation and labeling with application to clothing retrieval”. In: *IEEE Transactions on Multimedia* 18.6 (2016), pp. 1175–1186.

[69] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[70] C. Liu, F. Guo, and C. Faloutsos. “BBM: Bayesian Browsing Model from Petabyte-scale Data”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 537–546.

[71] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. “Deep sketch hashing: Fast free-hand sketch-based image retrieval”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2862–2871.

[72] Nathan N. Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. “Wisdom of the better few: cold start recommendation via representative based rating elicitation”. In: *Proceedings of the 5th ACM Conference on Recommender Systems*. ACM. 2011, pp. 37–44.

[73] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. “Context-aware sequential recommendation”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 1053–1058.

[74] Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. “Challenging common assumptions in the unsupervised learning of disentangled representations”. In: *arXiv preprint arXiv:1811.12359* (2018).

[75] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. “Recommender system application developments: a survey”. In: *Decision Support Systems* 74 (2015), pp. 12–32.

[76] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. “Adversarial autoencoders”. In: *arXiv preprint arXiv:1511.05644* (2015).

[77] Jiaxin Mao, Cheng Luo, Min Zhang, and Shaoping Ma. “Constructing Click Models for Mobile Search”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM. 2018, pp. 775–784.

[78] Jérémie Mary, Romaric Gaudel, and Philippe Preux. “Bandits and recommender systems”. In: *International Workshop on Machine Learning, Optimization and Big Data*. Springer. 2015, pp. 325–336.

[79] P. Masurel, K. Lefévre-Hasegawa, C. Bourguignat, and M. Scordia. “Dataiku’s solution to Yandex’s personalized web search challenge”. In: *WSDM ’14 workshop on web search click data*. 2014.

[80] Nicolaas Matthijs and Filip Radlinski. “Personalizing Web Search using Long Term Browsing History”. In: *Proceedings of the 4th ACM international conference on Web search and data mining*. 2011, pp. 25–34.

[81] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. “Image-based recommendations on styles and substitutes”. In: *Proceedings of the 38th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2015, pp. 43–52.

[82] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[83] Andriy Mnih and Ruslan R Salakhutdinov. “Probabilistic matrix factorization”. In: *Advances in neural information processing systems*. 2008, pp. 1257–1264.

[84] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[85] Harvey Morgan, Hauff Claudia, and Elsweiler David. “Learning by Example: Training Users with High-quality Query Suggestions”. In: *Proceedings of the 29th international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2006.

[86] Anh Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.

[87] Quang Nhat Nguyen and Francesco Ricci. “Long-term and session-specific user preferences in a mobile recommender system”. In: *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM. 2008, pp. 381–384.

[88] Sinno Jialin Pan, Qiang Yang, et al. “A survey on transfer learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

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

[90] Al Mamunur Rashid, George Karypis, and John Riedl. “Learning preferences of new users in recommender systems: an information theoretic approach”. In: *ACM SIGKDD Explorations Newsletter* 10.2 (2008), pp. 90–100.

[91] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. “Deep visual analogy-making”. In: *Advances in neural information processing systems*. 2015, pp. 1252–1260.

[92] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. “BPR: Bayesian personalized ranking from implicit feedback”. In: *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press. 2009, pp. 452–461.

[93] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[94] Joseph John Rocchio. “Relevance feedback in information retrieval”. In: *The SMART retrieval system: experiments in automatic document processing* (1971), pp. 313–323.

[95] Lior Rokach and Slava Kisilevich. “Initial profile generation in recommender systems using pairwise comparison”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 1854–1859.

[96] Bernardino Romera-Paredes and Philip Torr. “An embarrassingly simple approach to zero-shot learning”. In: *International Conference on Machine Learning*. 2015, pp. 2152–2161.

[97] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. “Active learning in recommender systems”. In: *Recommender systems handbook*. Springer, 2015, pp. 809–846.

[98] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. “Sequential Variational Autoencoders for Collaborative Filtering”. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM. 2019, pp. 600–608.

[99] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. “Collaborative filtering recommender systems”. In: *The Adaptive Web*. Springer, 2007, pp. 291–324.

[100] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. “Music recommender systems”. In: *Recommender systems handbook*. Springer, 2015, pp. 453–492.

[101] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. “Multileave gradient descent for fast online learning to rank”. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM. 2016, pp. 457–466.

[102] Hans Schwerdtfeger. *Introduction to Linear Algebra and the Theory of Matrices*. Noordhoff, 1950.

[103] Anna Sepliarskaia, Sahika Genc, and Maarten de Rijke. “A Deep Reinforcement Learning-Based Approach to Query-Free Interactive Target Item Retrieval”. In: *IEEE Transactions on Multimedia* (2019). Under review.

[104] Anna Sepliarskaia, Kiseleva Julia, and Maarten de Rijke. “Evaluating Disentangled Representations”. In: *Advances in Neural Information Processing Systems*. Under review. 2019.

[105] Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. “Preference Elicitation as an Optimization Problem”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM. 2018, pp. 172–180.

[106] Anna Sepliarskaia, Filip Radlinski, and Maarten de Rijke. “Simple Personalized Search Based on Long-term Behavioral Signals”. In: *European Conference on Information Retrieval*. Springer, 2017, pp. 95–107.

[107] Amit Sharma and Baoshi Yan. “Pairwise learning in recommendation: experiments with community recommendation on linkedin”. In: *Proceedings of the 7th ACM conference on Recommender systems*. ACM. 2013, pp. 193–200.

[108] Si Shen, Botao Hu, Weizhu Chen, and Qiang Yang. “Personalized click model through collaborative filtering”. In: *Proceedings of the 5th ACM international conference on Web search and data mining*. 2012, pp. 323–332.

[109] Milad Shokouhi, Ryen W. White, Paul Bennett, and Filip Radlinski. “Fighting search engine amnesia: reranking repeated results”. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 2013, pp. 273–282.

[110] Shahab Saquib Sohail, Jamshed Siddiqui, and Rashid Ali. “Book recommendation system using opinion mining technique”. In: *2013 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2013, pp. 1609–1614.

[111] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. “A taxonomy of queries for e-commerce search”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM. 2018, pp. 1245–1248.

[112] G. Song. “Point-Wise Approach for Yandex Personalized Web Search Challenge”. In: *WSDM ’14 workshop on web search click data*. 2014.

[113] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. “User intent, behaviour, and perceived satisfaction in product search”. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM. 2018, pp. 547–555.

[114] Nicolae Suditu and Francois Fleuret. “HEAT: Iterative relevance feedback with one million images”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2118–2125.

[115] Nicolae Suditu and François Fleuret. “Iterative relevance feedback with adaptive exploration/exploitation trade-off”. In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM. 2012, pp. 1323–1331.

[116] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

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

[118] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael Potts. “Information re-retrieval: repeat queries in Yahoo’s logs.” In: *Proceedings of the 30th international ACM SIGIR conference on Research and development in information retrieval*. 2007, pp. 151–158.

[119] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. “To personalize or not to personalize: modeling queries with variation in user intent”. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieva*. 2008, pp. 163–170.

[120] Jaime Teevan, Dan Liebling, and Gayathri Ravichandran Geetha. “Understanding and Predicting Personal Navigation”. In: *Proceedings of the 4th ACM international conference on Web search and data mining*. 2011, pp. 85–94.

[121] Chun-Yuen Teng, Yu-Ru Lin, and Lada A Adamic. “Recipe recommendation using ingredient networks”. In: *Proceedings of the 4th Annual ACM Web Science Conference*. ACM. 2012, pp. 298–307.

[122] Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. “Safety in numbers: Learning categories from few examples with multi model knowledge transfer”. In: *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*. 2010, pp. 3081–3088.

[123] Lyle H. Ungar and Dean P. Foster. “Clustering methods for collaborative filtering”. In: *AAAI Workshop on Recommendation Systems*. AAAI, 1998, pp. 114–129.

[124] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th International Conference on Machine Learning*. ACM. 2008, pp. 1096–1103.

[125] M. Volkovs. “Context Models For Web Search Personalization”. In: *WSDM ’14 workshop on web search click data*. 2014.

[126] Angadpreet Walia, Tanisha Gahlawat, Parul Kalra, and Deepti Mehrotra. “An Emperical Study: Relevance Feedback in Information Retrieval Systems”. In: *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*. IEEE. 2017, pp. 982–985.

[127] Xianwang Wang, Tong Zhang, Daniel R Tretter, and Qian Lin. “Personal clothing retrieval on photo collections by color and attributes”. In: *IEEE Transactions on Multimedia* 15.8 (2013), pp. 2035–2045.

[128] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[129] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. “Temporal recommendation on graphs via long-and short-term preference fusion”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 723–732.

[130] Qianli Xing, Yiqun Liu, Jian-Yun Nie, Min Zhang, Shaoping Ma, and Kuo Zhang. “Incorporating user preferences into click models”. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM. 2013, pp. 1301–1310.

[131] Heng Xu, Jun-yi Wang, and Lei Mao. “Relevance feedback for Content-based Image Retrieval using deep learning”. In: *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE. 2017, pp. 629–633.

[132] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. “Deep Matrix Factorization Models for Recommender Systems.” In: *IJCAI*. 2017, pp. 3203–3209.

[133] Yandex. *Personalized web search challenge*. `http://www.kaggle.com/c/yandex-personalized-web-search-challenge/details/prizes`. 2013.

[134] Emine Yilmaz, Manisha Verma, Nick Craswell, Filip Radlinski, and Peter Bailey. “Relevance and Effort: An Analysis of Document Utility”. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 2014, pp. 91–100.

[135] Aron Yu and Kristen Grauman. “Semantic Jitter: Dense Supervision for Visual Comparisons via Synthetic Images”. In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE. 2017, pp. 5571–5580.

[136] Yisong Yue and Thorsten Joachims. “Interactively optimizing information retrieval systems as a dueling bandits problem”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 1201–1208.

[137] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. “Deep learning based recommender system: A survey and new perspectives”. In: *ACM Computing Surveys (CSUR)* 52.1 (2019), p. 5.

[138] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. “Deep reinforcement learning for page-wise recommendations”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM. 2018, pp. 95–103.

[139] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. “Deep reinforcement learning for list-wise recommendations”. In: *arXiv preprint arXiv:1801.00209* (2017).

[140] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. “Drn: A deep reinforcement learning framework for news recommendation”. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2018, pp. 167–176.

[141] Xiang Sean Zhou and Thomas S Huang. “Relevance feedback in image retrieval: A comprehensive review”. In: *Multimedia systems* 8.6 (2003), pp. 536–544.

Summary

People use search engines and recommender systems for various information needs every day. The task that these systems solve is to provide users with the right information. This means that these systems must show information that is relevant to the submitted query, is up-to-date and satisfies a user's preferences. In this thesis, we focus on the last aspect: personalization. The common way to make a result personalized consists of two steps: 1. infer a user's profile; and 2. provide information that is relevant to people with this profile. There are two types of signal about users that are used to generate their profile: explicit and implicit. To collect explicit signals, users are encouraged to take the initiative and explicitly provide information about their preferences. Implicit signals are signals from user interaction with the system. Moreover, a user profile consists of two components: 1. long-term user preferences; and 2. short-term user preferences. Long-term preferences are preferences that represent lasting user interests. Short-term preferences are preferences that are present only in the current session.

In this thesis, we investigate different ways of inferring user profiles using different types of signal: explicit and implicit. We start with analyzing how to infer a user's preferences, using implicit signals from all sessions to perform personalized re-ranking of a non-personalized list of documents. We propose a model in which the probability of a click on a document depends on three components: 1. the position of the document on the result page; 2. the relevance of the document to the query; and 3. the user's preferences. We show that this model obtains the best or equal results with the state-of-the-art models that perform personalized re-ranking of a non-personalized list of documents.

We continue our research by studying how to create an informative questionnaire to gather explicit feedback and infer a user's preferences. In particular, we analyze how this problem can be formulated and propose an optimization problem as a mathematical formalization. Also, we propose an algorithm that solves the formulated optimization problem and creates an informative questionnaire. We show that using answers from the created questionnaire we can infer a more accurate user profile than using answers from previously proposed questionnaires, which consist of questions of relative preferences.

Next, we improve a list of recommendations in the current session by combining implicit signals from the users' historical logs and explicit signals from the current session. In particular, we formulate the recommendation process in the current session as a game in which a user interacts with a recommender agent by clicking on the best document from the displayed documents; and a recommender agent tries to maximize the user's satisfaction with the session. We show how to apply Reinforcement Learning methods for this task and overcome the difficulty of a large number of actions that can be performed by a recommender agent. In particular, we use a model-free Deep Reinforcement Learning algorithm that is based on ActorCritic framework.

Finally, we investigate how to create interpretable questionnaires. We think that disentangled representations can be useful for creating interpretable questionnaires. But in order to choose the best disentangled representation, there must be a metric of disentanglement. We study the available metrics of disentangled representation and conclude that most of the current metrics may either give a low score to disentangled representations or give a high score to entangled representations. We propose a new

metric of disentanglement, which has the desired behavior: it gives high scores to all disentangled representations and low scores to all entangled representations.

Samenvatting

Mensen gebruiken dagelijks zoekmachines en aanbevelingssystemen voor verschillende informatiebehoeften. De taak die deze systemen oplossen is om gebruikers van de juiste informatie te voorzien. Dit betekent dat deze systemen informatie moeten tonen die relevant is voor de ingediende zoekopdracht, up-to-date is en voldoet aan de voorkeuren van een gebruiker. In dit proefschrift richten we ons op het laatste aspect: personalisatie. De gebruikelijke manier om een resultaat gepersonaliseerd te maken, bestaat uit twee stappen: 1. leid een gebruikersprofiel af; en 2. geef informatie die relevant is voor mensen met dit profiel. Er zijn twee soorten signalen over gebruikers die worden gebruikt om hun profiel te genereren: expliciet en impliciet. Om expliciete signalen te verzamelen, worden gebruikers aangemoedigd om het initiatief te nemen en expliciet informatie te verstrekken over hun voorkeuren. Impliciete signalen zijn signalen van gebruikersinteractie met het systeem. Bovendien bestaat een gebruikersprofiel uit twee componenten: 1. lange-termijn voorkeuren; en 2. korte-termijn voorkeuren. Lange-termijn voorkeuren zijn voorkeuren die langdurige interesses van gebruikers vertegenwoordigen. Korte-termijn voorkeuren zijn voorkeuren die alleen aanwezig zijn in de huidige sessie.

In dit proefschrift onderzoeken we verschillende manieren om gebruikersprofielen af te leiden met behulp van verschillende soorten signalen: expliciet en impliciet. We beginnen met het analyseren van manieren waarop we de voorkeuren van een gebruiker kunnen afleiden, met behulp van impliciete signalen uit alle sessies om een niet-gepersonaliseerde lijst van documenten opnieuw te rangschikken. We stellen een model voor waarin de waarschijnlijkheid van een klik op een document afhankelijk is van drie componenten: 1. de positie van het document op de resultaatpagina; 2. de relevantie van het document voor de query; en 3. de voorkeuren van de gebruiker. We laten zien dat dit model de beste of gelijke resultaten behaalt met state-of-the-art modellen die een gepersonaliseerde herrangschikking van een niet-gepersonaliseerde lijst van documenten uitvoeren.

We zetten ons onderzoek voort door te bestuderen hoe we een informatieve vragenlijst kunnen maken om expliciete feedback te verzamelen en de voorkeuren van een gebruiker af te leiden. We analyseren met name hoe dit probleem kan worden geformuleerd en stellen een optimalisatieprobleem voor als een wiskundige formalisatie. We stellen ook een algoritme voor dat het geformuleerde optimalisatieprobleem oplost en een informatieve vragenlijst creert. We laten zien dat we met behulp van antwoorden uit de gemaakte vragenlijst een nauwkeuriger gebruikersprofiel kunnen afleiden dan met antwoorden uit eerder voorgestelde vragenlijsten, die bestaan uit vragen met relatieve voorkeuren.

Vervolgens verbeteren we een lijst met aanbevelingen in de huidige sessie door impliciete signalen uit de historische logs van gebruikers en expliciete signalen uit de huidige sessie te combineren. In het bijzonder formuleren we het aanbevelingsproces in de huidige sessie als een spel waarin een gebruiker interactie heeft met een aanbevelingsagent door te klikken op het beste document uit de weergegeven documenten; en een aanbevelingsagent probeert de tevredenheid van de gebruiker met de sessie te maximaliseren. We laten zien hoe Reinforcement Learning-methodes voor deze taak kunnen worden toegepast en lossen het probleem op van het grote aantal acties dat door een aan-

bevelingsagent kunnen worden uitgevoerd. In het bijzonder gebruiken we een modelvrij Deep Reinforcement Learning-algoritme dat is gebaseerd op ActorCritic-framework.

Ten slotte onderzoeken we hoe interpreteerbare vragenlijsten kunnen worden gemaakt. Wij denken dat *disentangled* representaties nuttig kunnen zijn voor het maken van interpreteerbare vragenlijsten. Maar om de beste *disentangled* weergave te kiezen, moet er een metriek van *disentanglement* zijn. We bestuderen de beschikbare metrieken van *disentangled* representaties en concluderen dat de meeste van de huidige metrieken ofwel een lage score kunnen geven aan *disentangled* representaties of een hoge score kunnen geven aan *entangled* representaties. We stellen een nieuwe metriek van *disentanglement* voor, die het gewenste gedrag heeft: het geeft hoge scores aan alle *disentangled* representaties en lage scores aan *entangled* representaties.