CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

| | |
|---|---|
| Title | Subprofile aware diversification of recommendations |
| Authors | Kaya, Mesut |
| Publication date | 2019 |
| Original Citation | Kaya, M. 2019. Subprofile aware diversification of recommendations. PhD Thesis, University College Cork. |
| Type of publication | Doctoral thesis |
| Rights | © 2019, Mesut Kaya. - http://creativecommons.org/licenses/by-nc-nd/3.0/ |
| Download date | 2026-11-04 11:00:16 |
| Item downloaded from | https://hdl.handle.net/10468/8374 |



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

# Subprofile Aware Diversification of Recommendations

Mesut Kaya
MSC
115220587

**Thesis submitted for the degree of Doctor of Philosophy**



NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
INSIGHT CENTRE FOR DATA ANALYTICS

July 2019

Head of Department: Prof Cormac Sreenan

Supervisors: Dr Derek Bridge
Professor Ken Brown

# Contents

List of Figures . . . . . v
List of Tables . . . . . vii
Abstract . . . . . ix
Acknowledgements . . . . . xii

**1 Introduction** **1**
1.1 Motivation . . . . . 1
1.2 Contributions . . . . . 4
1.2.1 SubProfile detection methods . . . . . 4
1.2.2 SPAD & RSPAD . . . . . 5
1.2.3 Adaptation of diversity metrics to subprofiles . . . . . 5
1.2.4 Community-Aware Diversification . . . . . 5
1.2.5 Intent-Aware Recommendation vs. Calibrated Recommendation . . . . . 6
1.2.6 Submodularity of intent-aware approaches . . . . . 6
1.2.7 Applying SPAD to Automatic Playlist Continuation . . . 6
1.3 Publications . . . . . 6
1.4 Definitions and Notation . . . . . 7
1.5 Thesis Structure . . . . . 8

**2 Related Work** **10**
2.1 Overview of Recommender Systems . . . . . 10
2.2 Beyond Accuracy in Recommender Systems . . . . . 12
2.3 Diversity in Recommender Systems . . . . . 13
2.3.1 Diversity as an objective . . . . . 14
2.3.2 Greedy re-ranking . . . . . 15
2.3.3 Intent-aware diversification . . . . . 17
2.3.3.1 xQuAD . . . . . 18
2.3.3.2 RxQuAD . . . . . 19
2.3.3.3 SxQuAD and SRxQuAD . . . . . 20
2.3.3.4 c-pLSA and other intent-aware work . . . . . 21
2.3.4 Personalized diversification . . . . . 22
2.4 Calibrated Recommendations . . . . . 23
2.5 Conclusions . . . . . 25

**3 Experiment Design** **26**
3.1 Datasets . . . . . 26
3.1.1 MovieLens1M . . . . . 27
3.1.2 LastFM . . . . . 28
3.1.3 LibraryThing . . . . . 29
3.1.4 Facebook . . . . . 30
3.1.5 MovieLens 20 Million . . . . . 30
3.1.6 Taste Profile Subset . . . . . 30
3.2 Recommendation Algorithms . . . . . 31
3.3 Evaluation Metrics . . . . . 33

3.4 Evaluation Methodology . . . . . 36
3.4.1 Data split . . . . . 36
3.4.2 Hyperparameter values for baseline recommenders . . . 37
3.4.3 Hyperparameter values for re-ranking algorithms . . . . 37
3.4.4 Evaluation on test sets . . . . . 38
3.5 Conclusions . . . . . 38

**4 Subprofile-Aware Diversification** **39**
4.1 Why Subprofiles as Aspects? . . . . . 40
4.2 SubProfile Aware Diversity . . . . . 41
4.2.1 SPAD . . . . . 41
4.2.2 RSPAD, SSPAD and SRSPAD . . . . . 42
4.3 Subprofile Detection . . . . . 42
4.3.1 Subprofile detection from nearest-neighbours of liked items . . . . . 43
4.3.1.1 NN-1 . . . . . 43
4.3.1.2 NN-2 . . . . . 44
4.3.2 Subprofile detection from the explanations of top-$n$ recommendations . . . . . 44
4.3.2.1 IB+ . . . . . 45
4.3.2.2 DAMIB . . . . . 46
4.3.2.3 DAMIB-COVER . . . . . 47
4.3.3 Subprofile detection using profile coverage . . . . . 48
4.4 Comparison of Subprofile Detection Methods . . . . . 50
4.4.1 Qualitative comparison . . . . . 50
4.4.2 Experimental comparison . . . . . 51
4.4.2.1 Subprofile analysis . . . . . 51
4.4.2.2 Results for different subprofile detection methods 54
4.4.2.3 SPAD vs. DAMIB-COVER . . . . . 57
4.5 Conclusions . . . . . 59

**5 An Evaluation of Subprofile-Aware Diversification** **60**
5.1 Comparison of SPAD and Existing Intent-Aware Diversification Methods . . . . . 60
5.1.1 Results for different algorithms . . . . . 61
5.1.2 Results for different values of $\lambda$ . . . . . 66
5.2 Diversity Measured by Subprofiles . . . . . 70
5.2.1 Modified diversity metrics . . . . . 71
5.2.2 Results for different values of $\lambda$ . . . . . 72
5.3 A Visualization of Trade-offs . . . . . 77
5.4 Conclusions . . . . . 78

**6 Community-Aware Diversification** **79**
6.1 Community-Aware Diversity . . . . . 80
6.2 Experimental Results . . . . . 82
6.2.1 Analysis of the subprofiles . . . . . 83
6.2.2 Results for different algorithms . . . . . 84

6.2.3 Results for different values of $\lambda$ . . . 89
6.2.4 Diversity measured by subprofiles . . . 90
6.2.5 A visualization of trade-offs . . . 96
6.3 Conclusions . . . 97

**7 A Comparison of Calibrated and Intent-Aware Recommendations 98**
7.1 Intent-Aware vs. Calibrated . . . 99
7.2 The Optimality of Intent-Aware Diversification . . . 100
7.3 Calibrated Recommendation using Subprofiles . . . 104
7.4 Experimental Results . . . 104
7.4.1 Calibration results . . . 105
7.4.2 Precision . . . 107
7.4.3 Diversity results . . . 109
7.4.4 A visualization of trade-offs . . . 111
7.4.5 Other results . . . 112
7.5 Conclusions . . . 113

**8 Automatic Playlist Continuation using SPAD 114**
8.1 Challenge Overview . . . 115
8.2 Other Approaches to the Challenge . . . 117
8.3 Our Approach . . . 118
8.3.1 Cold-Start-APC . . . 119
8.3.2 SPAD-APC . . . 120
8.4 Resources . . . 121
8.5 Methodology . . . 122
8.6 Experimental Results . . . 123
8.6.1 Subprofile analysis . . . 123
8.6.2 Validation set results . . . 125
8.6.3 Leaderboard results . . . 127
8.7 Conclusions . . . 129

**9 Conclusions & Future Work 131**
9.1 Conclusions . . . 131
9.1.1 Subprofile detection methods . . . 132
9.1.2 SPAD & RSPAD . . . 132
9.1.3 Adaptation of diversity metrics to subprofiles . . . 132
9.1.4 Community-Aware Diversification . . . 133
9.1.5 Intent-aware vs. calibrated recommendation . . . 133
9.1.6 Submodularity of intent-aware approaches . . . 134
9.1.7 Applying SPAD to Automatic Playlist Continuation . . . 134
9.2 Future Work . . . 134
9.2.1 Interpretability of SPAD . . . 135
9.2.2 Perceived diversity . . . 135
9.2.3 Using temporal aspects in SPAD . . . 135
9.2.4 Using subprofiles in the cost functions . . . 135
9.2.5 SPAD for task of recommendation to shared accounts . . . 136
9.2.6 Further subprofile detection methods . . . 136

9.2.7 SPAD for cold-start users . . . . . 136

**A Hyperparameter Values** **A1**
A.1 Hyperparameter values for baseline recommenders . . . . . A1
A.2 Hyperparameter values for different subprofile detection methods A1
A.3 Hyperparameter values for different diversification algorithms . A4

# List of Figures

3.1 The popularity distribution of datasets used in the experiments. 27
3.2 For ML1M, LFM, LT and ML20M datasets, distribution of the ratings. . . . . 29

4.1 Number of subprofiles plots. . . . . 52
4.2 Length of subprofiles plots. . . . . 52

5.1 ML1M dataset. $\mathrm{Precision}\,@10$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line. . . . . 67
5.2 ML1M dataset. $\alpha$-$\mathrm{nDCG}$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line. . . . . 68
5.3 LFM dataset. $\mathrm{Precision}\,@10$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line. . . . . 68
5.4 LFM dataset. $\alpha$-$\mathrm{nDCG}$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-$\mathrm{nDCG}$ for each baseline is shown by the dotted line. 69
5.5 LT dataset. $\mathrm{Precision}\,@10$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line. . . . . 69
5.6 LT dataset. $\alpha$-$\mathrm{nDCG}$ for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-$\mathrm{nDCG}$ for each baseline is shown by the dotted line. 70
5.7 ML1M dataset. $\alpha$-$\mathrm{nDCG}$ measured using features and subprofiles for varying $\lambda$. . . . . 73
5.8 ML1M dataset. $\mathrm{ERR\text{-}IA}$ measured using features and subprofiles for varying $\lambda$. . . . . 74
5.9 ML1M dataset. $\mathrm{S\text{-}recall}$ measured using features and subprofiles for varying $\lambda$. . . . . 74
5.10 ML1M dataset. $\mathrm{EILD}$ measured using features and subprofiles for varying $\lambda$. . . . . 75
5.11 ML1M dataset. $\mathrm{ILD}$ measured using features and subprofiles for varying $\lambda$. . . . . 76
5.12 ML1M dataset. Precision vs. $\alpha$-$\mathrm{nDCG}$ trade-off plots. . . . . 77
5.13 ML1M dataset. Precision vs. $\mathrm{ILD}$ trade-off plots. . . . . 77

6.1 ML1M, LFM, LT and FB datasets. Precision values for varying $\lambda$ using MF as baseline. Values for MF are shown by dotted lines. 91
6.2 ML1M, LFM, LT and FB datasets. $\alpha$-$\mathrm{nDCG}$ values for varying $\lambda$ using MF as baseline. Values for MF are shown by dotted lines. 92
6.3 ML1M dataset. $\alpha$-$\mathrm{nDCG}$ measured using features and subprofiles for varying $\lambda$. . . . . 93
6.4 ML1M dataset. $\mathrm{ERR\text{-}IA}$ measured using features and subprofiles for varying $\lambda$. . . . . 93

6.5 ML1M dataset. $\mathrm{S\text{-}recall}$ measured using features and subprofiles for varying $\lambda$. . . . . 94
6.6 ML1M dataset. $\mathrm{EILD}$ measured using features and subprofiles for varying $\lambda$. . . . . 95
6.7 ML1M dataset. $\mathrm{ILD}$ measured using features and subprofiles for varying $\lambda$. . . . . 95
6.8 ML1M dataset. Precision vs. $\alpha$-$\mathrm{nDCG}$ trade-off plots. . . . . 96
6.9 ML1M dataset. Precision vs. $\mathrm{ILD}$ trade-off plots. . . . . 96

7.1 ML20M and TasteProfile datasets. $\mathrm{C_{KL}}$ measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines. . . . . 106
7.2 ML20M and TasteProfile datasets. Precision values for different values of $\lambda$. Values for MF are shown by dotted lines. . . . . 107
7.3 ML20M and TasteProfile datasets. $\mathrm{ILD}$ measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines. . . . . 108
7.4 ML20M and TasteProfile datasets. $\alpha$-$\mathrm{nDCG}$ measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines. . . . . 109
7.5 ML20M dataset without binarization. $\mathrm{ILD}$ values for varying $\lambda$. Values for MF are shown by dotted lines. . . . . 110
7.6 ML20M and TasteProfile datasets. Precision vs. $\mathrm{ILD}$ trade-off plots measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines. . . . . 112

8.1 Number of subprofiles. . . . . 124
8.2 Subprofile length. . . . . 125
8.3 Validation set results against the number of recommendations, $N$. 126
8.4 Profile length plots. . . . . 128
8.5 Number of subprofiles plots. . . . . 129

# List of Tables

3.1 Datasets . . . . . 28

4.1 Subprofile detection methods . . . . . 43
4.2 Subprofile detection using profile coverage . . . . . 49
4.3 Subprofile statistics . . . . . 51
4.4 Example subprofiles. These are the final subprofiles of user 5870 in the ML1M dataset. . . . . 54
4.5 Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using MF as the baseline. 56
4.6 Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using pLSA as the baseline. 57
4.7 Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using FMBPR as the baseline. 58
4.8 DAMIB-COVER & SPAD results. . . . . 59

5.1 Precision and $\alpha$-nDCG results for ML1M dataset. . . . . 62
5.2 Diversity metrics except $\alpha$-nDCG for ML1M dataset. . . . . 63
5.3 Precision and $\alpha$-nDCG results for LFM dataset. . . . . 64
5.4 Diversity metrics except $\alpha$-nDCG for LFM dataset. . . . . 65
5.5 Precision and $\alpha$-nDCG results LT dataset. . . . . 66
5.6 Diversity metrics except $\alpha$-nDCG for LT dataset. . . . . 67

6.1 Subprofile statistics . . . . . 83
6.2 Results using MF as the baseline. . . . . 85
6.3 Diversity metrics except $\alpha$-nDCG using MF baseline. . . . . 86
6.4 Results using pLSA as the baseline. . . . . 87
6.5 Diversity metrics except $\alpha$-nDCG using pLSA baseline. . . . . 88
6.6 Results using FMBPR as the baseline. . . . . 89
6.7 Diversity metrics except $\alpha$-nDCG using FMBPR baseline. . . . . 90

8.1 MPD statistics . . . . . 117
8.2 Challenge dataset statistics . . . . . 117
8.3 Creative Challenge leaderboard . . . . . 130

A.1 Hyperparameter values for each baseline recommender. . . . . A2
A.2 hyperparameters values for different subprofile detection approaches for all three datasets and all three baselines. These are optimized using validation sets. . . . . A3

I, Mesut Kaya, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

*Mesut Kaya*

# Abstract

A user of a recommender system is more likely to be satisfied by one or more of the recommendations if each individual recommendation is *relevant* to her but additionally if the set of recommendations is *diverse*. The most common approach to recommendation diversification uses re-ranking: the recommender system scores a set of candidate items for relevance to the user; it then re-ranks the candidates so that the subset that it will recommend achieves a balance between relevance and diversity. Ordinarily, we expect a trade-off between relevance and diversity: the diversity of the set of recommendations increases by including items that have lower relevance scores but which are different from the items already in the set.

In early work, the diversity of a set of recommendations was given by an aggregate of their distances from one another, according to some semantic distance metric defined on item features such as movie genres. More recent *intent-aware* diversification methods formulate diversity in terms of coverage and relevance of *aspects*. The aspects are most commonly defined in terms of item features. By trying to ensure that the aspects of a set of recommended items cover the aspects of the items in the user's profile, the level of diversity is more personalized. In offline experiments on pre-collected datasets, intent-aware diversification using item features as aspects sometimes defies the relevance/diversity trade-off: there are configurations in which the recommendations exhibits increases in both relevance and diversity.

In this thesis, we present a new form of intent-aware diversification, which we call SPAD (Subprofile-Aware Diversification). In SPAD and its variants, the aspects are not item features; they are *subprofiles* of the user's profile. We present a number of different ways to extract subprofiles from a user's profile. None of them is defined in terms of item features. Therefore, SPAD and its variants are useful even in domains where item features are not available or are of low quality.

On several pre-collected datasets from different domains (movies, music, books, social network), we compare SPAD and its variants to intent-aware methods in which aspects are item features. We also compare them to calibrated recommendations, which are related to intent-aware recommendations. We find on these datasets that SPAD and its variants suffer even less from the relevance/diversity trade-off: across all datasets, they increase both relevance and

diversity for even more configurations than other approaches. Moreover, we apply SPAD to the task of automatic playlist continuation (APC), in which relevance is the main goal, not diversity. We find that, even when applied to the task of APC, SPAD increases both relevance and diversity.

To beloved ones...

# Acknowledgements

From October 2015 to today, throughout my study here at Insight Centre for Data Analytics, University College Cork, there have been so many people that have helped me one way or another. I cannot thank to all of them one by one, but I will try to express my gratitude to them as much as I can.

First of all, I would like to express my sincere appreciations to my supervisor, Dr. Derek Bridge, for his guidance, support, encouragement and positive attitude throughout my study. I am gladful for having chance of working with him. Today, when I look back to the day I have started my study under supervision of Derek and today, I can proudly say that I have learned so much. Derek's guidance with proper feedback, always led me to a better direction throughout my study.

Members of Recommender Systems Group in Insight Centre for Data Analytics deserve a special appreciation. I would like to thank to Francisco Peña, Diego Carraro, Rohit Dhamane and Arpit Rana for their support and for the discussions and chats that helped me a lot throughout my study. Francisco and Diego deserve a special mention, especially the time we have spent together outside of the work was also a motivating factor for me.

I would also like to thank to other members of the Insight Centre for Data Analytics. Especially, Milan De Cauwer, who is the reason why I am so obsessed with climbing today. Also, my climbing and camping partners, Yves Sohege, Mohamed Siala and Anne-Marie George deserves a special mention. My gratitude also goes to Begum Genc, Federico Toffano, Andrea Visentin, Tadhg Fitzgerald and Cathal Hoare.

Devrim Gunyel, who encouraged me to apply to study here deserves maybe the most special mention. In 2015, when I was working as a Research Engineer in Turkey, I had no intention to return to academia at all. I even was considering to quit working as a Computer Scientist. She was the one that changed my mind. Devrim has helped me a lot throughout this study. Especially, during the hard times, she was always there for me to motivate me and help me.

Another special mention must go to my friends in Batman, Turkey. Every single time that I have visited Batman, the first question was: "Did you finish writing your thesis?". Good news guys, these are the last sentences of the thesis. Thanks for your support.

I am very grateful that my family encouraged me throughout study. My mother and father, my sisters and my brother, they all deserve special gratefulness.

# Chapter 1

# Introduction

## 1.1 Motivation

Recommender systems have become an essential part of social networks (such as Facebook and Twitter), e-commerce sites (such as Amazon), music and video streaming platforms (such as Spotify and Netflix) and many other services on the web. Recommender systems aim to satisfy the users of these platforms by helping them to discover content that matches their interests.

Besides being an essential part of many platforms, recommender systems have become an active research field as well. For instance, the *ACM Conference on Recommender Systems* is one of the most relevant and among the most prestigious conferences in the field. Recommender systems have also become an important topic in other top conferences such as the *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, *ACM's Special Interest Group on Information Retrieval* and the *Web Conference (WWW)*. Recommender systems research is also published in journals, for instance the *ACM Transactions on Interactive Intelligent Systems* and *User Modeling and User-Adapted Interactions*.

An assumption of very early work on recommender systems was that the goal was to accurately predict the users' opinions of candidate items, and to use these predictions to select items to recommend. It was soon recognized that it is not enough for predictions to be accurate or recommendations to be merely relevant. A focus on accuracy or relevance may result in recommendations that are too obvious (e.g. sequels in a movie recommender), too popular (e.g. blockbuster movies), too similar to each other, or too similar to the user's profile. It

can lead to monotony in a user's interactions with the system [EMB17], and it may narrow, rather than broaden, a user's horizons [CWM+17]. In many domains, recommendations must be novel to the user or serendipitous, and a set of recommendations must be diverse [MRK06]. It is diversity that is the focus of this thesis.

Diversity is one response to uncertainty. A recommender cannot be certain of a user's short-term or longer-term interests, both because some user profiles are small and others, while they may not be so small, will contain preferences over different kinds of items. In the face of uncertainty, a diverse set of recommendations is more likely to contain one or more items that will satisfy the user. A number of user studies shows that a diverse set of recommendations can be more attractive to users, reduce the difficulty of selecting an item to consume from among the recommendations [NCGV18], and even increase satisfaction with the chosen item, e.g. [WGK16], especially when visual interfaces are designed to highlight the diversity of the recommended items, e.g. [TB17, TB18].

There is often thought to be a trade-off between accuracy and diversity. A set of *randomly-chosen* items, for example, is likely to be diverse, but the individual recommendations are less likely to be relevant to the user. Or, to give another example, recommending a set of popular items will, in many cases, result in high accuracy but may lead to lower diversity [AK08]. Past research has considered how to increase the level of diversity at the expense of negligible accuracy loss [AK09]. However, as we will show in the following chapters of this thesis, newer diversification methods may not be so susceptible to this trade-off and may even increase both the accuracy and the diversity of the recommendations.

A diverse set contains items that are different from one another. Early work measures the diversity of a set of items as an aggregate of the all-pairs dissimilarity of the items. Dissimilarity is computed by a distance metric (or the complement of a similarity metric) defined on item features (e.g. movie genres), item ratings or latent factors. Typically, a recommender system finds a set of recommendations incrementally, by considering the marginal contribution that would be made by adding a candidate item to the result set [CG98]. The marginal contribution is usually a linear combination of how relevant an item is to the user and how dissimilar it is from the set of items that have been included in the result set so far. Within the linear combination, a parameter determines the trade-off between accuracy and diversity. In principle, this parameter can have different values for different users (although this is rare in practice). But

the assumption in this early work is that diversity should be measured in the same way across all users (by item distances); it is not personalized. This assumption may be wrong. Consider a music recommender, for example. One user may like only pop and rock music; for her, a diverse set is one that covers both genres. Another user may have much more catholic tastes and, for him, a diverse set must cover a much wider range of genres.

More recently within Information Retrieval, a new approach to result set diversification has emerged, known as *intent-aware diversification* [SMO10, VCV12]. The idea is that, to satisfy a user, a result set must cover her intention to a certain extent. In the case of an ambiguous query, there is uncertainty about her intention. The query term "apple" could refer to the fruit or the corporation, for example. In this case, there should be items in the result set corresponding to each interpretation, thus ensuring that the user's intent is covered. The more ambiguous a query term is then, other things being equal, the more diverse the result set needs to be if it is to cover all the possible interpretations.

Vargas et al. [VCV11] have adapted intent-aware diversification from Information Retrieval to recommender systems. In this case, there is usually no query. Instead of covering different interpretations of an ambiguous query, the idea analogously is to cover the different tastes or interests of the user, as revealed by her profile. A user's tastes or interests are commonly modelled as a probability distribution over so-called *aspects* of the items, which can be explicit item features (such as news item categories or music genres) or implicit item features such as the latent factors computed, e.g., by a matrix factorization recommender system. Although the same aspects (e.g. music genres) are used for all users, the aspect probabilities, computed from the user profiles, may differ from user to user, making intent-aware diversification a more personalized approach. The recommendations to the user who likes only pop and rock music, for example, are diversified to cover both pop and rock music and, to a certain extent, to the degree that these are reflected in her profile.

Using item features for aspects brings several problems. In some domains, item features may not be readily available. Even when available, they may fail to capture the subtleties of a person's preferences; they may be too coarse; they may be inconsistently applied to the items; and they may be noisy. Instead of defining tastes and interests in terms of item features, alternatively patterns in user interactions with items can be used to define the distinct tastes and interests of the users [Kul18]. Patterns of interaction with items, *subprofiles* in

our case, rather than item features, form the basis of the aspects of an intent-aware approach to diversification throughout this thesis.

In this thesis, we are proposing a new intent-aware diversification framework, called Subprofile-Aware Diversification (SPAD). In SPAD, aspects are user subprofiles, rather than item features. A subprofile is simply a subset of the items that the user likes, representing one of the user's distinct tastes or interests. Immediately, we see that subprofiles tend to give us more fine-grained aspects than item features: if the user likes $m$ items, then there are $2^m - 1$ possible subprofiles, i.e. all subsets except the empty set. But also it is a user's interactions with items that define which of these possible subprofiles are aspects for that user.

## 1.2 Contributions

The main goal of this thesis is to address the question of *how best to produce a relevant but diverse set of recommendations*. To do this, first, we propose a new approach: a form of intent-aware diversification but one which diversifies with respect to subprofiles that we mine from the user's profile, rather than with respect to genres or other item features. The following sections describe the contributions that we make in addressing the main goal of the thesis.

### 1.2.1 SubProfile detection methods

We define eight methods for detecting subprofiles. We group them into three: there are two methods ($\mathrm{NN\text{-}1}$ and $\mathrm{NN\text{-}2}$) that use the nearest-neighbours of liked items; there are three methods ($\mathrm{IB+}$, $\mathrm{DAMIB}$ and $\mathrm{DAMIB\text{-}COVER}$) that use the explanations of top-$n$ recommendations, two of which are from [VG15]; and there are three methods ($\mathrm{IB+_{cp}}$, $\mathrm{DAMIB_{cp}}$ and $\mathrm{DAMIB\text{-}COVER_{cp}}$) that consider profile coverage.

In Chapter 4, we give a comprehensive empirical comparison of all eight methods for detecting subprofiles on three different datasets. The methods that use the nearest-neighbours of liked items have several advantages over the others, and the empirical comparison shows that one of these methods (designated $\mathrm{NN\text{-}1}$) is also the best in terms of recommendation accuracy and diversity.

We analyze the subprofiles that NN-1 finds. We give descriptive statistics and plot distribution graphs to better understand how subprofiles differ from dataset to dataset. We also give an explicit example of how subprofile detection works on one of the datasets we use.

### 1.2.2 SPAD & RSPAD

In Chapter 4, we define the Subprofile-Aware Diversificiation (SPAD) and Relevance based SPAD (RSPAD) intent-aware diversification methods.

In Chapter 5, using NN-1 as the subprofile detection method, we compare SPAD and RSPAD against several existing intent-aware-diversification frameworks where aspects are item features.

### 1.2.3 Adaptation of diversity metrics to subprofiles

The algorithms that we propose in this thesis, including SPAD and its variants, make no use of item features at all. Since the diversity metrics that we take from the literature and use in the experiments throughout this thesis are computed with respect to the item features, SPAD and its variants are at a disadvantage. In Chapter 5, we have adapted several diversity metrics so that they use subprofiles instead of item features. Using these new metrics alongside the existing ones gives a more balanced view of the performance of the recommender algorithms.

### 1.2.4 Community-Aware Diversification

In Chapter 6, we propose Community-Aware Diversification (CAD), in which aspects are again subprofiles but are detected indirectly through users who are similar to the active user. Thus, it explores the idea that a user's community (similar users) correlates with her tastes or interests (subprofiles). We show that in certan domains this can give better results.

### 1.2.5 Intent-Aware Recommendation vs. Calibrated Recommendation

In Chapter 7, we give a comparison between intent-aware approaches to recommendation diversification and recently-proposed calibrated recommendation, which has apparent similarities to intent-aware diversification. We define a new variant of calibrated recommender system, one which calibrates with respect to subprofiles, rather than item features. Finally, much as we adapted diversity metrics that used item features to ones that use subprofiles, we also adapt the calibration metric to give one that uses subprofiles instead of item features.

### 1.2.6 Submodularity of intent-aware approaches

As we will explain later, diversification of recommendations is most often achieved by a greedy algorithm. The algorithm has a $(1 - \frac{1}{e})$ optimality guarantee if its objective function is submodular and monotone. In Chapter 7, we adapt a proof from the literature to show that the objective function used in intent-aware approaches, such as xQuAD and SPAD, is both submodular and monotone.

### 1.2.7 Applying SPAD to Automatic Playlist Continuation

SPAD's main goal is diversification. But, our experimental results show, that it also always increases recommendation precision too. In Chapter 8, we apply SPAD to a task that is different from the others in this thesis — a task where precision, rather than diversity, is the main goal. The task comes from the *ACM RecSys Challenge 2018*, which is about Automatic Playlist Continuation (APC). We show that, playlists contain subprofiles and we show the advantages of applying SPAD to APC.

## 1.3 Publications

We have published the work described in this thesis in international journals, conferences, their workshops and poster sessions. These publications are listed

as follows:

- Mesut Kaya and Derek Bridge: *A Comparison of Calibrated and Intent-Aware Recommendations*, in Proceedings of the 13th ACM Conference on Recommender Systems, 2019.
- Mesut Kaya and Derek Bridge: *Subprofile-aware diversification of recommendations*. User Modeling and User-Adapted Interaction, Apr 2019.
- Mesut Kaya and Derek Bridge: *Community-Aware Diversification of Recommendations*, in Proceedings of the 34th Annual ACM Symposium on Applied Computing, pp. 1639-1646, 2019.
- Mesut Kaya and Derek Bridge: *Accurate and Diverse Recommendations Using Item-Based SubProfiles*, in Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, AAAI, pp.462-467, 2018.
- Mesut Kaya and Derek Bridge: *Automatic Playlist Continuation using Subprofile-Aware Diversification*, in Proceedings of the Workshop on the ACM Recommender Systems Challenge (Workshop Programme of the Twelfth ACM Conference on Recommender Systems), pp.1-6, 2018.
- Mesut Kaya and Derek Bridge: *Intent-Aware Diversification using Item-Based SubProfiles*, in Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems, CEUR Workshop Proceedings, vol-1905, 2017.

Unrelated to our work on diversification, we also published the following:

- Mesut Kaya and Derek Bridge: *Improved Recommendation of Photo-Taking Locations using Virtual Ratings*, in Proceedings of the Workshop on Recommenders in Tourism (Workshop Programme of the Tenth ACM Conference on Recommender Systems), CEUR Workshop Proceedings, vol-1685, pp.1-7, 2016.

## 1.4 Definitions and Notation

We give a brief summary of the main definitions and notation that we use for the rest of the thesis to make it easier to follow. We describe additional definitions and notation when necessary.

Let $I$ be the set of all items, $U$ be the set of all users, and $R$ be a $|U| \times |I|$ matrix, where $r_{u,i} \in R$ is $u$'s rating of $i$ or $r_{u,i} = \perp$ if $u$ has not rated $i$. We denote the user profile of user $u$ by $I_u = \{i \in I : r_{u,i} \neq \perp\}$.

Subprofiles of a user are defined in terms of items that the user likes, i.e. ones to which she has given a positive rating. Henceforth, we will use the phrase *liked-item-set* to refer to the set of items that the user likes, and will designate this set by $I_u^+$ where $I_u^+ \subseteq I_u$. In the case of a recommender system that uses positive-only feedback, user $u$'s liked-item-set is the set of items she has interacted with (liked, clicked on, purchased, etc.), and these are simply the ones in the user's profile i.e. for implicit ratings, $I_u^+ = I_u$. In the case of a recommender system that uses numeric explicit ratings $r_{u,i}$ (e.g. 1–5 stars), then $I_u^+$ must be defined in terms of items the user liked, which will usually involve thresholding the ratings, e.g. in our experiments in the following chapters, we use $I_u^+ = \{i \in I : r_{u,i} \geq 4\}$ for 1–5 stars ratings.

We denote the set of all item features of $I$ as $\mathcal{F}$, and the set of features of a specific item $i \in I$ as $\mathcal{F}_i$. Depending on the domain, item features may vary. Consider, for instance, the datasets we use in the following chapters: item features include movie and music genres, user-assigned tags and keywords.

## 1.5 Thesis Structure

The structure of the thesis is:

- In Chapter 1, we have presented the motivation behind our work and we have summarized the contributions that we have made.
- In Chapter 2, we review the literature on the main topics of interest for this thesis.
- In Chapter 3, we present our experiment design, which is used in the subsequent chapters. First, we give the details of the datasets. Second, we provide a detailed description of the recommendation algorithms that we use as baselines. Third, we present the definitions of the metrics we use to evaluate the performance of our algorithms. Fourth, we describe our evaluation methodology.
- In Chapter 4, we propose a novel intent-aware diversification framework, SPAD, which is the core contribution of the thesis, and a variant, Rele-

vance based SPAD (RSPAD). SPAD and RSPAD represent user tastes and interests by subprofiles, rather than item features. We define eight different methods for detecting subprofiles. We give a comprehensive empirical comparison of all eight methods for detecting subprofiles on different datasets and select one to use within SPAD and RSPAD for the rest of the thesis.

- In Chapter 5, we give a comprehensive empirical comparison of SPAD and RSPAD against a number of other diversification methods. We also adapt several diversity metrics that are measured by item features to give versions that use subprofiles instead of item features.
- In Chapter 6, we propose and evaluate an alternative to SPAD, Community-Aware Diversification, which we refer to as CAD. CAD detects subprofiles indirectly through user-user similarities. CAD allows us to explore whether a user's community (similar users) has correlation with her interests (subprofiles).
- In Chapter 7, we compare intent-aware approaches to recommendation diversification to a related concept, *calibrated recommendation*. We give the proof of the optimality of intent-aware approaches including SPAD and RSPAD. We adapt the calibration metric that uses item features to give a variant that uses subprofiles instead. We also define a new calibration recommender algorithm that uses subprofiles as user interests, instead of item features.
- In Chapter 8, we apply SPAD to the task of Automatic Playlist Continuation. By applying the idea of SPAD to a different task on a larger dataset, we show the generality of SPAD.
- In Chapter 9, we conclude the thesis and offer ideas for future work.

# Chapter 2

# Related Work

In this chapter, we study the state-of-the-art in the topics of interest to our research, especially in the area of diversification of recommender systems.

## 2.1 Overview of Recommender Systems

A recommender system aims to help its users to discover resources that align with their tastes and interests [RRS15]. They have become an essential part of a great variety of platforms. For instance, in social networks they recommend people to follow (Twitter), connect (Linkedin) or become friends with (Facebook); in e-commerce sites they recommend products to buy (Amazon); in music streaming platforms they recommend songs to listen to (Spotify, Deezer, Pandora); in video streaming platforms they recommend videos to watch (YouTube, Netflix); and they are part of many other services on the web.

There are many different recommendation algorithms. The choice of best algorithm depends on the domain they are applied to (e.g. music or news); the content delivery mechanism (e.g. streaming or download), the type of problem they solve (e.g. recommending complementary items or alternative items), the type of user feedback they use (e.g. explicit user ratings for products or user listening history for music), and many other factors. Although there exists a variety of different recommendation algorithms, and different classifications of them, we will distinguish between six classes of algorithms, as is done in [RRS15]:

- Content-based: recommendations are generated by using item features as

content.

- Collaborative Filtering: recommendations are generated by using patterns of user-item preferences.
- Demographic: recommendations are generated by using demographic information about the users.
- Knowledge-based: recommendations are generated by using specific domain knowledge about the degree of usefulness of items to the users.
- Community-based: recommendations are generated by using preferences of the users' friends.
- Hybrid: recommendations are generated by using combinations of the above mentioned algorithms.

Here we will not go into the details of each of the different classes; for detailed definitions, see [RRS15]. The recommendation algorithms that we use in the following chapters of the thesis are collaborative filtering algorithms. These have been widely used and proven to be effective. We give their details in Chapter 3.

As we mentioned before, selecting the type of recommendation algorithm depends on different criteria, one of which is the type of user feedback on user-item interactions. In general, user feedback can be categorized into two: *implicit* and *explicit*. In the case of implicit feedback, the user does not directly provide her preferences over items. The preferences are estimated from her behaviour with the items in the system (e.g. clicking to view product details, listening to music, watching videos, skipping music in a playlist, and so on). In the case of explicit feedback, however, the user signals her interest in items directly (e.g. by giving a star rating on a 1–5 scale to a movie, by writing reviews about restaurants or liking/disliking products). The majority of datasets that we use to test the performance of our approaches record explicit feedback. But, to show that our approaches work for both explicit and implicit feedback, we also use some implicit feedback datasets.

In order to measure and compare the performance of different recommendation algorithms, different evaluation methodologies are used. It is common to classify evaluation methodologies into three [SG11]:

- *Offline experiments*: These are the most common and easiest way to estimate recommender performance. Users do not actively participate in the

evaluation. Instead, performance is estimated automatically using pre-collected datasets.

- *User studies*: A set of users actively participates in the evaluation process. They are asked to perform some actions and perhaps answer related questions regarding their experience in a task or tasks designed to explore the recommender performance.
- *Online experiments*: Large-scale experiments are run on a deployed system. A typical example is A/B testing.

In this thesis, we use offline experiments to compare the performance of our proposed approaches with existing approaches. We use offline experiments because they are simple and cost-effective compared to user studies or online experiments. It would be too costly to run the volume of experiments that we conduct using user trials or online experiments.

## 2.2 Beyond Accuracy in Recommender Systems

In early work on recommender systems, the main goal, either explicitly or implicitly, was to accurately predict the users' opinions of candidate items or generate, from the candidate items, a top-$n$ list of relevant recommendations. It was soon recognized that prediction accuracy or relevancy of top-$n$ recommendations should not be the only measure of recommender quality [HKTR04, MRK06]. For example, recommendations must be novel or serendipitous to the user and a set of recommendations must be diverse [MRK06, KB16]. The novelty of a recommendation to a user depends on how familiar a user is with the item [KKT+15]. The serendipity of a recommendation to a user depends on how relevant, novel and unexpected the item is to the user [KKZV18]. The diversity of a set of recommendations depends on the variety of the items, or the differences between the items, in the set [Var15].

Consider, for instance, a video streaming service's recommender system. If it recommends us a movie that we have never heard of before, we would say that the recommended movie is novel to us to a certain extent. If, say, the movie is also a movie that we would not have found by our own and we enjoy watching it, we would say the recommendation was serendipitous as well. If the system recommends a set of movies to watch, and if the set includes movies of different types, we might say that the recommended set is diverse.

Novelty, serendipity and diversity are among the so-called beyond-accuracy factors in the evaluation of recommender systems. Note that there are different definitions of the beyond-accuracy factors, and no consensus on them. We will not give all the different definitions here. For a survey, see [KB16]. In this thesis, *diversity* is the beyond-accuracy objective that we focus on. In the next section, we review the state-of-the-art in diversity in recommender systems.

## 2.3 Diversity in Recommender Systems

Diversity, as a general concept, can be seen as the variety or the differences within parts of an experience [Var15]. For example, a movie recommender is producing a diverse set of recommendations if it includes movies of different types (e.g. genres) rather than different movies of similar types.

Note that, in this thesis, we use the term ”diversity” exclusively to refer to a property of a set of recommendations, i.e. the diversity of a set of recommendations generated for a user. Elsewhere, in the recommender systems literature, there are different definitions of ”diversity” — ones that do not refer to a property of a recommendation set. For example, the word “diversity” can also refer to a property of a recommender system as a whole, referring to the extent to which the system’s recommendations cover the item catalog. For instance, “sales diversity” and “aggregate diversity” (e.g. measured by the *Gini index* [SG11]) are system-centric definitions of diversity [AK12]. There is also a notion of “temporal diversity”, measuring the extent to which recommendation sets generated for an individual user differ between different points in time [LHCA10]. For surveys, see [KB16, CHV15]).

Diversity in a set of query results or in a recommendation set is one response to uncertainty. In Information Retrieval (IR), there is value in ensuring that each retrieved document is relevant to the user’s query but also that the set of retrieved documents is diverse, i.e. that they are different from one another [CG98, CKC+08]. In IR, diversity is useful in improving the extent to which the user’s intent is covered by at least one retrieved document, especially in cases of uncertainty caused by query ambiguity or underspecification [AGHI09]. In recommender systems, uncertainty is caused by small user profiles (e.g. for cold-start users) and by profiles that span different tastes. Besides, a recommender cannot be certain of a user’s short-term interests. Short-term interests

are, almost by definition, ephemeral and therefore difficult to predict. While a recommender should recommend items that it predicts are relevant to the user (satisfying her long-term interests), its chances of recommending items that satisfy the user on a given occasion can be increased by recommending a set of diverse items [CHV15, KB16].

However, a recommender system must strike a balance between accuracy or relevance on the one hand and diversity on the other hand. Indeed, early work on diversification implied a trade-off between the two: increasing one typically resulted in a decrease in the other [AK09].

### 2.3.1 Diversity as an objective

There is some work in which diversity is explicitly a part of the objective function that the recommender seeks to optimize when generating recommendations, e.g. [Hur13, SYCY13, CWM+17].

Hurley includes diversity (measured by item dissimilarity) into the objective function of a pairwise learning-to-rank approach, which learns user and item factors by minimizing an objective function that is formulated as the difference between the predicted and original ranking for item pairs [Hur13]. By including the item dissimilarities into the objective function, user and item factors are learned by the combination of ranking and diversity. He shows that his proposed approach produces more diverse recommendations with a small loss in relevance of the recommendations.

Su et al. propose what they call a set-oriented personalized ranking [SYCY13]. This integrates diversity into a matrix factorization model. The objective function used in the matrix factorization incorporates a set diversity bias term that models the diversity of the recommendation set. Diversity is measured as an all-pairs aggregate of the inner-product of their latent factors.

Cheng et al. propose a method for diversified collaborative filtering that they refer to as DCF [CWM+17]. In DCF, there are two components. First, a structural support vector machine (SVM) learns to recommend sets of items to users. It creates an accurate and diverse ground-truth for each user: to ensure accuracy, it uses highly-rated items from the user's profile; for diversity, a subset of the highly-rated items that maximizes set diversity measured by item dissimilarity is selected. Second, there is a parameterized matrix factorization algorithm

that generates lower-dimensional representations of the users and items that are used in each iteration of the structural SVM.

Although, diversity can be explicitly part of an objective function as explained above, the dominant approach to diversification is greedy re-ranking.

### 2.3.2 Greedy re-ranking

Recommendation diversification aims to determine an optimal recommendation set of size $N$ items, denoted here by $RL^*$. Commonly, the objective function is a linear combination of the relevance of the items in the recommendation set and the diversity of that set, the trade-off between the two being controlled by a parameter $\lambda$ ($0 \leq \lambda \leq 1$):

$$RL^* = \underset{RL,|RL|=N}{\arg\max}\, (1-\lambda)s(RL) + \lambda\, \mathrm{div}(RL) \tag{2.1}$$

where typically $s(RL) = \sum_{i \in RL} s(u,i)$ is a modular function, that is the sum of the predicted relevance scores $s(u,i)$ of each recommended item $i$ to user $u$, and $\mathrm{div}(RL)$ is typically a submodular function that measures the diversity of the set $RL$. Finding the optimal recommendation set $RL^*$ is NP-hard. When the objective function defined in Eq. 2.1 is *monotone* and *submodular*, a $(1-\frac{1}{e})$ approximation to the optimal solution can be computed greedily [NWF78], where $e$ is Euler's number. We will give definitions of monotonicity and submodularity in Chapter 7.

The greedy re-ranking approach assumes the existence of a conventional recommender algorithm (which we will refer to as the *baseline recommender*), which, for user $u$, produces a set of recommended items, $RS$, and, for each item $i$ in $RS$, a relevance score, $s(u,i)$ — the predicted relevance of recommended item $i$ to user $u$. The greedy algorithm re-ranks $RS$ by iteratively inserting into ordered result list $RL$ the item $i$ from $RS$ that maximizes a function, $f_{obj}(i, RL)$; see Algorithm 1. Similarly to Eq. 2.1, $f_{obj}$ is usually defined as a linear combination of the item's relevance score and the contribution item $i$ makes to the diversity of $RL$, $\mathrm{div}(i, RL)$, the trade-off between the two again being controlled by a parameter $\lambda$ ($0 \leq \lambda \leq 1$):

$$f_{obj}(i, RL) = (1-\lambda)s(u,i) + \lambda\, \mathrm{div}(i, RL) \tag{2.2}$$

**Algorithm 1** Greedy re-ranking algorithm

**Input:** $RS$, set of recommendations for user $u$, each with relevance score
**Output:** $RL$, ranked list containing all items in $RS$

```
RL ← [ ]
while |RS| > 0 do
    i* ← arg max_{i∈RS} f_obj(i, RL)
    delete i* from RS
    append i* to the end of RL
return RL
```

In Maximal Marginal Relevance (MMR) [CG98], for example, $\mathrm{div}(i, RL)$ is the maximum of the distances between $i$ and the items already selected:

$$\mathrm{div}(i, RL) = \max_{j \in RL} \mathrm{dist}(i, j) \tag{2.3}$$

The distance between items $i$ and $j$, $\mathrm{dist}(i, j)$, can be calculated from meta-data such as movie genres or book categories [SM01, ZMKL05] or from item ratings data [KB06]. Alternatively, $\mathrm{div}(i, RL)$ can be computed as the average (or sum) of the all-pairs intra-list distances, $\mathrm{div}(i, RL) = \mathrm{ILD}(\{i\} \cup RL)$, where the ILD of any list of items $L$ is given by:

$$\mathrm{ILD}(L) = \frac{2}{|L|(|L|-1)} \sum_{i \in L} \sum_{j \in L, j \neq i} \mathrm{dist}(i, j) \tag{2.4}$$

The final recommendation comprises the top-$N$ members of the re-ranked list, $RL$, where $N < |RL|$. Re-ranking using Eqs. 2.3 or 2.4 can result in a top-$N$ that comprises items that are dissimilar to each other.

The assumption behind this form of diversification is that dissimilar items will address the different tastes and interests of the user, but there is nothing in the operation of the system to explicitly ensure that each of the user's tastes and interests are addressed, nor that each is addressed to an appropriate degree.

Other approaches, going under the name *intent-aware diversification*, seek to select items that explicitly address different user tastes and interests and that each is addressed to a degree that is reflected by their prevalence in the user's profile.

Before reviewing the intent-aware approaches, we mention some other recent greedy re-ranking algorithms, which seek to increase diversity in somewhat different ways.

Some recent work, for example, uses Determinantal Point Processes (DPP), which give a probabilistic model of subset selection that prefers diverse sets [WRB+18, CZZ18]. For a set of items $S$, a point process $\mathcal{P}$ is a probability distribution on the powerset of $S$. In DPPs, the probabilities are proportional to a combined measure of the relevance and diversity of the items in the subsets. Wilhelm et al. propose a set-wise optimization approach for recommendation diversification [WRB+18]. They use DPPs that combine estimates of item quality and distances between pairs of items. They use a greedy approximation to apply their method to a large-scale video recommendation system. Their empirical results show that their model improves user engagement with the system. Chen et al. propose a fast greedy algorithm that uses DPPs to improve recommendation diversity [CZZ18]. Their approach to using DPPs is similar to Wilhelm et al.'s approach, as both use DPPs that combine estimates of item quality and distances between items. What differs is the implementation of the greedy algorithm. Chen et al. show that their implementation is faster than existing greedy implementations of DPPs. They also propose a variant of their algorithm that works for the scenarios where diversity is only required within a sliding window of recent recommendations.

Ashkan et al. propose an optimal greedy solution that maximizes a modular objective function (which computes the utility of the items), using submodular constraints that measure the increase in diversity in terms of what they call topic coverage [AKBW15]. For topics, in a movie dataset they use movie genres. When the number of topics is very large, their proposed approach may not be practical. They propose to overcome this limitation by using dimensionality reduction techniques such as topic modeling [BNJ03].

### 2.3.3 Intent-aware diversification

Intent-aware diversification in IR assumes a set of query aspects (e.g. document categories or query reformulations from a search engine) and diversifies by re-ranking the query result set in a way that balances relevance with the degree to which these aspects are covered [AGHI09, SMO10, VCV11, VCV12].

Intent-aware methods for recommendation diversification take inspiration from the work done in IR [VCV11, VCV12, WH16]. These methods assume a set of aspects $\mathcal{A}$ which describe the items and for which user interests can be estimated. The aspects might be explicit: for example, categories such as politics,

sport and entertainment in a news recommender, or genres such as comedy, thriller and horror in a movie recommender. Alternatively, aspects might be implicit, e.g. corresponding to the latent factors found by a matrix factorization recommender system [KB11].

In intent-aware methods Eq. 2.1 can be rewritten as:

$$RL^* = \underset{RL,|RL|=N}{\arg\max}\,(1-\lambda)s(RL) + \lambda\,\mathrm{div}_{\mathrm{IA}}(RL) \tag{2.5}$$

$\mathrm{div}_{\mathrm{IA}}(RL)$ measures the diversity of the set $RL$. But this is not simply a measure of how different the items are from each other, as it would be in more conventional approaches to diversity [CG98, SM01, ZMKL05]. Instead, it is defined in terms of coverage of the user's tastes and interests, but with coverage modulated by recommendation relevance (below).

#### 2.3.3.1 xQuAD

In Vargas's adaptation to recommender systems [Var15] of Santos et al.'s Query Aspect Diversification framework (xQuAD) [SMO10], we can write $\mathrm{div}_{\mathrm{IA}}(RL) = \mathrm{div}_{\mathrm{xQuAD}}(RL)$, where $\mathrm{div}_{\mathrm{xQuAD}}(RL)$ is defined as follows:

$$\mathrm{div}_{\mathrm{xQuAD}}(RL) = \sum_{a\in\mathcal{A}} p(a|u)\left(1 - \prod_{i\in RL}(1-p(i|u,a))\right) \tag{2.6}$$

User $u$'s interests can be formulated as a probability distribution $p(a|u)$ for aspects $a \in \mathcal{A}$. The probability of choosing an item $i$ from the set of recommendations $RS$ given an aspect $a$ of user $u$ is denoted by $p(i|u,a)$.

In xQuAD, diversification can be achieved by re-ranking a conventional recommender's recommendation set $RS$ as per Algorithm 1 and Eq. 2.2 but with $\mathrm{div}(i, RL) = \mathrm{div}_{\mathrm{xQuAD}}(i, RL)$ defined as:

$$\mathrm{div}_{\mathrm{xQuAD}}(i, RL) = \mathrm{div}_{\mathrm{xQuAD}}(\{i\} \cup RL) - \mathrm{div}_{\mathrm{xQuAD}}(RL) \tag{2.7}$$

Using Eq. 2.6, we can obtain the following [AGHI09, SMO10, Var15]:

$$\mathrm{div}_{\mathrm{xQuAD}}(i, RL) = \sum_{a\in A}[p(a|u)p(i|u,a)\prod_{j\in RL}(1-p(j|u,a))] \tag{2.8}$$

Consider the case where the aspects are explicit features $\mathcal{F}$, i.e. $\mathcal{A} = \mathcal{F}$, hence we will write $p(f|u)$ and $p(i|u, f)$ instead of $p(a|u)$ and $p(i|u, a)$. Let $\mathcal{F}_i$ be the subset of $\mathcal{F}$ that describes item $i$ (e.g. the genres of movie $i$) and, as in Section 1.4, let $I_u$ denote the items that are in the user's profile. Then $p(f|u)$ can be estimated as:

$$p(f|u) = \frac{|\{i \in I_u : f \in \mathcal{F}_i\}|}{\sum_{f' \in \mathcal{F}} |\{i \in I_u : f' \in \mathcal{F}_i\}|} \tag{2.9}$$

$p(i|u, f)$, the probability of choosing $i$ from a set of recommendations $RS$ given explicit aspect $f$ of user $u$, can be estimated as:

$$p(i|u, f) = \frac{\mathbb{1}(i, f)s(u, i)}{\sum_{j \in RS} \mathbb{1}(j, f)s(u, j)} \tag{2.10}$$

where $\mathbb{1}(i, f) = 1$ if $f \in \mathcal{F}_i$ and $0$ otherwise.

xQuAD is a generalization of the IA-Select method [AGHI09]. The latter does not consider relevance, hence they are equivalent in the case when $\lambda = 1$ in Eq. 2.2.

#### 2.3.3.2 RxQuAD

A possible weakness of xQuAD is that its formulation implies selection of a single item from the recommended set $RS$. In RxQuAD, Vargas et al. formulate a model based on maximizing relevance, rather than the probability of choosing a single item. Formally, $\mathrm{div}(i, RL) = \mathrm{div}_{\mathrm{RxQuAD}}(i, RL)$ defined as:

$$\mathrm{div}_{\mathrm{RxQuAD}}(i, RL) = \sum_{a \in A} [p(a|u)p(rel|i, u, a) \prod_{j \in RL} (1 - p(rel|j, u, a)p(stop|rel))] \tag{2.11}$$

$p(rel|i, u, a)$ is the probability that user $u$ finds recommended item $i$ relevant when interested in aspect $a$. In the case of explicit features, this probability is obtained by mapping from relevance scores $s(u, i)$ using an exponential function [VCV12]. $p(stop|rel)$ is the probability that a user stops exploring a recommendation list conditional on finding a relevant item. Vargas observes that, to maximize $\alpha$-nDCG, the best value for $p(stop|rel)$ is approximately equal to the value of $\alpha$ [Var15].

#### 2.3.3.3 SxQuAD and SRxQuAD

Vargas & Castells define another two forms of intent-aware diversification, which they refer to as SxQuAD and SRxQuAD, this time based on combining sets of recommendations [VC13]. Specifically, given the set of items that a user $u$ has interacted with and a set of explicit features $\mathcal{F}$, they define a subprofile for each feature $f \in \mathcal{F}$: the subset of the items $u$ has interacted with that possess feature $f$. This means that there are as many subprofiles as there are features ($|\mathcal{F}|$). When there are many features (e.g. where features are user-generated tags), there is a scalability problem, which Vargas & Castells handle by using only a user's top features. Then they make recommendations to each subprofile. These sets of recommendations are combined using a modified version of either Eq. 2.8 or Eq. 2.11, which in turn is used for greedy re-ranking using Eq. 2.2, from which a top-$N$ can finally be recommended.

For SxQuAD, $p(i|u,a)$ in Eq. 2.8 is replaced by $p(i|u_f)$, where $u_f$ is the subset of items that the user has interacted with and that possess feature $f$:

$$\text{div}_{\text{SxQuAD}}(i, RL) = \sum_{f \in \mathcal{F}} [p(f|u)p(i|u_f) \prod_{j \in RL} (1 - p(j|u_f))] \tag{2.12}$$

$p(i|u_f)$ is estimated as:

$$p(i|u_f) = \frac{s(u_f, i)}{\sum_{j \in R_{u_f}} s(u_f, j)} \tag{2.13}$$

where $s(u_f, i)$ is the predicted score for item $i$ based on just subprofile $u_f$ and $R_{u_f}$ is the set of recommendations generated for subprofile $u_f$.

For SRxQuAD $p(rel|i,u,a)$ in Eq. 2.11 is replaced by $p(rel|i,u_f)$.

There is an apparent similarity between our new approach to diversification, SPAD and its variants (Chapters 4 and 6), and Vargas & Castell's SxQuAD and SRxQuAD. Both use subprofiles. But, this similarity is superficial; they differ in several ways. First, Vargas & Castells define subprofiles in terms of explicit features: the items a user has interacted with and that share a feature. By contrast, we extract subprofiles without reference to any meta-data, based instead purely on patterns of items that the user likes (Section 4.3). Second, Vargas & Castells make separate recommendations to each subprofile and then combine them. We do not do this at all. By contrast, we make a set of recommendations to the user and re-rank them in the style of xQuAD, i.e. by treating each subp-

rofile as an aspect and modeling the user's interests as a probability distribution over the aspects.

#### 2.3.3.4 c-pLSA and other intent-aware work

The advantage in intent-aware approaches such as xQuAD and its variants of using explicit aspects, such as movie genres, is their interpretability. A disadvantage is that they may be less accurate. The advantage, by contrast, of using implicit aspects, such as latent factors, is that they have been chosen for their predictive performance; their disadvantage is that they may be less interpretable [WH16]. Wasilewski & Hurley propose an intent-aware diversification method that is based on explicit aspects (and is hence interpretable) but in which the probabilities are learned (and hence are optimized for predictive performance) [WH16]. The learning is done by a constrained pLSA model [Hof04]. They call their approach c-pLSA. More recently, the same authors presented an intent-aware framework that uses a minimum variance criterion based on portfolio theory from finance [WH17].

Wasilewski and Hurley subsequently used their intent-aware personalized covariance within an item-based collaborative filtering recommender [WH18]. Unlike the work we have cited so far, this is not an example of greedy reranking; rather, intent-aware covariance is used in neighbourhood selection and weighting.

Recently, Anelli et al. have incorporated temporal aspects into an intent-aware diversification framework based on xQuAD [ABDN+17]. Noting that a user's intent can change during an interaction, they propose versions of Eq. 2.9 that include a temporal decay function on the one hand and that handle sessions on the other hand.

The advantage of intent-aware approaches is that they personalize the level of diversification. Probabilities differ between users since they are computed from each user's profile. In the next section, we review other work on personalized diversification.

### 2.3.4 Personalized diversification

There has been an amount of recent research on personalizing the level of diversification in IR, e.g. [VC12, LRD14], and in recommender systems, e.g. [SZW+12, DRTD17, PUG16, ZH08, VC13, WH17].

In IR, Vallet & Castells propose to diversify search results in a personalized way by taking intent-aware diversification methods such as xQuAD and introducing the user as an explicit random variable [VC12]. Liang et al. take a very different approach, treating diversification as a form of supervised learning using document terms and latent topics as features [LRD14]. They learn a model that estimates whether a document relates to a user's interests using a loss function that combines user relevance with diversification.

Turning to recommender systems, in the work of Shi et al., the level of diversification for a user increases with the uncertainty in the user's tastes [SZW+12]. There is more uncertainty if a user's profile is small or if a user's profile already exhibits a wide range of tastes, measured by the variances of the latent factors of the items in the profile.

Di Noia et al. model a user's propensity to diversity [DRTD17]: for each attribute of the items, they measure the entropy across the values of that attribute for items in the user's profile and then, from this, classify the user into one of four quadrants according to whether they have low or high entropy and whether they have small or large profiles. They then re-rank recommendation sets using modified versions of both Eq. 2.4 and Eq. 2.8. The modifications introduce quadrant-specific weights, thereby controlling the degree of diversification in a personalized way.

Puthiya Parambath et al. do not use greedy re-ranking [PUG16]. Instead, they recommend a set of items that *covers* the items in the user's profile. Relevance follows from a definition of coverage that takes into account the positively-rated items in the user's profile and the similarities between the items in the user's profile (using rating similarity). Diversification follows from the definition of coverage being submodular so that there is more gain in covering uncovered items than in covering ones that are already covered.

Eskandian et al. use a clustering approach for personalizing diversity [EMB17]. They do not use greedy re-ranking. Instead, they cluster the users based on the degree of diversity in their profiles and perform collaborative filtering indepen-

dently on the cluster of users. They measure the degree of diversity for users based on item categories (like genres in movies etc.).

Zhang & Hurley present the problem of maximizing the diversity of a recommendation set while maintaining the accuracy as constrained binary optimization problems, and solve those optimization problems [ZH08]. In other work, the same authors cluster the items in a user's profile, using rating similarities [ZH09]. They make recommendations to each partition and solve optimization problems to combine these recommendations into a final recommendation set in a way that balances relevance with diversity.

Teo et al. present a personalized, submodular diversification algorithm that re-ranks the most relevant items based on categories whose weights are learned and therefore personalized [TNH+16]. In online experiments, they show that their algorithms significantly improve click-through-rate and session duration.

Wu et al. present a personality-based greedy re-ranking approach to generating the recommendation list [WCZ18]. First, they conduct a user survey and find out that personality traits can significantly influence users' diversity preferences. Based on this finding, they develop a personality-based greedy re-ranking approach to improving the recommendation diversity. They compute weights for the personality traits of each user. Then, they use modified version of Eq. 2.2 for re-ranking the recommendation sets. For the diversity part of the equation they use personality weights that models the user's propensity towards diversity.

Abdollahpouri et al. present a personalized diversification re-ranking approach that manages popularity bias by increasing the representation of less popular items [ABM19]. Specifically, they adapt Santos et al.'s xQuAD [SMO10], proposing a variant that adds a personalized bonus to long-tail items. They determine the personalization factor based on each user's historical interest in long-tail items.

## 2.4 Calibrated Recommendations

Recently, Steck proposed the concept of calibrated recommendation [Ste18]. Calibrated and intent-aware recommendation have apparent similarities in that both try to cover the user's profile. However, they do differ. We postpone an in-depth comparison of the two to Chapter 7. Here, in our state-of-the-art chapter, we simply present the key concepts in calibrated recommendation.

Calibrated recommendation aims to produce a recommendation set that covers the user's different tastes and interests in proportion to the extent they occur in the user's profile [Ste18]. Consider a music domain, for example, in which a user listens to jazz 70% of the time and to rock music 30% of the time. Then calibrated recommendations directly aims to reflect these proportions in the final set of recommendations.[1]

In [Ste18], the degree of calibration, $\mathrm{C_{KL}}(p, q)$, is quantified by taking the Kullback-Leibler divergence between two probability distributions: the first, $p(f|u)$, is the distribution of features $f$ across the items in user $u$'s liked-item-set, $I_u^+$; the second, $q(f|u)$, is the distribution of features $f$ across the items in a recommendation set $RL$. These two distributions are defined in terms of another distribution, $p(f|i)$, which is the distribution of features for each item $i$. A simple definition is that, for each feature $f$ possessed by an item $i$, $p(f|i)$ will be equal and such that $\sum_f p(f|i) = 1$ [Ste18].

Specifically, Steck defines $p(f|u)$ as:

$$p(f|u) = \frac{\sum_{i \in I_u^+} w_{u,i} p(f|i)}{\sum_{i \in I_u^+} w_{u,i}} \tag{2.14}$$

In this definition, $w_{u,i}$ is a weight for item $i$. As an example, Steck suggests that the weight could be based on how recently item $i$ was consumed by user $u$. However, in the rest of his paper, he takes $w_{u,i} = 1$ for all $u$ and $i$, and we do the same in this thesis.

Steck defines $q(f|u)$ as:

$$q(f|u) = \frac{\sum_{i \in RL} w_{r(i)} p(f|i)}{\sum_{i \in RL} w_{r(i)}} \tag{2.15}$$

$w_{r(i)}$ denotes the weight of item $i$ due to its rank, $r(i)$ in $RL$, although again both in his paper, and in this thesis, $w_{r(i)} = 1$.

The degree of calibration is the Kullback-Leibler divergence of the two distributions, taking $p$ as the target distribution:

$$\mathrm{C_{KL}}(p, q) = KL(p||\tilde{q}) = \sum_f p(f|u) \log \frac{p(f|u)}{\tilde{q}(f|u)} \tag{2.16}$$

[1] This example simplifies by ignoring the possibility that a song might have more than one genre.

Because Kullback-Leibler divergence diverges if $q(f|u) = 0$ and $p(f|u) > 0$, Steck uses $\tilde{q}$ instead of $q$:

$$\tilde{q} = (1 - \alpha)q(f|u) + \alpha p(f|u) \tag{2.17}$$

with a small positive value of $\alpha$ so that $q \approx \tilde{q}$. Steck uses $\alpha = 0.01$, and we do the same.

Now that we can measure the degree of calibration, we can define an objective function. It is a linear combination of the relevance of the items in the recommendation set and the degree of calibration of that set:

$$RL^* = \underset{RL,|RL|=N}{\arg\max}\, (1 - \lambda)s(RL) - \lambda\, \mathrm{cal}(p, q) \tag{2.18}$$

We emphasize that, while $p$ is different for each user, $q$ is different for each recommendation set being considered in the equation.

Steck proves that Eq. 2.18 satisfies the conditions for a greedy re-ranking approach to find a $1 - \frac{1}{e}$ approximation to the optimal solution. The greedy re-ranking approach uses the following objective function:

$$f_{obj}(i, RL) = (1 - \lambda)s(u, i) + \lambda\, \mathrm{cal}(i, RL) \tag{2.19}$$

with $\mathrm{cal}(i, RL) = -(\mathrm{C_{KL}}(p, q(RL \cup \{i\})) - \mathrm{C_{KL}}(p, q(RL)))$.

## 2.5 Conclusions

In this chapter, we summarized the state-of-the-art in the topics of interest to our research, especially in the area of diversification of recommendations. We have explained intent-aware diversification approaches and a related concept to intent-aware approaches, calibrated recommendation, in detail.

In the next chapter, we give details of the experimental design that we followed throughout the experiments conducted in this thesis.

# Chapter 3

# Experiment Design

There are a lot of experimental results throughout this dissertation. In this chapter, we describe the datasets we have used in our experiments (Section 3.1). We also give the details of all baseline recommendation algorithms that we use (Section 3.2). Next, we present evaluation metrics that we use to evaluate the performances of our proposed methods and existing methods (Section 3.3). Finally, we explain the methodology we have followed in these experiments (Section 3.4). The experiment design presented in this chapter aims to be as clear as possible in order to provide reproducibility of our experiments and their results.

## 3.1 Datasets

The datasets we use are the MovieLens 1 Million (ML1M) dataset, MovieLens 20 Million (ML20M) dataset[1] [HK15], the LastFM (LFM) dataset,[2] the LibraryThing (LT) dataset [CdVR08], a Facebook (FB) dataset [FTTC+16], and the Taste Profile Subset dataset (TasteProfile)[3]. They cover different recommendation domains: movies, music, books, and social network. Table 3.1 summarizes the characteristics of the datasets. The popularity bias of the datasets is shown in Figure 3.1. The popularity bias of LFM is lower than the rest of the datasets: that of ML20M is the highest. We provide further details in the following subsections.

[1] http://grouplens.org/datasets/movielens/

[2] http://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html

[3] https://labrosa.ee.columbia.edu/millionsong/tasteprofile

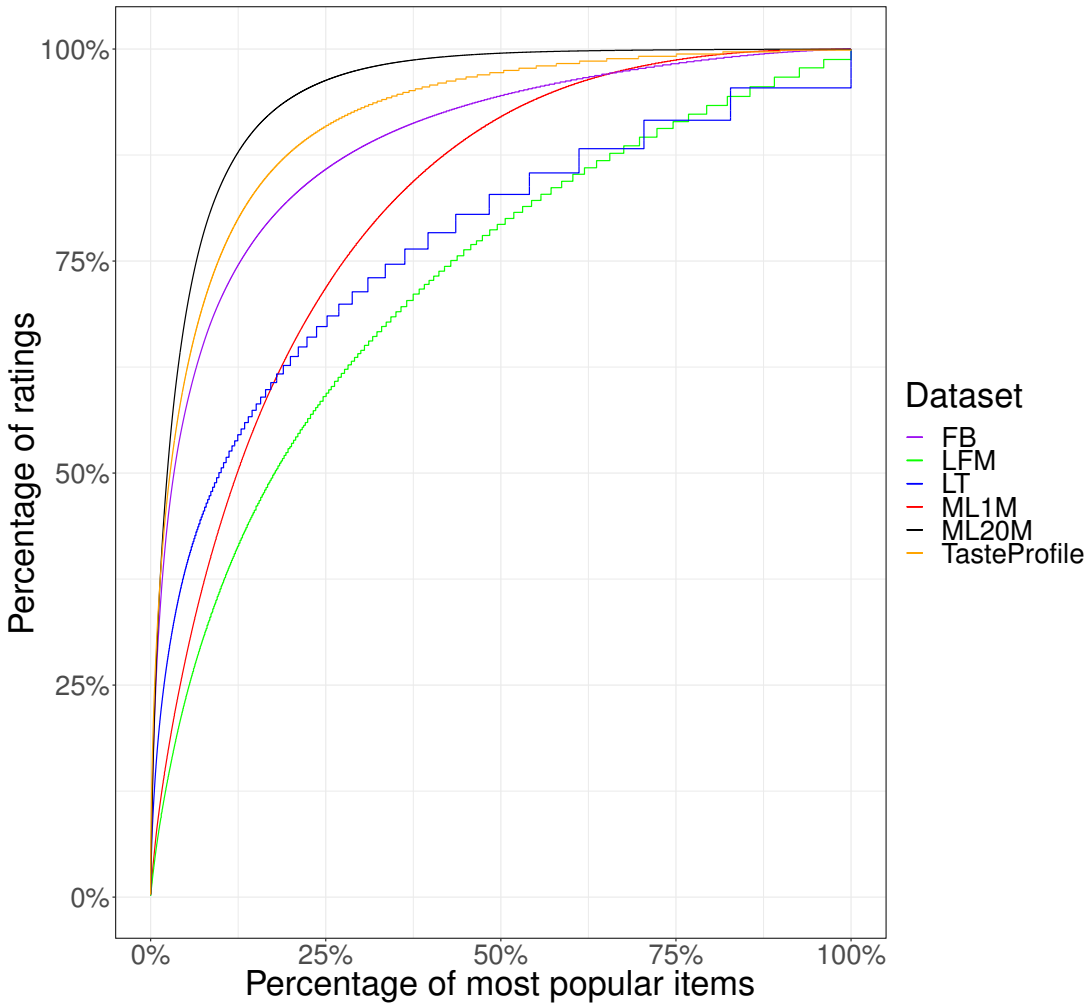



Figure 3.1: The popularity distribution of datasets used in the experiments.

Note that, in Chapter 8, we use the Million Playlist Dataset (MPD) [CLSZ18]. We used this dataset for Automatic Playlist Continuation (APC). Since APC is a rather different task, we follow a different experimental methodology. Hence, we explain that methodology and the MPD in Chapter 8.

### 3.1.1 MovieLens1M

The MovieLens1M (ML1M) dataset [HK15] is one of the most widely used datasets in recommender systems research. It has ∼1 million ratings on ∼3,700 movies by ∼6,000 users. As item features, movie genres are provided. There is a total of 18 genres.

Figure 3.2a shows the distribution of the ratings for the ML1M dataset. We can see that users tend to give higher ratings to the movies, rather than lower

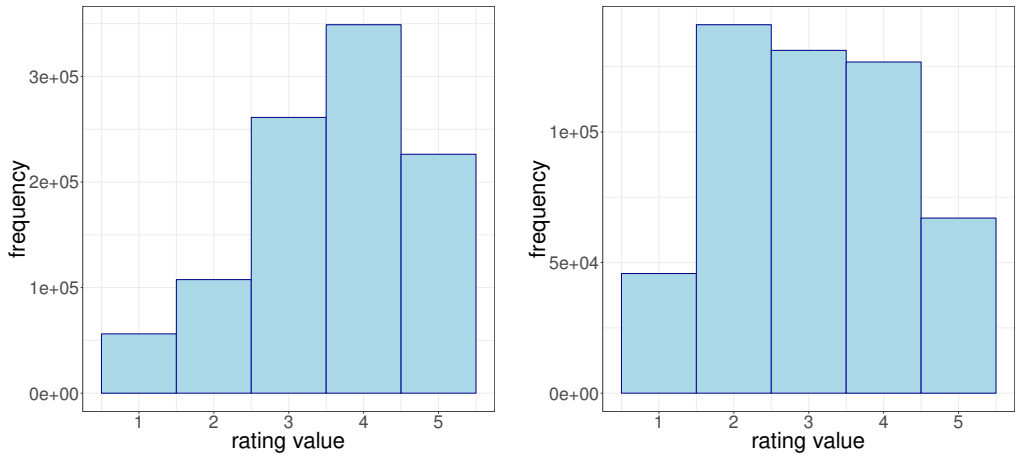
Table 3.1: Datasets

| | |
|---|---|
| ML1M | 6040 users 3706 movies ∼1M ratings |
| | avg. 165.6 ($\sigma = 192.74$) movies per user |
| | avg. 95.25 ($\sigma = 105.0$) liked movies per user |
| | 18 genres in total; avg. 1.65 per movie |
| LFM | 992 users 7280 artists ∼500k ratings |
| | avg. 515.94 ($\sigma = 475.14$) artists per user |
| | avg. 195.35 ($\sigma = 194.95$) liked artists per user |
| | 71833 tags in total; avg. 8 per artist |
| LT | 7279 users 37232 books ∼750K ratings |
| | avg. 102.95 ($\sigma = 132.68$) books per user |
| | avg. 66.43 ($\sigma = 82.56$) liked books per user |
| | 4800 tags in total; avg. 9.08 per book |
| FB | 104178 users 15374 pages ∼4.75M ratings |
| | avg. 45.94 ($\sigma = 49.43$) pages per user |
| | avg. 45.94 ($\sigma = 49.43$) liked pages per user |
| | 33,660 categories in total; avg. 10.62 per page |
| ML20M | 138493 users 26502 movies ∼20M ratings |
| | avg. 115.53 ($\sigma = 184.18$) movies per user |
| | avg. 58.02 ($\sigma = 81.81$) liked movies per user |
| | 19 genres in total; avg. 2.01 per movie |
| TasteProfile | 375749 users 190629 pages ∼19M ratings |
| | avg. 41.15 ($\sigma = 34.42$) songs per user |
| | 21 genres in total; avg. 1 per song |

ratings. This is the well-known and common bias that users tend to rate movies that they like.

### 3.1.2 LastFM

The LastFM (LFM) dataset is another widely used dataset in the literature. We modify it in the same way as in [KB16]. For example, the dataset is augmented with additional meta-data (namely, user-generated tags from the LastFM web site, a maximum of the 10 most popular labels for every artist), and artists having fewer than 3 LastFM tags and artists that were listened to by fewer than 20 users are filtered out. Then, the listening event frequencies in the LFM dataset are converted into ratings on the scale 1–5. Figure 3.2b shows the rating distribution after these modifications. After the modifications, there are ∼500K ratings on 7280 artists by 992 users.




(a) ML1M (b) LFM

(c) LT (d) ML20M

Figure 3.2: For ML1M, LFM, LT and ML20M datasets, distribution of the ratings.

### 3.1.3 LibraryThing

The LibraryThing (LT) dataset [CdVR08] was originally collected from the LibraryThing site[4], which is a cataloging and social networking site for book lovers. It has the ratings that users have given to books, and the tags they have assigned to them. We retrieved a maximum of the 10 most popular tags for every book and kept the tags that appeared in the profiles of at least 10 books. After this preprocessing, there are ∼750K ratings on ∼37K books by ∼7,000 users. The ratings on the website are on a 1 to 5 stars scale with steps of 0.5. The rating distribution of the LT dataset is shown in Figure 3.2c. Similar to the ML1M dataset, users tend to give higher ratings.

[4]https://www.librarything.com/

### 3.1.4 Facebook

Unlike the other datasets that we use in our experiments, the Facebook (FB) dataset contains implicit ratings ('likes') and it is multi-domain since it consists of user preferences for Facebook pages pertaining to movies, music and books. For the FB dataset, we keep users who have at least 10 likes and pages that are liked by at least 10 users; for meta-data, we use categories crawled from DBpedia [FTTC+16, TFTDNC16]. There is a total of 33,660 categories (10.62 per page on average), and ∼4.75M ratings by ∼105K users on 15,374 pages.

### 3.1.5 MovieLens 20 Million

In the original MovieLens 20 Million (ML20M) dataset, all users have at least 20 movies in their profile and movies have one or more of 19 genres. Some of the movies have no genre information. Since we measure diversity in the experiments, we eliminate movies that have no genre information. The resulting dataset has $\sim 20$ million explicit ratings made by $\sim 140$k users for $\sim 27$k movies. We use this dataset in Chapter 7, where we compare intent-aware diversification with Steck's Calibrated Recommendation (explained in Section 2.4) [Ste18]. We follow [Ste18] by binarizing the numeric ratings by dropping ratings lower than 4 stars and we eliminate movies that have no genre information. The resulting binarized dataset has $\sim 10$ million implicit ratings made by $\sim 140$k users for $\sim 21$k movies.

Figure 3.2d shows the distribution of the ratings for the ML20M dataset. We can see that, similar to ML1M dataset, users tend to give higher ratings to the movies.

### 3.1.6 Taste Profile Subset

TasteProfile contains counts of the number of times a user has listened to a song. Each song appears in the Million Song Dataset, from which we can take information about up to 21 genres.[5] We eliminate songs that have no genre information and users who have fewer than 20 songs in their profile. We also binarize the song counts so that we get an implicit dataset; the dataset is binarized in the same way in [LACB16, Aio13]. Note that, we take the binarized play

[5] http://www.ifs.tuwien.ac.at/mir/msd/partitions/msd-MAGD-genreAssignment.cls

counts since play counts for this dataset are not reliable and do not necessarily correlate with numeric ratings [Aio13]. The resulting dataset is comprised of $\sim 19$ million implicit ratings made by $\sim 375$k users for $\sim 190$k songs.

## 3.2 Recommendation Algorithms

In this section, we explain the baseline recommendation algorithms that we use in the experiments conducted in the following chapters. In this thesis we use collaborative filtering algorithms as they have been widely used and proven to be effective in terms of user satisfaction.

We use three recommenders that represent well-known variants of model-based collaborative filtering algorithms: matrix factorizaton, probabilistic latent semantic analysis and factorization machines. We use their RankSys[6] implementations.

For matrix factorization, we use a fast alternative least-squares matrix factorization recommender (MF) [PZT10] based on the implicit Matrix Factorization algorithm (iMF) of Hu et al. [HKV08], which factorizes the interaction matrix $\mathcal{R}$ into two lower dimension matrices $\mathcal{P} \in \mathbb{R}^{|U|,k}$ and $\mathcal{Q} \in \mathbb{R}^{|I|,k}$. Scoring function $s_{MF}(u,i)$, measuring how relevant item $i$ is to user $u$, is computed by using $\mathcal{P}$ and $\mathcal{Q}$ as:

$$s_{MF}(u,i) = \mathcal{P}_u.\mathcal{Q}_i^T \tag{3.1}$$

where $\mathcal{P}_u$ is the latent factor vector of size $k$ corresponding to user $u$ and $\mathcal{Q}_i$ is the latent factor vector of size $k$ corresponding to item $i$. The user and item latent factor matrices $\mathcal{P}$ and $\mathcal{Q}$ are learned by minimizing the following loss function by using alternating least squares (ALS):

$$\mathcal{L}_{iMF} = \sum_{u,i} c_{u,i}(\mathcal{P}_u.\mathcal{Q}_i^T - r_{u,i})^2 + \lambda \left( \sum_u ||\mathcal{P}_u||^2 + \sum_i ||\mathcal{Q}_i||^2 \right) \tag{3.2}$$

where $c_{u,i}$ is the confidence level, i.e. how confident we are about the preference of user $u$ for item $i$ based on the value of $c_{u,i}$. The values of $c_{u,i}$ are assigned as:

[6] http://ranksys.org/

$$c_{u,i} = \begin{cases} 1 + \alpha r_{u,i} & (u,i) \in \mathcal{R} \\ 1 & \text{otherwise} \end{cases} \tag{3.3}$$

$\lambda$ is the regularization parameter and $r_{u,i}$ is the rating given by user $u$ to item $i$ ($r_{u,i} = 0$ if $i$ has not been rated by $u$ yet). $\alpha$ controls the rate of increase in the confidence level $c_{u,i}$.

Pilaszy et al. [PZT10] minimize the same loss function by using fast ALS variants with a significantly lower computational complexity and their results show that their fast ALS variant offers a better trade-off between running time and accuracy. Because of its advantages over iMF [HKV08], we use Pilaszy et al.'s version of matrix factorization.

For latent semantic analysis, we use the probabilistic Latent Semantic Analysis (pLSA) of Hofmann [Hof04]. The idea is to introduce hidden variables $\mathcal{Z}$ with states $z$ for every $(u,i)$ pair such that the states $z$ will model a hidden cause for the interaction between $u$ and $i$. For this purpose the following mixture model is proposed:

$$P(u,i;\Theta) = \sum_z P(u,i,z) = \sum_z P(u|z)P(i|z)P(z) \tag{3.4}$$

This model results in the following scoring function:

$$s_{pLSA}(u,i) = P(u,i) = \sum_z P(u|z)P(i|z)P(z) \tag{3.5}$$

The model parameters $\Theta$ are fit by minimizing the empirical logarithmic loss:

$$\mathcal{L}_{pLSA}(\Theta) = \sum_{u,i} r_{u,i} log P(u,i) \tag{3.6}$$

Factorization Machines (FMs) are generic methods that are trained on a matrix $\mathbf{X} = [x_1, x_2, \ldots, x_n]$, where $x_i$ is the feature vector of $i$-th sample, and $n$ is the number of training samples. A second-order FM predicts values $\hat{y}$ as follows:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j>i}^{n} \langle v_i, v_j \rangle x_i x_j \tag{3.7}$$

where $w_0$ is the global bias, $w_i$ is the weight of the $i$-th feature and $v_i \in \mathbb{R}^k$ is the latent factor vector for the $i$-th feature. The weight of each feature pair $x_i x_j$ is given by the product of the corresponding latent feature vectors, $\langle v_i, v_j \rangle$.

FMs can be identical to MF models by using simple one-hot encodings of users and items:

$$x = \{\ldots, 0, \overbrace{1}^{x_i}, 0, \ldots, 0, \overbrace{1}^{x_j}, 0, \ldots\}$$

where $x_i = 1$ corresponds to user $i$, and $x_j = 1$ corresponds to item $j$. After dropping all zero features and biases in Eq. 3.7, it reduces to:

$$\hat{y}(x) = w_0 + w_i + w_j + v_i^T v_j \tag{3.8}$$

This is identical to a variation of matrix factorization that predicts how relevant an item is to a user, in which $v_i$ and $v_j$ will be the latent factor vectors of user and item respectively.

For Factorization Machines (FM), we use a factorization machine that uses Bayesian pairwise loss for ranking (BPR), that we refer as to FMBPR [Bay15]. In other words, it learns the latent factor vectors, $v_i$ and $v_j$ using the BPR loss function [RFGST09]. BPR makes the following pairwise assumption: users are more interested in items with which they interact than the remaining items with which they have not interacted. FMBPR creates a set of pairwise preference $D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$, where $I_u^+$ is the liked-item-set of user $u$. FMBPR's loss function is as follows:

$$\mathcal{L}_{FMBPR} = \sum_{(u,i,j) \in D_S} \ln \delta(x_{uij}) + \lambda ||\Theta||^2 \tag{3.9}$$

where $\Theta$ denotes the set of parameters and $\lambda$ control the regularization. $\delta(x) = \frac{1}{1+exp(-x)}$ and $x_{uij} := f(u, i) - f(u, j)$, where $f(.,.)$ indicates a scoring function.

## 3.3 Evaluation Metrics

In this section, we define the evaluation metrics that we use to measure the performance of different recommendation diversification algorithms used in the experiments of the following chapters of the thesis. We measure relevance,

diversity and calibration.

For relevance, we measure $\text{Precision}\,@N$ for $N = 10$:

$$\text{Precision}\,@N = \frac{\sum_{i=1}^{N} \text{rel}(u,i)}{N} \tag{3.10}$$

where $\text{rel}(u,i)$ means item $i$ is relevant to user $u$. We treat test set items with a rating of 4 or higher as being relevant for the ML1M, ML20M, LFM, and LT datasets. Any test set item in the FB and TasteProfile datasets is relevant, because FB and TasteProfile are implicit rating datasets.

There is no one ideal metric for diversity. Accordingly, we use five metrics: (i) $\alpha\text{-nDCG}$ [CKC+08], which is an aspect-aware version of $\text{nDCG}$; (ii) intent-aware expected reciprocal rank, $\text{ERR-IA}$ [AGHI09]; (iii) subtopic recall, $\text{S-recall}$ [ZCL03]; (iv) Intra-List Diversity ($\text{ILD}$, Eq. 2.4) [ZMKL05]; and (v) Expected Intra-List Diversity, $\text{EILD}$ [VC11], which is a rank- and relevance-aware version of $\text{ILD}$. We give their definitions below. For each, we use their RankSys implementations.

$\alpha\text{-nDCG}$ is based on $\text{nDCG}$ but it is aspect and redundancy-aware, which makes it a measure of diversity:

$$\alpha\text{-nDCG}(L) = \frac{1}{\alpha\text{-IDCG}} \sum_{i \in L} \left[ \frac{1}{\log_2(\text{r}(i,L)+1)} \sum_{f \in \mathcal{F}} \text{rel}(i|u,f) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - \alpha\,\text{rel}(j|u,f)) \right] \tag{3.11}$$

where $\alpha\text{-IDCG}$ is the highest possible value of $\alpha\text{-nDCG}(L)$, in which the recommendation set is made of ideally diversified relevant items, $L$ is the set of recommended items (of size $N$), $\text{r}(i,L)$ is the position of $i$ in $L$, and $\text{rel}(i|u,f)$ is 1 if item $i$ has feature $f$ and is relevant to user $u$ but 0 otherwise. $\alpha$ is the parameter that controls the penalty for redundancy. We use $\alpha = 0.5$. (Hence, following the argument from [Var15] given earlier, we use $p(stop|rel) = 0.5$ in RxQuAD and RSPAD too — see Sections 2.3.3.2 and 4.2.2, respectively.)

We also use $\text{ERR-IA}$ [AGHI09], which is an intent-aware measure:

$$\text{ERR-IA}(L) = \sum_{f \in \mathcal{F}} p(f|u) \sum_{i \in L} \frac{1}{\text{r}(i,L)} p(\text{rel}\,|i,u,f) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - p(\text{rel}\,|j,u,f)) \quad (3.12)$$

where $p(f|u)$ is the probability of feature $f$ given user $u$, and $p(\text{rel}\,|i,u,f)$ is the probability that user $u$ finds recommended item $i$ relevant when interested in feature $f$. When evaluating xQuAD and its variants, it is obvious what values to use for these probabilities: the ones computed by the xQuAD algorithm. But other algorithms, including SPAD and RSPAD, do not have these probabilities and therefore ERR-IA cannot be used directly [WH16]. What we do is to use the probabilities computed by xQuAD (Section 2.3.3.1), even when evaluating SPAD and its variants. This gives xQuAD an advantage, which must be kept in mind when looking at the results.

Subtopic Recall, S-recall, is a metric that measures how well the recommendation set covers the feature space [ZCL03]:

$$\text{S-recall}(L) = \frac{|\cup_{i \in L} \mathcal{F}_i|}{|\mathcal{F}|} \quad (3.13)$$

where $\mathcal{F}_i$ is the set of features of item $i$ and $\mathcal{F}$ is the set of all features.

Intra-List Diversity, ILD, measures the average pairwise distance of the items in a recommendation set [ZMKL05]. We presented its definition already as Eq. 2.4.

Expected Intra-List Diversity [VC11] is a rank- and relevance-aware version of Eq. 2.4:

$$\text{EILD}(L) = \sum_{i,j \in L, i \neq j} C_i \,\text{disc}(k_i)\,\text{disc}(k_j|k_i) p(\text{rel}\,|i,u) p(\text{rel}\,|j,u)\,\text{dist}(i,j) \quad (3.14)$$

where $\text{disc}(k_i) = \frac{1}{log(k_i+2)}$ is the rank discount for item $i$ at position $k$, and $\text{disc}(k_j|k_i) = \text{disc}(\text{max}(1, k_j - k_i))$ is a relative rank discount for an item $j$ at position $k_j$ knowing that position $k_i$ has been reached. $p(\text{rel}\,|i,u)$ is a binary relevance factor, the value of which is 1 if and only if in the test set item $i$ is a relevant item for user $u$. $\text{dist}(i,j)$ is the Jaccard distance between the fea-

tures of items $i$ and $j$. $C_i = \frac{C}{\sum_{j' \in L \setminus \{i\}} \text{disc}(k'_j|k_i)p(\text{rel}\,|j',u)}$ is a normalizing constant given $C = \frac{1}{|L|}$. Note that when there is no rank discount, no relative rank discount (i.e. when $\text{disc}(k_i) = 1$ and when $\text{disc}(k_j|k_i) = 1$) and when the measure is not relevance-aware (i.e. $p(\text{rel}\,|i,u) = 1$ and $p(\text{rel}\,|j,u) = 1$), then EILD is equivalent to ILD [VC11].

To measure the degree of calibration, we use $\text{C}_{\text{KL}}$, the definition of which has been already presented as Eq. 2.16.

We have already mentioned that xQuAD has an advantage in the case of ERR-IA. But, in fact, all five measures of diversity and the calibration metric are computed with respect to item features, $\mathcal{F}$. All may therefore favour recommenders that re-rank using those features, such as MMR [CG98], xQuAD [VCV11], RxQuAD [VCV12] and c-pLSA [WH16]. Our new methods, SPAD and its variants, make no use of the features at all and so they are at a disadvantage in our experiments. To allow for a more rounded view of the algorithms that we evaluate, we have adapted the diversity metrics and the calibration metric defined above to produce a corresponding set of metrics that use subprofiles instead of item features. We give the definitions of the adapted diversity metrics in Section 5.2, and the adapted calibration metric in Section 7.3.

## 3.4 Evaluation Methodology

In this section, we give the details of our evaluation methodology for all the experiments presented in the next four chapters. First, we give details of the how the data is split. Next, we explain how we optimize the hyperparameters of the baseline recommenders. Then, we we explain hyperparameter optimization of the re-ranking algorithms. Finally, we explain how we evaluate the performance of different approaches on the test sets.

### 3.4.1 Data split

For the ML1M, LFM, LT, FB, ML20M and TasteProfile datasets, we randomly partition the ratings into training, validation and test sets such that 60% of each user's ratings are in the training set, 20% of them are in the validation set and 20% are in the test set. For the smaller datasets (ML1M, LFM, LT and FB) results are averaged over five runs with different random splits, whereas for

the larger datasets (ML20M and TasteProfile) experiments are run with only one random split.

### 3.4.2 Hyperparameter values for baseline recommenders

Here, we explain how we optimize the hyperparameters of the baseline recommenders (Section 3.2). We emphasize that all of hyperparameter values are found using validation sets. We select hyperparameter values that optimize precision on the validation sets [VCV12]. All the baseline recommenders produce a ranked set of recommendations for all users who have data in the training and validation subsets. For each user $u$, for all items $i$ which are not in the training subset of the user, a score $s(u, i)$ is computed by the baseline recommender, and items are sorted in decreasing order by their computed scores to produce ranked set of recommendations; we select the top-$N$ recommendations, $N = 10$. Then, the resulting recommendations are evaluated in terms of Precision by considering items that appear in the validation subset of the user and which are relevant to the user. For relevancy, we consider items having a rating value above a threshold.

Table A.1 in the Appendix shows optimized hyperparameter values of each baseline recommendation algorithm on each dataset.

### 3.4.3 Hyperparameter values for re-ranking algorithms

Some of the re-ranking algorithms also have hyperparameters. Here we explain how we choose their value.

After optimizing the hyperparameter values of the baseline recommenders, for each user, we generate a recommendation set $RS$, where $|RS| = 100$ using the baseline recommender with its best hyperparameter values. We re-rank $RS$ to produce ranked list $RL$ using each of the re-ranking methods with each of their combinations of hyperparameter values. Then, from each $RL$, we select the top-$N$ recommendations, $N = 10$. Finally, for each re-ranking method, we select hyperparameter values that give the best $\alpha$-nDCG on the validation set. For re-ranking methods we choose $\alpha$-nDCG since it is a metric that combines accuracy and diversity and the approaches proposed in the following chapters aim to increase the accuracy and diversity of the baseline recommender algorithms. In

the Appendix to this thesis, we give the values of these hyperparameters. Unless explicitly indicated, all re-ranking methods follow the methodology given here to optimize their hyperparameters.

### 3.4.4 Evaluation on test sets

After optimizing the hyperparameters of the re-ranking algorithms, to generate the final set of top-$N$ recommendations, we train the baselines using their selected hyperparameter values on the union of the training and validation sets and, for each user, generate a recommendation set $RS$, where $|RS| = 100$. Then, we re-rank each $RS$ to produce ranked lists $RL$ using each of the re-ranking methods with their selected hyperparameter values. Then, from each $RL$, we select the top-$N$ recommendations, $N = 10$, and measure the evaluation metrics that we described in Section 3.3.

## 3.5 Conclusions

In this chapter, we have explained the details of the experimental design followed throughout the experiments conducted in the next four chapters of this thesis.

In the following chapter, we will explain our proposed approaches to recommendation diversification using subprofiles.

# Chapter 4

# Subprofile-Aware Diversification

Intent-aware diversification seeks to ensure that the set of recommended items covers the different tastes and interests revealed by the user's profile but using a formulation that takes item relevance into account [VCV11] (see Section 2.3.3 for the details). Consider, for instance, a music recommender. A user who only likes jazz and blues will receive jazz and blues recommendations; another user who likes jazz, blues and some classical will receive jazz, blues and classical music recommendations similarly.

The most common way to characterize a user's tastes and interests is as a probability distribution over so-called *aspects* of the items. Aspects are either explicit item features or implicit item features such as the latent factors found, e.g., by a matrix factorization recommender. Since the aspect probabilities may differ across users, diversity is personalized to a certain extent.

In this chapter, we explain our approach, Subprofile Aware Diversification (SPAD), in detail. Like other intent-aware approaches, SPAD aims to ensure that the final set of recommendations covers the aspects revealed by the user's profile to a certain extent. In the work on intent-aware diversification that we described earlier (Chapter 2), aspects were often based on explicit item features. In SPAD, by contrast, aspects are user *subprofiles*, i.e. subsets of the items that the user likes.

In this chapter, first, we try to motivate the use of subprofiles as aspects. Next, we explain SPAD's formulation in detail. Then, we define eight methods for detecting subprofiles. Finally, we give a comprehensive empirical comparison of all eight methods for detecting subprofiles on three different datasets.

## 4.1 Why Subprofiles as Aspects?

Intent-aware methods for recommendation diversification take inspiration from the work done in IR by establishing an analogy between query intentions with tastes and interests revealed by the user's profile [VCV11, VCV12, WH16]. User tastes or interests vary and they are heterogeneous, therefore it is a complex task to capture them. As we mentioned before, the most common way to characterize a user's tastes and interests is as a probability distribution over so-called *aspects* of the items, usually by using explicit item features.

Using item features for aspects brings several problems. In some domains, features are not available. Where they are available, the features themselves may be noisy (especially in the case of user-generated tags) or they may be inconsistently applied. They are often not very fine-grained; for example, the well-known MovieLens system describe movies using just 18 genres [HK15]. In domains such as movies and music where tastes and interests are complex, subtle and highly subjective, it may not be possible to fully represent those tastes and interests by a probability distribution over a small set of item features.

There is an alternative to defining tastes and interests in terms of item features. Patterns in user interactions with items can be seen as indications of the distinct tastes and interests of the users [Kul18]. Recommender systems already record these interactions as explicit or implicit feedback in their ratings matrix. Patterns of interaction with items, rather than item features, could form the basis of the aspects of an intent-aware approach to diversification.

We propose to define aspects as subprofiles of a user's profile, by considering the patterns of user interactions with items. In defining these subprofiles, we make no use of item features at all. Using subprofiles has two advantages over using item features. First, they can be used in domains where item features are not available or are unreliable. Second, as we show in different chapters of this thesis, across multiple datasets and across almost all settings, re-ranking approaches that use subprofiles instead of item features improve both accuracy and diversity. They suffer much less from the relevance/diversity trade-off. We hypothesize that this is because subprofiles are defined in terms of users' interactions with items and tend to give us more fine-grained aspects than item features: if the user likes $m$ items, then there are $2^m - 1$ possible subprofiles, i.e. all subsets except the empty set.

## 4.2 SubProfile Aware Diversity

Subprofiles are defined in terms of items that the user likes (the liked-item-set), i.e. ones to which she has given a positive rating, $I_u^+$.

User $u$'s candidate subprofiles are simply the non-empty subsets of $I_u^+$. In SPAD, we select from among these candidate subprofiles ones that capture the different tastes and interests of the user. We will denote user $u$'s set of subprofiles by $\mathcal{S}_u^*$. Different subprofiles $S \in \mathcal{S}_u^*$ can be of different lengths; the number of subprofiles $|\mathcal{S}_u^*|$ can differ across users. We have explored several ways of deciding which of the candidate subprofiles best capture the user's tastes and interests. We postpone our presentation of this to the next section. For now, we will show how SPAD uses the subprofiles $\mathcal{S}_u^*$ to re-rank recommendations.

We produce a set of recommendations $RS$ using some baseline recommender. This can be any recommender that produces relevance scores, $s(u,i)$, for the items that it recommends. The set $RS$ is greedily re-ranked (Algorithm 1) using the objective function given as Eq. 2.2 with $\mathrm{div}(i, RL) = \mathrm{div}_{\mathrm{SPAD}}(i, RL)$ (Eq. 4.1 below), which is analogous to $\mathrm{div}_{\mathrm{xQuAD}}(i, RL)$ (Eq. 2.8).

### 4.2.1 SPAD

What differs between SPAD and other forms of intent-aware diversification is the computation of the probabilities used in Eq. 2.8. Given that aspects are now subprofiles, we will write $p(S|u)$ and $p(i|u,S)$ instead of $p(a|u)$ and $p(i|u,a)$ for $S \in \mathcal{S}_u^*$, resulting in:

$$\mathrm{div}_{\mathrm{SPAD}}(i, RL) = \sum_{S \in \mathcal{S}_u^*} [p(S|u)p(i|u,S) \prod_{j \in RL} (1 - p(j|u,S))] \tag{4.1}$$

Analogously to Eq. 2.9, $p(S|u)$ can be estimated as:

$$p(S|u) = \frac{|S|}{\sum_{S' \in \mathcal{S}_u^*} |S'|} \tag{4.2}$$

$p(i|u,S)$, the probability of choosing $i$ from a set of recommendations $RS$ given subprofile $S$ of user $u$, can be estimated as:

$$p(i|u,S) = \frac{\mathbb{1}(i,S)s(u,i)}{\sum_{j \in RS} \mathbb{1}(j,S)s(u,j)} \tag{4.3}$$

But here there is a problem. We want $\mathbb{1}(i, S)$ to be 1 when item $i$ is 'related to' subprofile $S$, and 0 otherwise. We cannot just use membership ($i \in S$), because $i$ is a candidate recommendation and therefore will not in general already be a member of the user's profile or its subprofiles. Accordingly, in SPAD we define $\mathbb{1}(i, S)$ as follows:

$$\mathbb{1}(i, S) = \begin{cases} 1 & \text{if } i \in \bigcup_{j \in S} \mathrm{KNN}(j) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $\mathrm{KNN}(j)$ is the set of $j$'s $k$-nearest-neighbours in $I$. In other words, $i$ must be a neighbour of a member of $S$. (We have tried a simpler version of Eq. 4.4, where we use similarity more directly but, in preliminary experiments, it performed less well. We may explore it further in future.)

### 4.2.2 RSPAD, SSPAD and SRSPAD

Analogously to the relationship between xQuAD and RxQuAD [VCV12], we can define RSPAD, which is a variant of SPAD, by replacing aspects $a$ by subprofiles $S$ in Eq. 2.11.

Similarly, we can also define two further approaches, SSPAD and SRSPAD, which are analogous to SxQuAD and SRxQuAD (Section 2.3.3.3). SSPAD and SRSPAD first generate recommendations for each subprofile $S \in \mathcal{S}_u^*$. Then, they combine these recommendations in the same way that SxQuAD and SRxQuAD do this [VC13].

In our experiments, SSPAD and SRSPAD did not work well, so we do not give any further details, nor do we show their results in this thesis.

In the next section, we explain how we compute the subprofiles.

## 4.3 Subprofile Detection

In the previous section, we explained how SPAD and RSPAD re-rank a recommendation set $RS$ generated by a baseline recommender using a user's subprofiles. What this does not yet explain is how we compute the subprofiles. Here, we present eight different subprofile detection methods. We group them into three and summarize them in Table 4.1. The following sections present the eight methods in detail.

Table 4.1: Subprofile detection methods

| Description | Name |
|---|---|
| Candidate subprofiles are based on the nearest-neighbours of $u$'s liked-item-set | NN-1 |
| | NN-2 |
| A recommender generates a set of top-$n$ recommendations from $u$'s unrated items. Candidate subprofiles are explanations for the recommendations | IB+ |
| | DAMIB |
| | DAMIB-COVER |
| A recommender generates a ranked list of $u$'s unrated items. Candidate subprofiles are explanations for items in the ranked list, such that the explanations cover $I_u^+$ | $\text{IB+}_{\text{cp}}$ |
| | $\text{DAMIB}_{\text{cp}}$ |
| | $\text{DAMIB-COVER}_{\text{cp}}$ |

## 4.3.1 Subprofile detection from nearest-neighbours of liked items

The subprofile detection approaches that we explain in this section rest on the intuition that similar items in a user's liked-item-set will tend to be in the same subprofile. We present two approaches that use an item's nearest-neighbours to achieve this, referring to them as NN-1 and NN-2.

### 4.3.1.1 NN-1

For each item in the user's liked-item-set $i \in I_u^+$, we form a candidate subprofile $S_u^i$ that contains $i$ itself and any other items $j$ in the user's liked-item-set $j \in I_u^+, j \neq i$ that have item $i$ as one of their $k$-nearest-neighbours:

$$S_u^i = \{j \in I_u^+ : i \in \text{KNN}(j) \wedge i \neq j\} \cup \{i\} \tag{4.5}$$

$\text{KNN}(j)$ contains the top-$k$ items $j' \in I$ whose similarity to $j$, $\text{sim}(j, j')$, is highest. For similarity, $\text{sim}(j, j')$, we use cosine similarity between $j$ and $j$'s ratings in $R$.

It follows that the number of candidate subprofiles is the same as the number of items in the liked-item-set. But the candidate subprofiles themselves can be of any length between 1 and $|I_u^+|$.

Let $\mathcal{S}_u$ be the set of candidate subprofiles. We prune the candidates to obtain the final set of subprofiles for this user, $\mathcal{S}_u^*$. Specifically, we define $\mathcal{S}_u^*$ to be those members of $\mathcal{S}_u$ that do not contain any other members of $\mathcal{S}_u$:

$$\mathcal{S}_u^* = \{S \in \mathcal{S}_u : \neg\exists S' \in \mathcal{S}_u \wedge S \subset S'\} \tag{4.6}$$

We obtain $\mathcal{S}_u^*$ from $\mathcal{S}_u$ by sorting the elements of $\mathcal{S}_u$ in descending order of size and greedily retaining those that are not subsets of any already chosen. This pruning step is in fact used in all of the subprofile detection approaches that we explain in this chapter.

#### 4.3.1.2 NN-2

A simple alternative to NN-1 suggests itself. Still taking each $i \in I_u^+$ in turn, instead of finding other members of $j \in I_u^+$ which have $i$ among their nearest neighbours, we include $j$ in $i$'s candidate subprofile if $j$ is in $i$'s nearest neighbours:

$$S_u^i = \{j \in I_u^+ : j \in \text{KNN}(i) \wedge i \neq j\} \cup \{i\} \tag{4.7}$$

It follows that there is still one candidate subprofile per member of $I_u^+$. But now the length of a candidate subprofile is at most $1 + k$.

The candidates are pruned as before using Eq. 4.6.

Both NN-1 and NN-2 introduce a hyperparameter, namely $k$, the number of neighbours to use. This is in addition to, and different from, the hyperparameter $k$ used in the indicator function in Eq. 4.4. To distinguish them, we refer to the latter using $k_{ind}$ and the former by $k_{nn}$.

### 4.3.2 Subprofile detection from the explanations of top-$n$ recommendations

In this section, we present three further approaches to subprofile detection, which we designate IB+, DAMIB and DAMIB-COVER. They share the following intuition: a subprofile can be an explanation of a recommendation. These approaches generate a set of top-$n$ recommendations for $u$, each with an explanation. It is important to emphasize that $u$ is not shown these recommendations. Generating these recommendations is simply a step *within* the process of detecting subprofiles. Explanations are subsets of $u$'s liked-item-set. These explanations are the candidate subprofiles. Finally, they prune the candidate subprofiles in the same way as NN-1 and NN-2, using Eq. 4.6.

The recommender that IB+, DAMIB and DAMIB-COVER use is an item-based nearest-neighbours recommender [DK04]. This is chosen because it has a straightforward way of defining explanations and these explanations are subsets

of the user's liked-item-set (see below), which means that we can treat the explanations as candidate subprofiles.

These three methods introduce two more hyperparameters, namely $n$ and $k$. $n$ is the number of recommendations and, since all three methods use item-based nearest-neighbours recommenders (see below), $k$ is the number of neighbours used by these recommenders. This is yet another different $k$. When we need to distinguish it from the others, we will use $k_{IB}$.

In these three methods, since there is one candidate subprofile per recommendation, it follows that there will be $n$ candidate subprofiles. Since the candidate subprofiles are explanations and drawn from the liked-item-set, their length will be between 1 and $|I_u^+|$ inclusive.

We will now present IB+, DAMIB and DAMIB-COVER in turn.

#### 4.3.2.1 IB+

IB+ is an item-based nearest-neighbours recommender for implicit ratings (i.e. for positive-only ratings) [DK04]. It recommends those candidate items that are most similar to the items in $u$'s liked-item-set. Candidate items are ones that are not in the user's liked-item-set, $I \setminus I_u^+$. For each candidate item $i$, IB+ finds items in the user's liked-item-set that have the candidate items as one of their $k$-nearest-neighbours:

$$S_u^i = \{j \in I_u^+ : i \in \mathrm{KNN}(j)\} \tag{4.8}$$

Here, the set $S_u^i$ is the explanation for why candidate $i$ should be recommended: items that $u$ likes and that are similar to $i$.

IB+ scores each candidate by taking the sum of the similarities of the candidate to the items in $S_u^i$:

$$s_{\mathrm{IB+}}(u,i) = \sum_{j \in S_u^i} \mathrm{sim}(i,j) \tag{4.9}$$

The candidate subprofiles ($\mathcal{S}_u$) are the explanations ($S_u^i$) for the the $n$ candidate items whose scores are highest. These are pruned using Eq. 4.6 to give the final subprofiles ($\mathcal{S}_u^*$).

#### 4.3.2.2 DAMIB

The DAMIB and DAMIB-COVER recommender systems were originally developed by Verstrepen & Goethals for recommending to shared accounts [VG15]. They assume that a group of people, such as a family, share a single account, e.g. a single online shopping or TV-streaming account. The user profile for this account therefore captures the various tastes of several individual family members. Informally, the goal of DAMIB and DAMIB-COVER is to recommend a set of $n$ items that includes recommendations targeted toward subprofiles (corresponding to the different family members) and to avoid recommending items that are overly general, which might be suitable for the profile as a whole but which do not suit the individuals who share the account.

What DAMIB and DAMIB-COVER are really doing is recommending to different subprofiles within a single account. It can be a shared account but it can just as well be a single-user account. In the case of a single-user account, the different subprofiles will represent that user's different tastes or interests, and there will be recommendations targeted at each of these tastes or interests — as many as can be accommodated in a top-$n$ recommendation list.

We use DAMIB and DAMIB-COVER in the same way as we use IB+ above, i.e. as a recommender whose explanations will be the subprofiles that we use in SPAD. Even though we are re-purposing these two recommender system to use them for subprofile extraction, we will continue to refer to them here as DAMIB and DAMIB-COVER.

Consider the powerset of $u$'s liked-item-set, $2^{I_u^+}$. This is the set of all of $u$'s possible subprofiles. DAMIB computes the relevance of each candidate item $\{i \in I : r_{u,i} = \perp\}$ to each member of the powerset $S \in 2^{I_u^+}$. It does this in a similar way to the IB+ recommender: for each item $j$ in $S$ whose set of $k$-nearest-neighbours contains candidate item $i$, the score for $i$ is increased by its similarity to $j$:

$$s_{DAMIB}(S, i) = \sum_{j \in S, i \in KNN(j)} \text{sim}(i, j) \tag{4.10}$$

The relevance of a candidate item $i$ to $u$, $s_{DAMIB}(u, i)$, is then based on the highest of the $s_{DAMIB}(S, i)$:

$$s_{DAMIB}(u, i) = \max_{S \in 2^{I_u^+}} \frac{1}{|S|^{0.75}} s_{DAMIB}(S, i) \tag{4.11}$$

As can be seen in Eq. 4.11, DAMIB multiplies the scores by $\frac{1}{|S|^{0.75}}$, where the value 0.75 is chosen based on experimental results in [VG15]. Since $p > 0$, this has the effect of penalizing scores that come from larger subprofiles. The intuition here is to give higher scores to candidates with high similarities to a few items than to candidates with small similarities to many items. The member of the powerset that maximizes $s(u, i)$ is the corresponding explanation:

$$S_u^i = \arg\max_{S \in 2^{I_u^+}} \frac{1}{|S|^p} s(S, i) \tag{4.12}$$

Eq. 4.11 and Eq. 4.12 imply exponential amounts of computation: for each candidate item, computing the maximum of $2^m$ scores, where $m = |I_u^+|$. However, Verstrepen & Goethals use a prefix property to eliminate some of the computations and prove that it can be computed in $O(m \log m)$ time.

DAMIB would recommend the $n$ candidate items with the highest scores, computed as $s_{DAMIB}(u, i)$. We, instead, use their explanations as candidate subprofiles, which we prune using Eq. 4.6.

#### 4.3.2.3 DAMIB-COVER

Verstrepen & Goethals identify a problem with using the DAMIB algorithm to recommend items to the users of a shared account. If we recommend the top-$n$ items with the highest $s_{DAMIB}(u, i)$, it may be the case that this top-$n$ fails to include any recommendations for some of the users who share the account. DAMIB-COVER works exactly as DAMIB except, instead of recommending the $n$ candidates with the highest $s_{DAMIB}(u, i)$, DAMIB-COVER feeds the candidate items and their scores into a coverage algorithm that attempts to maximize the number of users who have at least one recommended item in the top-$n$. In essence, this part of the system forms a top-$n$ from the candidates by including an item in the top-$n$ only if its explanation includes at least one item that is not a member of the unions of the explanations of the higher-ranked items.

As we did with DAMIB, we adapt DAMIB-COVER so that, instead of returning $n$ recommendations, it returns $n$ candidate subprofiles (which are the explanations for those recommendations). We show how we do this in Algorithm 2. As usual, the candidate subprofiles ($\mathcal{S}_u$) are then pruned using Eq. 4.6 to give the final subprofiles ($\mathcal{S}_u^*$).

**Algorithm 2** DAMIB-COVER$(u, n)$

**Input:** $u \in U$, $n \in \mathbb{N}^+$
**Output:** $n$ candidate subprofiles from $I_u^+$

```
1: Use DAMIB to compute s_DAMIB(u,i) and S_u^i for all i for which r_{u,i} = ⊥
2: Produce ranked list t of all i for which r_{u,i} = ⊥ in descending order of s(u,i),
   where t[r] refers to the item in position r
3: r ← 1
4: Covered ← {}
5: S_u ← {}
6: while |S_u| < n do
7:     i ← t[r]
8:     if |S_u^i \ Covered| ≥ 1 then
9:         insert S_u^i into S_u
10:        Covered ← Covered ∪ S_u^i
11:        remove i from t
12:        if Covered = I_u^+ then
13:            Covered ← {}
14:            r ← 1
15:    else
16:        r ← r + 1
17:        if r > |t| then
18:            Covered ← {}
19:            r ← 1
20: return S_u
```

### 4.3.3 Subprofile detection using profile coverage

As methods for subprofile detection, IB+, DAMIB and DAMIB-COVER have the weakness that we must decide in advance the maximum number of candidate subprofiles, $n$, corresponding to the $n$ recommendations. We decided to design variants of IB+ DAMIB and DAMIB-COVER that would not be constrained in this way. These three variants, designated IB+$_{\text{cp}}$, DAMIB$_{\text{cp}}$ and DAMIB-COVER$_{\text{cp}}$, attempt to find candidate subprofiles that cover a certain percentage of the user's liked-item-set. This is done by Algorithm 3.

Algorithm 3 takes in four inputs, $u$, $cp$, $alg$ and $cover$. $u$ is the user. $cp \in (0, 1]$ is the parameter that controls how much of the user's liked-item-set we want the subprofiles to cover. For example, if $cp = 0.5$, we want the candidate subprofiles to contain at least 50% of the items in the liked-item-set. This can be seen in line 12 of the algorithm: we break from the loop when the subprofiles cover enough of $I_u^+$. $alg$ determines which method we want to use when scoring the candidate items, either IB+ (Eq. 4.9) or DAMIB (Eq. 4.11). Finally, $cover$ is

**Algorithm 3** $\mathrm{COVER}(u, cp, alg, cover)$

**Input:** $u \in U$, $cp \in (0, 1]$, $alg \in \{\mathrm{IB+}, \mathrm{DAMIB}\}$, $cover \in \{true, false\}$
**Output:** candidate subprofiles from $I_u^+$

1: Use $alg$ to compute $s_{DAMIB}(u, i)$ and $S_u^i$ for all $i$ for which $r_{u,i} = \perp$
2: Produce ranked list $t$ of all $i$ for which $r_{u,i} = \perp$ in descending order of $s(u, i)$, where $t[r]$ refers to the item in position $r$
3: $r \leftarrow 1$
4: $Covered \leftarrow \{\}$
5: $\mathcal{S}_u \leftarrow \{\}$
6: **while** $true$ **do**
7: $\quad i \leftarrow t[r]$
8: $\quad$ **if** $cover \wedge |S_u^i \setminus Covered| < 1$ **then**
9: $\quad\quad$ **continue**
10: $\quad$ insert $S_u^i$ into $\mathcal{S}_u$
11: $\quad Covered \leftarrow Covered \cup S_u^i$
12: $\quad$ **if** $|Covered| \geq cp \times |I_u^+|$ **then**
13: $\quad\quad$ **break**
14: $\quad r \leftarrow r + 1$
15: $\quad$ **if** $r > |t|$ **then**
16: $\quad\quad$ **break**
17: **return** $\mathcal{S}_u$

Table 4.2: Subprofile detection using profile coverage

| Subprofile detection method | Method call |
|---|---|
| $\mathrm{IB+_{cp}}$ | $\mathrm{COVER}(u, cp, \mathrm{IB+}, false)$ |
| $\mathrm{DAMIB_{cp}}$ | $\mathrm{COVER}(u, cp, \mathrm{DAMIB}, false)$ |
| $\mathrm{DAMIB\text{-}COVER_{cp}}$ | $\mathrm{COVER}(u, cp, \mathrm{DAMIB}, true)$ |

a Boolean which, if true, will additionally apply DAMIB-COVER's coverage criterion. This can be seen in line 8 of the algorithm: if $cover$ is $true$, we ignore the candidate subprofile if it is not different enough from what has already been covered. The key point, in summary, is that this algorithm does not have to keep looping until it finds $n$ candidate subprofiles. It stops as soon as its coverage criterion is satisfied.

By the way we call Algorithm 3, we obtain our three new detection methods, $\mathrm{IB+_{cp}}$, $\mathrm{DAMIB_{cp}}$ and $\mathrm{DAMIB\text{-}COVER_{cp}}$, as shown in Table 4.2. $\mathrm{IB+_{cp}}$, $\mathrm{DAMIB_{cp}}$ and $\mathrm{DAMIB\text{-}COVER_{cp}}$ replace hyperparameter $n$ by hyperparameter $cp$. They produce a number of candidate subprofiles that is not constrained to be $n$ (unlike IB+, DAMIB and DAMIB-COVER); instead, there can be up to $|I_u^+|$ of them (like NN-1 and NN-2).

# 4.4 Comparison of Subprofile Detection Methods

## 4.4.1 Qualitative comparison

We have presented eight subprofile detection methods. We compare them empirically in Section 4.4.2.2. Here, we compare them in a more qualitative way.

- *Algorithmic*. Referring to Table 4.1, we can see a major algorithmic difference: the first two methods (NN-1 and NN-2) use the nearest-neighbours of items the user likes; the other six methods predict scores using a recommender algorithm (whose recommendation are never shown to the user but whose explanations are candidate subprofiles). NN-1 and NN-2 are therefore considerably simpler from an algorithmic point of view and run faster.
- *Methodological*. The eight methods differ in their hyperparameters. All have $k_{ind}$ for use by Eq. 4.4. But NN-1 and NN-2 have just one more hyperparameter, $k_{nn}$, the number of neighbours to use in Eq. 4.5 and Eq. 4.7 respectively. The other six methods, all of which use an item-based recommender algorithm, share hyperparameter $k_{IB}$, which is used within the recommender algorithm (Eq. 4.8). Then, IB+, DAMIB and DAMIB-COVER additionally have hyperparameter $n$, the number of recommendations to generate (and hence the number of candidate subprofiles), whereas $\text{IB+}_{\text{cp}}$, $\text{DAMIB}_{\text{cp}}$ and $\text{DAMIB-COVER}_{\text{cp}}$ have hyperparameter $cp$, the proportion of $I_u^+$ to cover, instead of $n$.
- *Information used*. NN-1 and NN-2 compute subprofiles directly from the user's liked-item-set and item similarities. The other approaches are indirect since they produce subprofiles from recommendations for items that are not in the user's profile. This difference might even result in some strange behaviour: when a new item is rated by user's other than $u$, in the methods that find subprofiles from recommendations for items unseen by $u$, the subprofiles might change, whereas they are less likely to change in the case of NN-1 and NN-2 (unless item similarities change significantly).

It seems then that, from a qualitative point of view, there is a clear preference for NN-1 and NN-2.

Table 4.3: Subprofile statistics

| | |
|---|---|
| ML1M | avg. 51.09 ($\sigma = 61.5$) subprofiles per user |
| | avg. len of subprofiles is 7.78 ($\sigma = 6.25$) |
| | avg. sim of subprofiles is 0.0379 |
| LFM | avg. 134.37 ($\sigma = 161.41$) subprofiles per user |
| | avg. len of subprofiles is 30.08 ($\sigma = 28.11$) |
| | avg. sim of subprofiles is 0.1045 |
| LT | avg. 32.49 ($\sigma = 46.71$) subprofiles per user |
| | avg. len of subprofiles is 8.8 ($\sigma = 10.72$) |
| | avg. sim of subprofiles is 0.044 |

### 4.4.2 Experimental comparison

In this section, we compare the performance of the eight subprofile detection methods. We explained the experimental setup and evaluation methodology in detail in Chapter 3. There are many hyperparameters for different subprofile detection approaches. We show all the hyperparameter values in the Appendix to this thesis (See Table A.2).

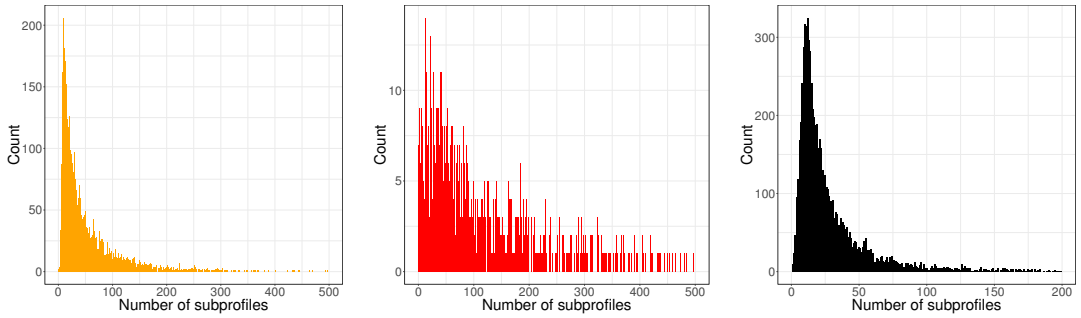
We use three datasets: the MovieLens 1M (ML1M) dataset, the LastFM (LFM) dataset and the LibraryThing (LT) dataset. For the characteristic of the datasets, see Chapter 3.

SPAD uses greedy re-ranking, therefore it needs a baseline recommender, whose recommendation sets are re-ranked. We use the baseline recommenders that we explained in Chapter 3: probabilistic latent semantic analysis (pLSA) [Hof04], a fast alternative least-squares matrix factorization recommender (MF) [PZT10], and a factorization machine that uses Bayesian pairwise loss for ranking (FM-BPR) [Bay15].

Three baselines paired with nine re-ranking approaches (SPAD with eight different subprofile detection approaches and the baseline itself) gives 27 systems to compare on each dataset.

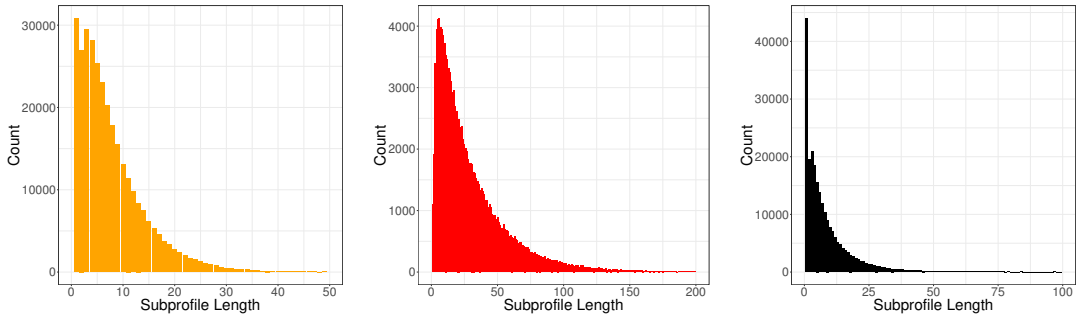
#### 4.4.2.1 Subprofile analysis

For each dataset, we extract subprofiles from each user's liked-item-set and compute descriptive statistics and plot the distribution graphs. We use $\mathrm{NN}\text{-}1$ as subprofile detection algorithm since, from the discussion we gave in Section 4.4 and the results we will give in Section 4.4.2.2, it is the one we select to use in




(a) ML1M (b) LFM (c) LT

Figure 4.1: Number of subprofiles plots.




(a) ML1M (b) LFM (c) LT

Figure 4.2: Length of subprofiles plots.

the following chapters of this thesis. NN-1 has hyperparameters. We set their values using the method already described. Specifically, we use the values in the Table A.2 where MF is the baseline recommender.

Table 4.3 shows the average number of subprofiles per user and the average length of the subprofiles. In more detail, Figure 4.1 contains histograms for the number of subprofiles, i.e. how many users have just one subprofile (equal to the whole liked-item-set), how many have two, how many have three, and so on. Figure 4.2 is a histogram for subprofile lengths, i.e. how many subprofiles contain just one item, how many contain two, and so on. We can see that for the LFM dataset both the number of subprofiles and the lengths of the subprofiles are greater than for the ML1M and LT datasets. This can be partly explained by characteristics of the music domain. The time to listen to a piece of music is typically much less than the time to watch a movie or read a book. Therefore, users can consume more pieces of music and this appears to remain the case even when consumption is aggregated by artists, as is the case in the LastFM dataset. Hence, user profiles are longer in the LFM dataset — see Table 4.3. Longer user profiles tends to imply longer liked-item-sets, which will tend to

give rise to more subprofiles and longer subprofiles.

Table 4.3 also shows the average similarity of subprofiles for each dataset. For a given user $u$, we compute the all-pairs average similarity of $u$'s subprofiles:

$$\text{avg-p-sim}(u) = \frac{\sum_{S \in \mathcal{S}_u^*} \sum_{S' \in \mathcal{S}_u^*, S \neq S'} \text{p-sim}(S, S')}{|\mathcal{S}_u^*|(|\mathcal{S}_u^*| - 1)} \tag{4.13}$$

where $\mathcal{S}_u^*$ is the final set of subprofiles for $u$ and $\text{p-sim}$ measures the similarity between two subprofiles. Subprofiles are just set of items (movies, books, etc.) and so the similarity between two subprofiles that we want here is simply how much they overlap, for which Jaccard similarity seems appropriate: $\text{p-sim}(S, S') = \frac{|S \cap S'|}{|S \cup S'|}$. The value in Table 4.3 is the mean $\text{avg-p-sim}(u)$ for all users $u$.

As the Table shows, LFM subprofiles are more similar to each other than ML1M and LT subprofiles. Intuitively, this means that while a book or a movie covers a few different tastes or interests (subprofiles) of a user, a musician covers more tastes or interests.

We also give an explicit example of the final subprofiles for a user in the ML1M dataset — Table 4.4. The Table shows the user's liked-item-set. Then, for each member of the liked-item-set, the Table shows the corresponding subprofile. Consider, for instance, the movie *Star Wars: Episode IV — A New Hope*. The user likes this movie and it is one of the nearest-neighbours of the movies *A Clockwork Orange*, *Back to the Future*, *Indiana Jones and the Last Crusade* and *The Matrix*, which are also in the liked-item-set. Consider now the movie *The Shining*. The Table shows that its corresponding subprofile contains only the movie itself, which means none of the other members of the liked-item-set has *The Shining* as one of their nearest-neighbours.

The twelve subprofiles shown in Table 4.4 are used as the aspects of the user in SPAD and RSPAD. We regard them as defining the different tastes and interests of this user. Note that the other thirteen members of this user's liked-item-set will also be associated with candidate subprofiles. We did not show these because they are removed using Eq. 4.6: they are subsets of other candidate subprofiles.

Table 4.4: Example subprofiles. These are the final subprofiles of user 5870 in the ML1M dataset.

| **User 5870's liked-item-set** | |
|---|---|
| October Sky, Star Wars: Episode IV — A New Hope, Back to the Future, A Clockwork Orange, Pulp Fiction, Swingers, The Sixth Sense, The Matrix, Dogma, Alive, Being John Malkovich, The Big Lebowski, Full Metal Jacket, Fight Club, This is Spinal Tap, The Shining, Rushmore, Fear and Loathing in Las Vegas, Indiana Jones & the Last Crusade, The Shawshank Redemption, Fargo, Grosse Pointe Blank, Natural Born Killers, Brazil, Die Hard, American Beauty | |
| **Item** | **Corresponding candidate subprofile** |
| Star Wars: Episode IV — A New Hope | Star Wars: Episode IV — A New Hope, A Clockwork Orange, Back to the Future, Indiana Jones & the Last Crusade, The Matrix |
| The Matrix | The Matrix, Star Wars: Episode IV — A New Hope, The Sixth Sense |
| The Shining | The Shining |
| Rushmore | Rushmore, Swingers, The Big Lebowski, Being John Malkovich, Fear and Loathing in Las Vegas, Fight Club |
| Natural Born Killers | Natural Born Killers |
| Brazil | Brazil, A Clockwork Orange |
| Pulp Fiction | Pulp Fiction, Fargo, Swingers, The Big Lebowski, Being John Malkovich, The Sixth Sense, American Beauty, Rushmore, Grosse Pointe Blank, The Shawshank Redemption, Fight Club |
| American Beauty | American Beauty, Fargo, Being John Malkovich, Pulp Fiction, The Sixth Sense, Rushmore, Dogma, The Shawshank Redemption, Fight Club |
| This Is Spinal Tap | This Is Spinal Tap |
| The Shawshank Redemption | The Shawshank Redemption, Fargo, Alive, October Sky, Pulp Fiction, The Sixth Sense, American Beauty |
| Grosse Pointe Blank | Grosse Pointe Blank, Swingers, The Big Lebowski, Dogma |
| Die Hard | Die Hard, Full Metal Jacket, Indiana Jones & the Last Crusade |

#### 4.4.2.2 Results for different subprofile detection methods

In this section, we compare the performance of the eight subprofile detection methods.

Results for the ML1M, LFM and LT datasets using MF as baseline recommender

are shown in Table 4.5. Results for all three datasets using pLSA as baseline recommender are shown in Table 4.6, and results using FMBPR as baseline recommender are shown in Table 4.7. In all three tables, we report precision and $\alpha$-nDCG for different subprofile detection approaches (each of which are used to re-rank the a sorted list generated by one of the MF, pLSA or FMBPR baseline recommenders based on predicted scores). In each block of the tables, results for the baseline recommenders are presented first, and then results for each of the re-ranking algorithms are given. The best result for each metric is highlighted in bold for each block. The value of $\lambda$ that optimizes $\alpha$-nDCG for each baseline and re-ranking strategy is given [WH16]. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$) except those shown in italics.

The tables show that all of the eight re-ranking algorithms achieve both their best precision and $\alpha$-nDCG values using MF as the baseline recommender.

Table 4.5 shows that, when MF is used as the baseline recommendation algorithm, of the eight methods, NN-1 has the highest precision for all but the LT dataset, where NN-2 has the highest precision. Looking at the diversity metric ($\alpha$-nDCG), we see that the results are more mixed. For the ML1M dataset, NN-2 performs the best; for the LFM dataset, it is NN-1; and for the LT, it is DAMIB-COVER$_{\text{cp}}$.

Looking at the results shown in Table 4.6 for the pLSA baseline recommender, both precision and $\alpha$-nDCG results are mixed. For ML1M dataset, DAMIB-COVER has the highest precision and NN-2 has the highest $\alpha$-nDCG; for the LFM dataset NN-1 has both the highest precision and $\alpha$-nDCG; and for the LT dataset DAMIB$_{\text{cp}}$ has both the highest precision and $\alpha$-nDCG.

Table 4.7 shows that, when FMBPR is used as the baseline recommendation algorithm, of the eight methods, NN-1 has the highest precision for all but the LT dataset, where DAMIB has the highest precision. Looking at the diversity metric ($\alpha$-nDCG), we see that the results are more mixed. For the ML1M dataset, NN-2 has the highest $\alpha$-nDCG value; for the LFM dataset, NN-1 has the highest; and for the LT dataset DAMIB-COVER$_{\text{cp}}$ has the highest.

On balance, looking at the Tables 4.5, 4.6 and 4.7, it can be seen that NN-1 is the best-performing method, and NN-2 is the second best-performing method. Of the nine different configurations (three different baseline recommendation algorithms tested on three different datasets), NN-1 has the highest precision

Table 4.5: Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using MF as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| MF | | 0.2916 | 0.3197 | | |
| NN-1 | 0.4 | **0.3005** | 0.3351 | 3.03 | 4.81 |
| NN-2 | 0.5 | 0.2982 | **0.3403** | 2.27 | 6.42 |
| IB+ | 0.6 | 0.2947 | 0.337 | 1.04 | 5.39 |
| DAMIB | 0.6 | 0.2952 | 0.3372 | 1.23 | 5.46 |
| DAMIB-COVER | 0.6 | 0.2961 | 0.3371 | 1.53 | 5.44 |
| $IB+_{cp}$ | 0.4 | 0.2953 | 0.3305 | 1.27 | 3.36 |
| $DAMIB_{cp}$ | 0.5 | *0.2934* | 0.3313 | 0.62 | 3.61 |
| $DAMIB\text{-}COVER_{cp}$ | 0.5 | 0.2981 | 0.337 | 2.21 | 5.39 |
| LFM | | | | | |
| MF | | 0.4654 | 0.4244 | | |
| NN-1 | 0.2 | **0.4742** | **0.4296** | 1.9 | 1.24 |
| NN-2 | 0.2 | 0.4725 | 0.4276 | 1.53 | 0.76 |
| IB+ | 0.3 | 0.4733 | 0.429 | 1.69 | 1.11 |
| DAMIB | 0.2 | 0.4708 | 0.4272 | 1.16 | 0.68 |
| DAMIB-COVER | 0.3 | 0.4739 | 0.4288 | 1.84 | 1.06 |
| $IB+_{cp}$ | 0.2 | *0.4682* | *0.4242* | 0.61 | -0.03 |
| $DAMIB_{cp}$ | 0.2 | 0.47 | *0.4255* | 1.0 | 0.27 |
| $DAMIB\text{-}COVER_{cp}$ | 0.2 | 0.471 | 0.4278 | 1.21 | 0.81 |
| LT | | | | | |
| MF | | 0.1733 | 0.2412 | | |
| NN-1 | 0.4 | 0.1896 | 0.2588 | 9.4 | 7.28 |
| NN-2 | 0.4 | **0.1924** | 0.2623 | 11.05 | 8.72 |
| IB+ | 0.5 | 0.1849 | 0.253 | 6.7 | 4.89 |
| DAMIB | 0.4 | 0.1895 | 0.2607 | 9.38 | 8.07 |
| DAMIB-COVER | 0.4 | 0.19 | 0.2623 | 9.66 | 8.75 |
| $IB+_{cp}$ | 0.4 | 0.1857 | 0.256 | 7.18 | 6.14 |
| $DAMIB_{cp}$ | 0.4 | 0.1892 | 0.2601 | 9.16 | 7.83 |
| $DAMIB\text{-}COVER_{cp}$ | 0.5 | 0.1909 | **0.2625** | 10.16 | 8.82 |

for 5 out of 9 configurations, and the highest $\alpha$-nDCG for 3 out of 9. NN-2 has the highest $\alpha$-nDCG for 3 out of 9 configurations and has the highest precision for one configuration.

The qualitative arguments we gave in Section 4.4 also favoured NN-1 and NN-2: they are simpler, run faster and require that we set fewer hyperparameter values. Hence, in the following chapters of this thesis, we present results for SPAD and RSPAD using NN-1.

Table 4.6: Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using pLSA as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| pLSA | | 0.2639 | 0.2842 | | |
| NN-1 | 1.0 | 0.2803 | 0.3171 | 6.2 | 11.57 |
| NN-2 | 1.0 | 0.2775 | **0.3238** | 5.14 | 13.91 |
| IB+ | 1.0 | 0.2783 | 0.3233 | 5.44 | 13.74 |
| DAMIB | 1.0 | 0.2785 | 0.3234 | 5.51 | 13.8 |
| DAMIB-COVER | 0.6 | **0.2813** | 0.3192 | 6.6 | 12.29 |
| $\text{IB+}_{\text{cp}}$ | 0.7 | 0.2727 | 0.3088 | 3.32 | 8.65 |
| $\text{DAMIB}_{\text{cp}}$ | 1.0 | 0.2712 | 0.3095 | 2.74 | 8.87 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 1.0 | 0.2773 | 0.3183 | 5.06 | 11.99 |
| LFM | | | | | |
| pLSA | | 0.3804 | 0.3426 | | |
| NN-1 | 0.6 | **0.4299** | **0.3878** | 13 | 13.19 |
| NN-2 | 0.5 | 0.421 | 0.3808 | 10.67 | 11.14 |
| IB+ | 0.5 | 0.4199 | 0.3823 | 10.37 | 11.59 |
| DAMIB | 0.5 | 0.4195 | 0.3819 | 10.27 | 11.47 |
| DAMIB-COVER | 0.6 | 0.4196 | 0.3805 | 10.28 | 11.06 |
| $\text{IB+}_{\text{cp}}$ | 0.5 | 0.4059 | 0.3673 | 6.68 | 7.19 |
| $\text{DAMIB}_{\text{cp}}$ | 0.6 | 0.4 | 0.3606 | 5.15 | 5.23 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 0.6 | 0.4182 | 0.3795 | 9.92 | 10.76 |
| LT | | | | | |
| pLSA | 0.0965 | 0.1376 | | | |
| NN-1 | 0.6 | 0.1407 | 0.1937 | 45.8 | 40.75 |
| NN-2 | 0.6 | 0.1419 | 0.1959 | 46.98 | 42.4 |
| IB+ | 0.5 | 0.1393 | 0.1937 | 44.28 | 40.74 |
| DAMIB | 0.5 | 0.1433 | 0.1987 | 48.49 | 44.39 |
| DAMIB-COVER | 0.5 | 0.1428 | 0.1983 | 47.96 | 44.13 |
| $\text{IB+}_{\text{cp}}$ | 0.5 | 0.1397 | 0.1906 | 44.76 | 40.73 |
| $\text{DAMIB}_{\text{cp}}$ | 0.6 | **0.1442** | **0.1996** | 49.41 | 45.09 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 0.6 | 0.1436 | 0.1992 | 48.78 | 44.78 |

#### 4.4.2.3 SPAD vs. DAMIB-COVER

DAMIB-COVER, which is one of the subprofile detection algorithms we have presented in Section 4.3.2.3, is also a recommender in its own right, and it is designed to make recommendations to the different tastes exhibited by a user profile (whether those be the tastes of different users of a shared account or the different tastes of a single user account). In some sense, then, it tries to make a diverse set of recommendations, and so in this subsection, we compare

Table 4.7: Precision and $\alpha$-nDCG for different subprofile detection approaches for ML1M, LFM and LT datasets using FMBPR as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| FMBPR | | 0.2655 | 0.3025 | | |
| NN-1 | 0.4 | **0.2829** | 0.3219 | 6.58 | 6.43 |
| NN-2 | 0.4 | 0.2786 | **0.3233** | 4.94 | 6.87 |
| IB+ | 0.4 | 0.2765 | 0.321 | 4.16 | 6.13 |
| DAMIB | 0.4 | 0.2766 | 0.3205 | 4.18 | 5.95 |
| DAMIB-COVER | 0.4 | 0.2772 | 0.3203 | 4.41 | 5.9 |
| $\text{IB+}_{\text{cp}}$ | 0.4 | 0.2718 | 0.3141 | 2.37 | 3.83 |
| $\text{DAMIB}_{\text{cp}}$ | 0.4 | 0.2718 | 0.3136 | 2.37 | 3.66 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 0.4 | 0.2769 | 0.3185 | 4.31 | 5.31 |
| LFM | | | | | |
| FMBPR | | 0.3737 | 0.3409 | | |
| NN-1 | 0.4 | **0.4231** | **0.3864** | 13.21 | 13.35 |
| NN-2 | 0.5 | 0.4178 | 0.3792 | 11.78 | 11.25 |
| IB+ | 0.5 | 0.4126 | 0.3793 | 10.41 | 11.26 |
| DAMIB | 0.5 | 0.4129 | 0.379 | 10.48 | 11.18 |
| DAMIB-COVER | 0.6 | 0.4149 | 0.3787 | 11.01 | 11.09 |
| $\text{IB+}_{\text{cp}}$ | 0.5 | 0.3986 | 0.366 | 6.65 | 7.38 |
| $\text{DAMIB}_{\text{cp}}$ | 0.6 | 0.3893 | 0.3553 | 4.17 | 4.22 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 0.6 | 0.4128 | 0.378 | 10.44 | 10.88 |
| LT | | | | | |
| FMBPR | | 0.0829 | 0.112 | | |
| NN-1 | 0.5 | 0.1275 | 0.1696 | 53.78 | 51.38 |
| NN-2 | 0.5 | 0.1288 | 0.1732 | 55.36 | 54.63 |
| IB+ | 0.6 | 0.1214 | 0.1656 | 46.41 | 47.78 |
| DAMIB | 0.6 | **0.1296** | 0.174 | 56.36 | 55.3 |
| DAMIB-COVER | 0.5 | 0.1292 | 0.1748 | 55.8 | 56.01 |
| $\text{IB+}_{\text{cp}}$ | 0.5 | 0.1243 | 0.1671 | 49.98 | 49.1 |
| $\text{DAMIB}_{\text{cp}}$ | 0.5 | 0.1294 | 0.1747 | 56.08 | 55.94 |
| $\text{DAMIB-COVER}_{\text{cp}}$ | 0.5 | 0.1286 | **0.1753** | 55.15 | 56.46 |

DAMIB-COVER as a recommender with SPAD.

The results of the comparison are given in Table 4.8. The best result for each metric is highlighted in bold for each block. The value of $\lambda$ that optimizes $\alpha$-nDCG for SPAD is given. Results that are statistically significant with respect to the +ve-only IB baseline (Wilcoxon signed rank with $p < 0.05$) are marked with $^{a}$. SPAD results are also marked with $^{b}$ if they are statistically significant with respect to DAMIB-COVER. The baseline here is a positive-only item-based

Table 4.8: DAMIB-COVER & SPAD results.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| +ve-only IB | | 0.2205 | 0.2653 | | |
| DAMIB-COVER | | $0.215^{a}$ | $0.255^{a}$ | -2.52 | -3.88 |
| SPAD | 0.9 | $\mathbf{0.2219}^{ab}$ | $\mathbf{0.2676}^{ab}$ | 0.64 | 0.87 |
| LFM | | | | | |
| +ve-only IB | | 0.2961 | 0.2865 | | |
| DAMIB-COVER | | $0.2877^{a}$ | $0.2705^{a}$ | -2.83 | -5.93 |
| SPAD | 0.3 | $\mathbf{0.3113}^{ab}$ | $\mathbf{0.299}^{ab}$ | 5.14 | 4.35 |
| LT | | | | | |
| +ve-only IB | | **0.1609** | **0.2181** | | |
| DAMIB-COVER | | $0.146^{a}$ | $0.21^{a}$ | -9.29 | -3.9 |
| SPAD | 0.1 | $\mathit{0.1605}^{b}$ | $0.2174^{ab}$ | -0.25 | -0.29 |

recommender, which is the input to DAMIB-COVER; its results are re-ranked by SPAD, to give a fair comparison. We see that SPAD outperforms DAMIB-COVER for both accuracy and diversity on all datasets. This justifies the additional complexity of SPAD's aspect-aware approach over DAMIB-COVER.

## 4.5 Conclusions

In this chapter, we have presented Subprofile-Aware Diversification (SPAD) and its variant Relevance-based SPAD (RSPAD). We have presented eight different subprofile detection methods for SPAD. We have presented qualitative arguments and experimental results which favour one of the methods, NN-1. Hence, in the rest of this thesis, when we refer to SPAD and RSPAD, we will be using NN-1 as the subprofile detection method.

In the next chapter, we give an extensive comparison of SPAD and RSPAD with existing intent-aware diversification approaches.

# Chapter 5

# An Evaluation of Subprofile-Aware Diversification

In the previous chapter, we presented Subprofile-Aware Diversification (SPAD) and its variant, Relevance-based SPAD (RSPAD). We presented eight different subprofile detection methods for SPAD. We presented qualitative arguments and experimental results that favored one of the methods (designated NN-1).

In this chapter, using NN-1 as the subprofile detection method, we compare SPAD and RSPAD with other diversification techniques. First, we give the experimental results. Next, we adapt diversity metrics that use item features to measure diversity, such that they use user subprofiles instead of item features. Using these new metrics alongside the existing ones gives a more balanced view of the performance of the recommender algorithms. Then, we show results showing relevance/diversity trade-off.

## 5.1 Comparison of SPAD and Existing Intent-Aware Diversification Methods

In this section, we report our empirical investigation of SPAD, RSPAD and other approaches to diversification.

We use the same datasets that we have used in the previous chapter: the MovieLens 1M (ML1M) dataset, the LastFM (LFM) dataset and the LibraryThing (LT) dataset. For the characteristic of the datasets, see Chapter 3.

We compare SPAD and RSPAD with the other diversification techniques available in the RankSys library[1]: MMR [CG98], xQuAD [VCV11], RxQuAD [VCV12] and c-pLSA [WH16]. We also compare our methods with SxQuAD and SRxQuAD [VC13]. Since they are not available in the RankSys library, we implemented them ourselves.

All of these approaches to diversification use greedy re-ranking, therefore they need a baseline recommender, whose recommendation sets are re-ranked. We use the same baseline recommenders that we have used in the previous chapter (again using their RankSys implementations): probabilistic latent semantic analysis (pLSA) [Hof04], a fast alternative least-squares matrix factorization recommender (MF) [PZT10], and a factorization machine that uses Bayesian pairwise loss for ranking (FMBPR) [Bay15].

Three baselines paired with nine re-ranking approaches (the eight above but also none at all) gives 27 systems to compare on each dataset. However, we were unable to obtain results for c-pLSA on the LFM and LT datasets because the implementation is based on the maximum possible item features (71833 user-generated tags in LFM and 4800 in LT), whereas the other re-ranking approaches that use item features only depend on the number of distinct features that describe the items in $I_u$.

We divide this section into two: first we give results that compare SPAD and RSPAD with other re-ranking algorithms; then we show the precision and diversity (measured by $\alpha$-nDCG) results for different values of $\lambda$.

### 5.1.1 Results for different algorithms

For each of the three datasets, we show precision and $\alpha$-nDCG results in one table, and the other diversity metrics (ERR-IA, S-recall, EILD and ILD) in another table. In each block of Tables 5.1 and 5.2 (and also Tables 5.3, 5.4, 5.5 and 5.6), results for the baseline recommendation algorithm are presented first, and then results for each of the re-ranking algorithms. The best result for each metric is highlighted in bold for each block. For each algorithm, we report the results using the value of $\lambda$ that gives the highest $\alpha$-nDCG on the validation set [WH16]. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$), except those shown in italics.

[1] https://github.com/RankSys

Table 5.1: Precision and $\alpha$-nDCG results for ML1M dataset.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| MF | | 0.2916 | 0.3197 | | |
| MMR | 0.2 | 0.2906 | 0.3243 | -0.34 | 1.43 |
| xQuAD | 0.5 | 0.2739 | 0.3668 | -6.08 | 14.72 |
| RxQuAD | 0.7 | 0.2629 | 0.3586 | -9.85 | 12.15 |
| SxQuAD | 0.6 | 0.2743 | **0.3687** | -5.93 | 15.32 |
| SRxQuAD | 0.6 | 0.2715 | 0.3658 | -6.9 | 14.39 |
| c-pLSA | 0.3 | 0.2978 | 0.3292 | 2.1 | 2.96 |
| SPAD | 0.4 | **0.3005** | 0.3351 | 3.03 | 4.81 |
| RSPAD | 0.7 | 0.2975 | 0.3356 | 2.02 | 4.97 |
| pLSA | | 0.2639 | 0.2842 | | |
| MMR | 0.3 | *0.2635* | 0.2913 | -0.17 | 2.47 |
| xQuAD | 0.7 | 0.2456 | **0.3428** | -6.93 | 20.61 |
| RxQuAD | 1.0 | 0.2452 | 0.3341 | -7.1 | 17.53 |
| SxQuAD | 0.5 | 0.2686 | 0.3396 | 1.76 | 19.49 |
| SRxQuAD | 0.5 | 0.2676 | 0.3348 | 1.38 | 17.78 |
| c-pLSA | 0.5 | 0.2763 | 0.3075 | 3.88 | 6.12 |
| SPAD | 1.0 | 0.2803 | 0.3171 | 6.2 | 11.57 |
| RSPAD | 1.0 | **0.2824** | 0.3177 | 7 | 11.79 |
| FMBPR | | 0.2655 | 0.3025 | | |
| MMR | 0.2 | 0.2649 | 0.3068 | -0.22 | 1.42 |
| xQuAD | 0.4 | 0.2534 | **0.3376** | -4.56 | 11.61 |
| RxQuAD | 0.5 | 0.2429 | 0.3272 | -8.48 | 8.16 |
| SxQuAD | 0.5 | 0.2313 | 0.3197 | -12.85 | 5.68 |
| SRxQuAD | 0.5 | 0.2318 | 0.318 | -12.69 | 5.13 |
| c-pLSA | 0.3 | 0.2754 | 0.3157 | 3.75 | 4.38 |
| SPAD | 0.4 | **0.2829** | 0.3219 | 6.58 | 6.43 |
| RSPAD | 0.4 | 0.2818 | 0.3221 | 6.14 | 6.48 |

The precision and $\alpha$-nDCG results for the experiments on the ML1M dataset are shown in Table 5.1; and the results for the other diversity metrics (i.e. ERR-IA, S-recall, EILD and ILD) are given in Table 5.2. Consider precision and $\alpha$-nDCG first. Either SPAD or RSPAD achieve the highest precision and c-pLSA achieves either the second or the third best precision scores. For $\alpha$-nDCG, xQuAD is the best re-ranking method for all but the MF baseline, where SxQuAD is the best method. In almost all settings, all variations of xQuAD perform better than SPAD and RSPAD. SPAD and RSPAD are at a disadvantage since they make no use of the explicit item features. Indeed, $\alpha$-nDCG is very similar to what is used for re-ranking xQuAD and its variations. Even so, SPAD and RSPAD always have higher diversity than c-pLSA and MMR, and they have higher diversity

Table 5.2: Diversity metrics except $\alpha$-nDCG for ML1M dataset.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| MF | | 0.2102 | 0.4783 | 0.2123 | 0.7176 | | | | |
| MMR | 0.2 | 0.2115 | 0.5116 | 0.2217 | 0.7566 | 0.61 | 6.97 | 4.45 | 5.44 |
| xQuAD | 0.5 | 0.2232 | **0.6271** | 0.2033 | 0.7592 | 6.17 | 31.13 | -4.24 | 5.80 |
| RxQuAD | 0.7 | 0.2369 | 0.5834 | 0.1907 | 0.7405 | 12.67 | 21.98 | -10.02 | 3.19 |
| SxQuAD | 0.6 | 0.2297 | 0.5846 | 0.2016 | 0.7353 | 9.25 | 22.23 | -5.06 | 2.46 |
| SRxQuAD | 0.6 | **0.24** | 0.6199 | 0.2094 | **0.7768** | 14.19 | 29.61 | -1.38 | 8.25 |
| c-pLSA | 0.3 | 0.2174 | *0.478* | 0.2171 | 0.7083 | 3.44 | -0.06 | 2.27 | -1.29 |
| SPAD | 0.4 | 0.2197 | 0.4957 | **0.2219** | 0.7244 | 4.52 | 3.63 | 4.5 | 0.95 |
| RSPAD | 0.7 | 0.2202 | 0.5071 | 0.2211 | 0.7309 | 4.74 | 6.03 | 4.13 | 1.86 |
| pLSA | | 0.1831 | 0.4664 | 0.1864 | 0.7023 | | | | |
| MMR | 0.3 | 0.1851 | 0.5141 | 0.1978 | 0.7561 | 1.1 | 10.22 | 6.15 | 7.66 |
| xQuAD | 0.7 | 0.2041 | **0.6589** | 0.1815 | **0.7706** | 11.46 | 41.27 | -2.62 | 9.73 |
| RxQuAD | 1.0 | 0.217 | 0.5916 | 0.1769 | 0.7458 | 18.53 | 26.85 | -5.06 | 6.2 |
| SxQuAD | 0.5 | 0.2124 | 0.5959 | 0.1987 | 0.7529 | 16.01 | 27.76 | 6.64 | 7.21 |
| SRxQuAD | 0.5 | **0.2196** | 0.5868 | 0.2023 | 0.7666 | 19.93 | 25.8 | 8.54 | 9.16 |
| c-pLSA | 0.5 | 0.2011 | 0.4754 | 0.1976 | 0.6966 | 9.87 | 1.93 | 6.01 | -0.81 |
| SPAD | 1.0 | 0.2037 | 0.5137 | 0.2055 | 0.7282 | 11.26 | 10.15 | 10.29 | 3.69 |
| RSPAD | 1.0 | 0.2048 | 0.5069 | **0.2069** | 0.7274 | 11.85 | 8.69 | 11.02 | 3.58 |
| FMBPR | | 0.1982 | 0.5119 | 0.1854 | 0.7166 | | | | |
| MMR | 0.2 | 0.1991 | 0.5563 | 0.1958 | **0.765** | 0.43 | 8.67 | 5.6 | 6.76 |
| xQuAD | 0.4 | 0.206 | **0.6215** | 0.1797 | 0.7404 | 3.92 | 21.41 | -3.05 | 3.36 |
| RxQuAD | 0.5 | **0.2146** | 0.5899 | 0.1701 | 0.7395 | 8.27 | 15.24 | -8.25 | 3.19 |
| SxQuAD | 0.5 | *0.1999* | 0.6074 | 0.1577 | 0.7201 | 0.84 | 18.65 | -14.93 | 0.49 |
| SRxQuAD | 0.5 | 0.2059 | 0.6151 | 0.1665 | 0.7527 | 3.87 | 20.15 | -10.18 | 5.04 |
| c-pLSA | 0.3 | 0.2094 | *0.5087* | 0.1951 | 0.7086 | 5.66 | -0.63 | 5.24 | -1.11 |
| SPAD | 0.4 | 0.2107 | 0.5208 | **0.2055** | 0.7337 | 6.29 | 1.73 | 10.84 | 2.39 |
| RSPAD | 0.4 | 0.2109 | 0.5222 | 0.2045 | 0.7359 | 6.4 | 2.01 | 10.32 | 2.69 |

than SxQuAD and SRxQuAD where FMBPR is the baseline. Furthermore, when we look at precision and $\alpha$-nDCG together, we see that xQuAD and its variants achieve their diversity at the expense of the largest decreases in precision.

Next, consider the other diversity metrics shown in Table 5.2. SPAD, RSPAD and MMR improve all the diversity metrics. But MMR does this at the cost of lowering precision, whereas SPAD and RSPAD increase precision in comparison with the baselines. It is surprising in the case of SPAD and RSPAD, since we have argued that they are at a disadvantage using these diversity metrics. xQuAD and its variations increase the diversity with one exception only: for EILD, xQuAD and RxQuAD always decrease the value of the baseline recommenders. SxQuAD and SRxQuAD also decrease the EILD value of the MF and FMBPR baselines. Considering Tables 5.1 and 5.2 together, for ML1M dataset, only SPAD and RSPAD always improve precision and all diversity metrics over the baselines.

The results for the LFM dataset are in Tables 5.3 and 5.4. Recall that, c-pLSA is missing from these results because we were unable to run it to completion on a dataset with so many explicit features (tags). Consider precision and $\alpha$-nDCG first, which are shown in Table 5.3. Here, either SPAD or RSPAD achieve the

Table 5.3: Precision and $\alpha$-nDCG results for LFM dataset.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| MF | | 0.4654 | 0.4244 | | |
| MMR | 0.3 | 0.4545 | 0.4312 | -2.35 | 1.62 |
| xQuAD | 0.3 | 0.4701 | **0.4354** | 1.01 | 2.61 |
| RxQuAD | 0.3 | *0.4654* | 0.4253 | 0.01 | 0.22 |
| SxQuAD | 0.2 | 0.4694 | 0.4251 | 0.87 | 0.17 |
| SRxQuAD | 0.3 | *0.4665* | 0.427 | 0.23 | 0.63 |
| SPAD | 0.2 | 0.4742 | 0.4296 | 1.9 | 1.24 |
| RSPAD | 0.4 | **0.4774** | 0.4302 | 2.57 | 1.38 |
| pLSA | | 0.3804 | 0.3426 | | |
| MMR | 0.3 | 0.3773 | 0.3499 | -0.84 | 2.13 |
| xQuAD | 0.5 | 0.41 | 0.3847 | 7.78 | 12.28 |
| RxQuAD | 0.8 | 0.3993 | 0.3631 | 4.96 | 5.96 |
| SxQuAD | 0.5 | 0.4053 | 0.3649 | 6.54 | 6.51 |
| SRxQuAD | 0.5 | 0.3968 | 0.3659 | 4.29 | 6.79 |
| SPAD | 0.6 | **0.4299** | **0.3878** | 13 | 13.19 |
| RSPAD | 0.6 | 0.4295 | 0.3866 | 12.89 | 12.83 |
| FMBPR | | 0.3737 | 0.3409 | | |
| MMR | 0.1 | *0.3727* | 0.3432 | -0.27 | 0.67 |
| xQuAD | 0.4 | 0.3972 | 0.3758 | 6.28 | 10.23 |
| RxQuAD | 0.5 | 0.3856 | 0.3529 | 3.17 | 3.52 |
| SxQuAD | 0.5 | 0.3976 | 0.3547 | 6.39 | 4.06 |
| SRxQuAD | 0.5 | 0.3831 | 0.3562 | 2.52 | 4.49 |
| SPAD | 0.4 | 0.4231 | **0.3864** | 13.21 | 13.35 |
| RSPAD | 0.5 | **0.4247** | 0.3854 | 13.65 | 13.05 |

highest and second highest precision scores for all baseline recommendation algorithms. Again, despite making no use of explicit features, SPAD and RSPAD always increase diversity measured by $\alpha$-nDCG. Besides, SPAD achieves the highest $\alpha$-nDCG with one exception only, where MF is the baseline and xQuAD achieves the highest $\alpha$-nDCG. Interestingly, all the re-ranking methods, except MMR, increase precision (albeit only slightly in the case of RxQuAD where MF is the baseline) as well as increasing $\alpha$-nDCG. None increase precision as much as SPAD and RSPAD, which arguably achieve the best balance between increased precision and increased $\alpha$-nDCG.

Next, consider the other diversity metrics, which are in Table 5.4. The diversity with respect to ERR-IA is increased by all of the re-ranking approaches compared with the baselines. With respect to S-recall, only MMR and xQuAD outperforms all of the baselines. For EILD, SPAD achieves the highest EILD with

Table 5.4: Diversity metrics except $\alpha$-nDCG for LFM dataset.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| MF | | 0.2012 | 6.39E-4 | 0.3671 | 0.7638 | | | | |
| MMR | 0.3 | *0.2013* | **7.06E-4** | **0.378** | **0.8094** | 0.06 | 10.37 | 2.99 | 5.97 |
| xQuAD | 0.3 | 0.2067 | 6.61E-4 | 0.3779 | 0.778 | 2.74 | 3.4 | 2.94 | 1.87 |
| RxQuAD | 0.3 | 0.21 | 6.34E-4 | *0.3661* | 0.7652 | 4.38 | -0.89 | -0.25 | 0.19 |
| SxQuAD | 0.2 | 0.2067 | 6.26E-4 | *0.3666* | 0.7552 | 2.75 | -2.05 | -0.12 | -1.12 |
| SRxQuAD | 0.3 | **0.211** | 6.37E-4 | 0.3689 | 0.7682 | 4.91 | -0.38 | 0.49 | 0.58 |
| SPAD | 0.2 | *0.2036* | 6.36E-4 | 0.3734 | 0.762 | 1.22 | -0.53 | 1.87 | -0.23 |
| RSPAD | 0.4 | 0.2041 | 6.35E-4 | 0.3743 | *0.7624* | 1.48 | -0.71 | 1.97 | -0.18 |
| pLSA | | 0.1629 | 6.14E-4 | 0.2863 | 0.7424 | | | | |
| MMR | 0.3 | 0.1642 | 6.68E-4 | 0.298 | 0.7828 | 0.82 | 8.79 | 4.09 | 5.44 |
| xQuAD | 0.5 | 0.1829 | **6.85E-4** | 0.3296 | **0.7918** | 12.33 | 11.47 | 15.14 | 6.65 |
| RxQuAD | 0.8 | 0.1848 | 6.21E-4 | 0.3049 | 0.7606 | 13.46 | 1.1 | 6.51 | 2.45 |
| SxQuAD | 0.5 | 0.1864 | 6.06E-4 | 0.3065 | 0.7406 | 14.46 | -1.3 | 7.06 | -0.24 |
| SRxQuAD | 0.5 | **0.1872** | 6.32E-4 | 0.3087 | 0.7682 | 14.96 | 2.87 | 7.82 | 3.47 |
| SPAD | 0.6 | 0.184 | 6.36E-4 | **0.3328** | 0.7601 | 12.95 | 3.46 | 16.26 | 2.38 |
| RSPAD | 0.6 | 0.1849 | 6.32E-4 | 0.3307 | 0.7586 | 13.51 | 2.94 | 15.54 | 2.19 |
| FMBPR | | 0.1614 | 6.48E-4 | 0.2868 | 0.7724 | | | | |
| MMR | 0.1 | *0.1615* | 6.66E-4 | 0.2906 | 0.7854 | 0.04 | 2.76 | 1.32 | 1.69 |
| xQuAD | 0.4 | 0.1742 | **6.95E-4** | 0.3216 | **0.8016** | 7.94 | 7.18 | 12.15 | 3.78 |
| RxQuAD | 0.5 | 0.1792 | 6.25E-4 | 0.2945 | 0.7694 | 11.04 | -3.5 | 2.7 | -0.39 |
| SxQuAD | 0.5 | 0.1799 | 6.02E-4 | 0.2965 | 0.7409 | 11.45 | -7.19 | 3.39 | -3.82 |
| SRxQuAD | 0.5 | 0.1822 | 6.35E-4 | 0.2998 | 0.7796 | 12.9 | -2.07 | 4.56 | 0.93 |
| SPAD | 0.4 | 0.1835 | 6.51E-4 | **0.3314** | 0.7736 | 13.66 | 0.42 | 15.56 | 0.15 |
| RSPAD | 0.5 | **0.1847** | 6.44E-4 | 0.3307 | 0.7695 | 14.42 | -0.61 | 15.31 | -0.38 |

one exception only. The exception is the MF baseline, where MMR achieves the highest EILD. With respect to ILD, MMR, xQuAD and SRxQuAD always outperform all baselines; but other re-ranking approaches show a decrease sometimes.

The results for the LT dataset are in Tables 5.5 and 5.6. c-pLSA is missing from these results for the same reason we explained above for the LFM dataset. First, consider the results shown in Table 5.5. They are very similar to the ones for the LFM dataset. SPAD and RSPAD achieve the highest precision and SPAD achieves the highest $\alpha$-nDCG for all baselines but MF, where xQuAD achieves highest $\alpha$-nDCG. SPAD and RSPAD both increase precision along with $\alpha$-nDCG. All of the re-ranking approaches, except MMR, always increase precision along with $\alpha$-nDCG. Arguably, SPAD, RSPAD give the best balance between increased precision and increased $\alpha$-nDCG.

Finally, the results for the other diversity metrics are presented in Table 5.6. With a few exceptions, all the re-ranking approaches increase all the diversity metrics. The exceptions are where MMR is the re-ranking method for the MF baseline for the ERR-IA metric, and where SxQuAD is the re-ranking method for the MF and FMBPR baselines for ILD and S-recall. xQuAD achieves the highest S-recall and ILD. SPAD achieves the highest EILD with one exception only, for the MF baseline, where xQuAD achieves the highest EILD.

Table 5.5: Precision and $\alpha$-nDCG results LT dataset.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| MF | | 0.1733 | 0.2412 | | |
| MMR | 0.1 | 0.1724 | *0.2415* | -0.53 | 0.1 |
| xQuAD | 0.5 | 0.1866 | **0.264** | 7.7 | 9.44 |
| RxQuAD | 0.7 | 0.1801 | 0.2503 | 3.91 | 3.76 |
| SxQuAD | 0.6 | 0.185 | 0.2521 | 6.75 | 4.5 |
| SRxQuAD | 0.6 | 0.1819 | 0.2537 | 4.97 | 5.15 |
| SPAD | 0.4 | 0.1896 | 0.2588 | 9.4 | 7.28 |
| RSPAD | 0.4 | **0.1899** | 0.2576 | 9.59 | 6.77 |
| pLSA | | 0.0965 | 0.1376 | | |
| MMR | 0.2 | 0.0969 | 0.1391 | 0.33 | 1.05 |
| xQuAD | 0.8 | 0.1233 | 0.1816 | 27.76 | 31.94 |
| RxQuAD | 1.0 | 0.1190 | 0.1695 | 23.27 | 23.16 |
| SxQuAD | 0.6 | 0.1368 | 0.1907 | 41.73 | 38.62 |
| SRxQuAD | 0.6 | 0.1352 | 0.1914 | 40.09 | 39.08 |
| SPAD | 0.6 | 0.1407 | **0.1937** | 45.8 | 40.75 |
| RSPAD | 0.7 | **0.1416** | 0.1934 | 46.7 | 40.55 |
| FMBPR | | 0.0829 | 0.112 | | |
| MMR | 0.1 | 0.0833 | 0.1129 | 0.44 | 0.8 |
| xQuAD | 0.5 | 0.1161 | 0.1677 | 40.04 | 49.71 |
| RxQuAD | 0.9 | 0.1081 | 0.1503 | 30.4 | 34.12 |
| SxQuAD | 0.4 | 0.0855 | 0.1158 | 3.12 | 3.35 |
| SRxQuAD | 0.4 | 0.0849 | 0.1163 | 2.35 | 3.79 |
| SPAD | 0.5 | **0.1275** | **0.1696** | 53.78 | 51.38 |
| RSPAD | 0.6 | 0.1255 | 0.1662 | 51.38 | 48.33 |

### 5.1.2 Results for different values of $\lambda$

Here we look at the effect of parameter $\lambda$, which controls the balance between relevance and diversity. The results we have shown so far use whichever values for $\lambda$ give highest $\alpha$-nDCG. Here, instead, we plot precision and $\alpha$-nDCG on the test set for different values of $\lambda$, for all datasets and all three baselines.

For the ML1M dataset (Figure 5.1 for precision and Figure 5.2 for $\alpha$-nDCG), SPAD and RSPAD achieve higher precision than all the other re-ranking approaches and for all values of $\lambda$. Indeed, SPAD and RSPAD re-ranking of the pLSA baseline always has higher precision than the baseline itself. SPAD and RSPAD drop below the baseline in the case of the MF and FMBPR baselines, but they do this only in the case of high values for $\lambda$. SPAD and RSPAD applied to the pLSA and MF recommendation lists achieve higher $\alpha$-nDCG than the baselines

Table 5.6: Diversity metrics except $\alpha$-nDCG for LT dataset.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| MF | | 0.1143 | 0.0097 | 0.1137 | 0.7577 | | | | |
| MMR | 0.1 | *0.1143* | 0.0099 | 0.115 | 0.7687 | - | 2.44 | 1.11 | 1.46 |
| xQuAD | 0.5 | **0.1213** | **0.0104** | **0.1314** | **0.7884** | 6.19 | 7.17 | 15.53 | 4.05 |
| RxQuAD | 0.7 | 0.1197 | 0.0097 | 0.1209 | 0.7721 | 4.82 | 0.51 | 6.29 | 1.9 |
| SxQuAD | 0.6 | 0.1195 | 0.0092 | 0.1223 | 0.7441 | 4.58 | -4.9 | 7.52 | -1.79 |
| SRxQuAD | 0.6 | 0.1212 | 0.0098 | 0.1257 | 0.7802 | 6.1 | 1.3 | 10.58 | 2.97 |
| SPAD | 0.4 | 0.1183 | 0.0099 | 0.1302 | 0.7726 | 3.57 | 2.18 | 14.48 | 1.97 |
| RSPAD | 0.4 | 0.1178 | 0.0097 | 0.128 | 0.7655 | 3.07 | 0.22 | 12.56 | 1.03 |
| pLSA | | 0.0672 | 0.0085 | 0.0478 | 0.698 | | | | |
| MMR | 0.2 | 0.0679 | 0.0089 | 0.0492 | 0.7186 | 0.99 | 4.57 | 2.84 | 2.96 |
| xQuAD | 0.8 | 0.088 | **0.0106** | 0.074 | **0.7955** | 30.95 | 25.11 | 54.72 | 13.96 |
| RxQuAD | 1.0 | 0.0827 | 0.0097 | 0.0662 | 0.7662 | 22.96 | 13.68 | 38.38 | 9.77 |
| SxQuAD | 0.6 | 0.0933 | 0.0093 | 0.0808 | 0.7407 | 38.86 | 8.97 | 68.88 | 6.12 |
| SRxQuAD | 0.6 | **0.0943** | 0.0094 | 0.0811 | 0.7528 | 40.22 | 10.98 | 69.52 | 7.85 |
| SPAD | 0.6 | 0.0912 | 0.0094 | **0.0851** | 0.7494 | 35.74 | 10.81 | 77.94 | 7.36 |
| RSPAD | 0.7 | 0.0911 | 0.0092 | 0.0838 | 0.741 | 35.52 | 8.27 | 75.43 | 6.17 |
| FMBPR | | 0.0546 | 0.009 | 0.0394 | 0.7179 | | | | |
| MMR | 0.1 | 0.0549 | 0.0095 | 0.0401 | 0.7399 | 0.46 | 4.6 | 3.03 | 3.06 |
| xQuAD | 0.5 | 0.0801 | **0.0107** | 0.0683 | **0.7938** | 46.72 | 18.52 | 73.55 | 10.57 |
| RxQuAD | 0.9 | 0.0756 | *0.0091* | 0.0553 | 0.7477 | 38.39 | 0.32 | 40.51 | 4.15 |
| SxQuAD | 0.4 | 0.0576 | 0.0088 | 0.0413 | 0.7069 | 5.44 | -2.47 | 4.89 | -1.53 |
| SRxQuAD | 0.4 | 0.0578 | 0.0092 | 0.0423 | 0.7323 | 5.86 | 1.86 | 7.37 | 2.01 |
| SPAD | 0.5 | **0.0803** | 0.0094 | **0.0742** | 0.7401 | 46.98 | 3.98 | 88.6 | 3.1 |
| RSPAD | 0.6 | 0.0792 | 0.0092 | 0.0707 | 0.7344 | 44.93 | 2.02 | 79.58 | 2.3 |

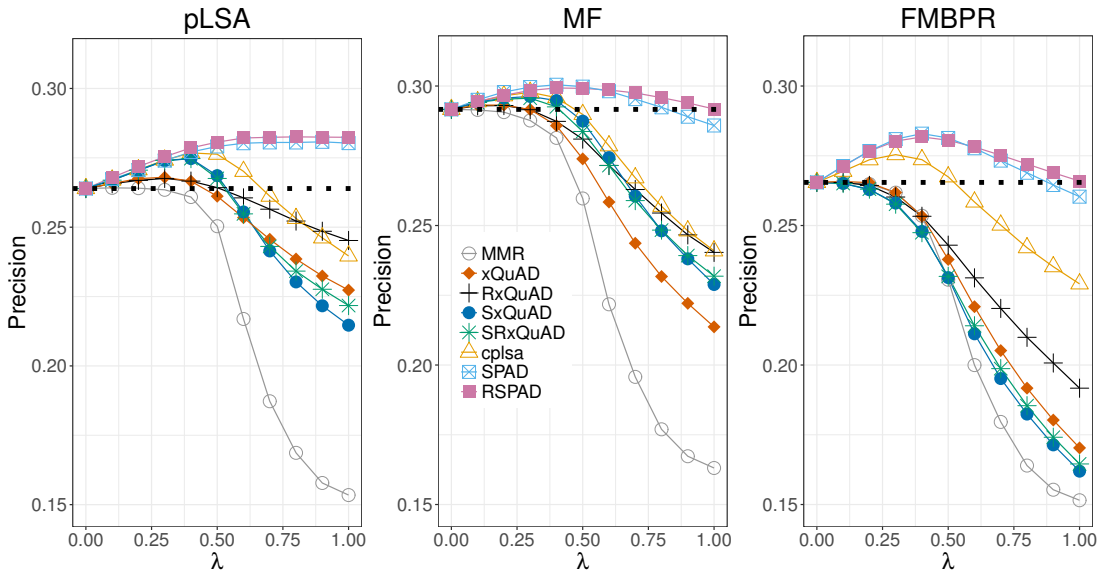



Figure 5.1: ML1M dataset. Precision@10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

for all values of $\lambda$. When the baseline is FMBPR, SPAD and RSPAD become worse than the baseline only for high values of $\lambda$. xQuAD and its variations have higher $\alpha$-nDCG than SPAD and RSPAD for many values of $\lambda$ but they soon suffer from decreases in precision. c-pLSA and MMR, on the other hand, suffer from decreases in $\alpha$-nDCG for values of $\lambda$ of about 0.5 and higher.

The results for LFM can be found in Figures 5.3 and 5.4. Again, for precision,

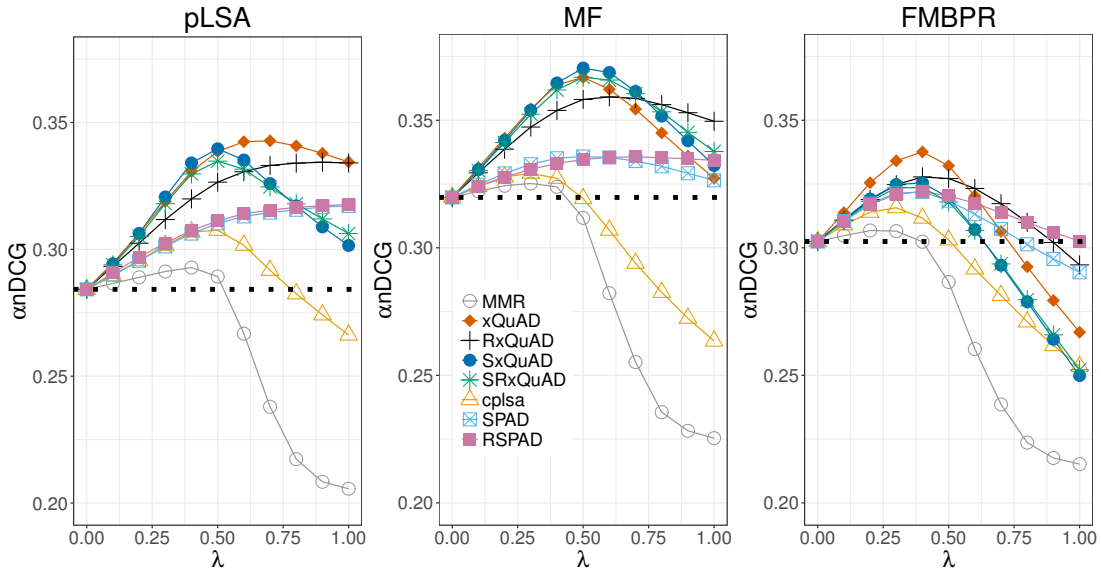



Figure 5.2: ML1M dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

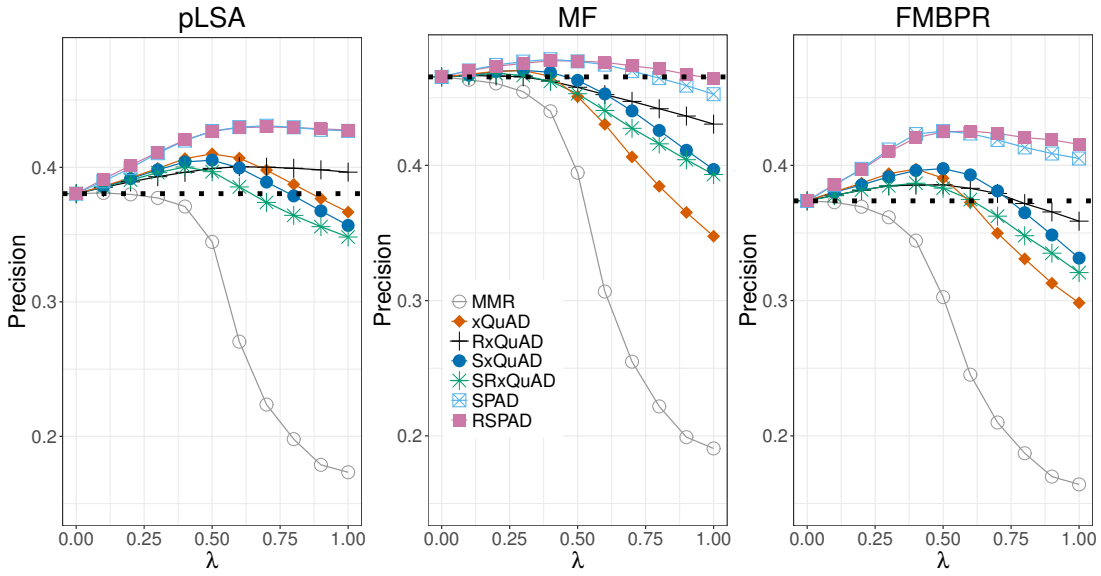



Figure 5.3: LFM dataset. Precision @10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

for all values of $\lambda$, SPAD and RSPAD perform better than all other re-ranking approaches, and they only become worse than the baseline for high values of $\lambda$ and only in the case of the MF baseline. MMR always decreases the precision. xQuAD and its variations increase the precision for smaller values of $\lambda$, but then, with one exception, suffer from decreases. The exception is RxQuAD: for all values of $\lambda$ it has higher precision than the baseline in the case where pLSA is the baseline. For $\alpha$-nDCG, for all values of $\lambda$, SPAD and RSPAD outperform all other re-ranking approaches where pLSA and FMBPR are the baselines. Only

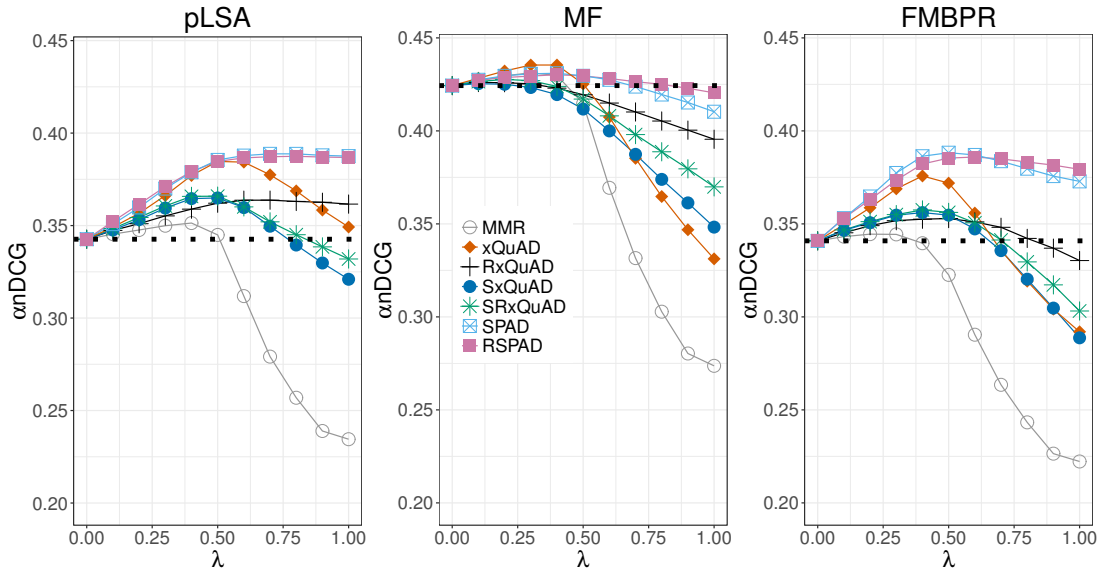



Figure 5.4: LFM dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-nDCG for each baseline is shown by the dotted line.

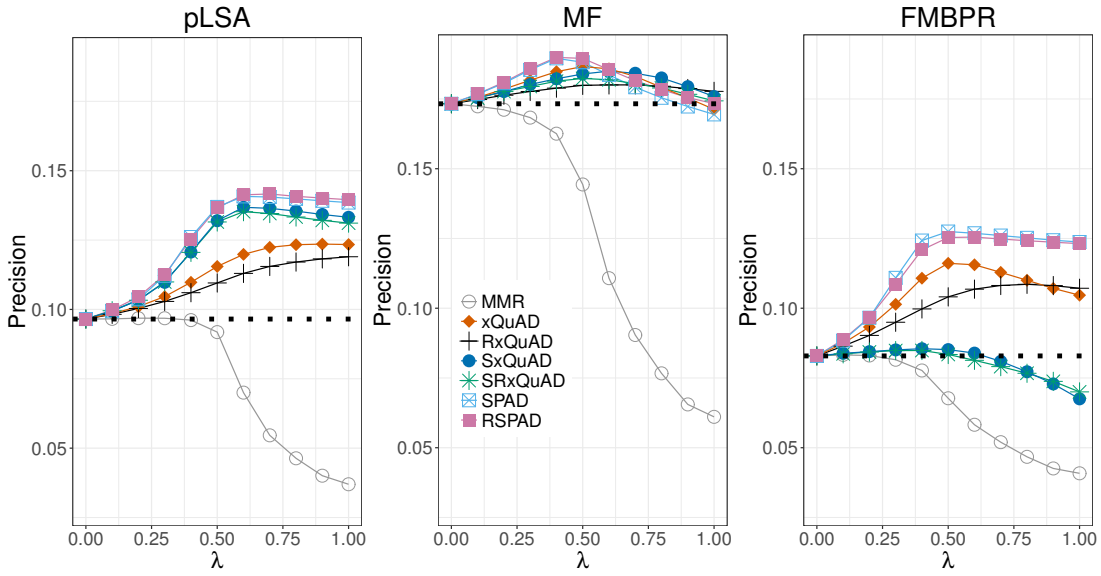



Figure 5.5: LT dataset. Precision @10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

xQuAD and MMR achieve higher $\alpha$-nDCG values than SPAD and RSPAD for some small values of $\lambda$, where MF is the baseline. For the LastFM dataset, SPAD and RSPAD gives the best balance between increased precision and $\alpha$-nDCG for almost all values of $\lambda$.

LT results are in Figures 5.5 and 5.6. Precision results for SPAD and RSPAD are similar to those for LFM. One small difference is where MF is the baseline: for larger values of $\lambda$, xQuAD and its variations slightly outperform SPAD and

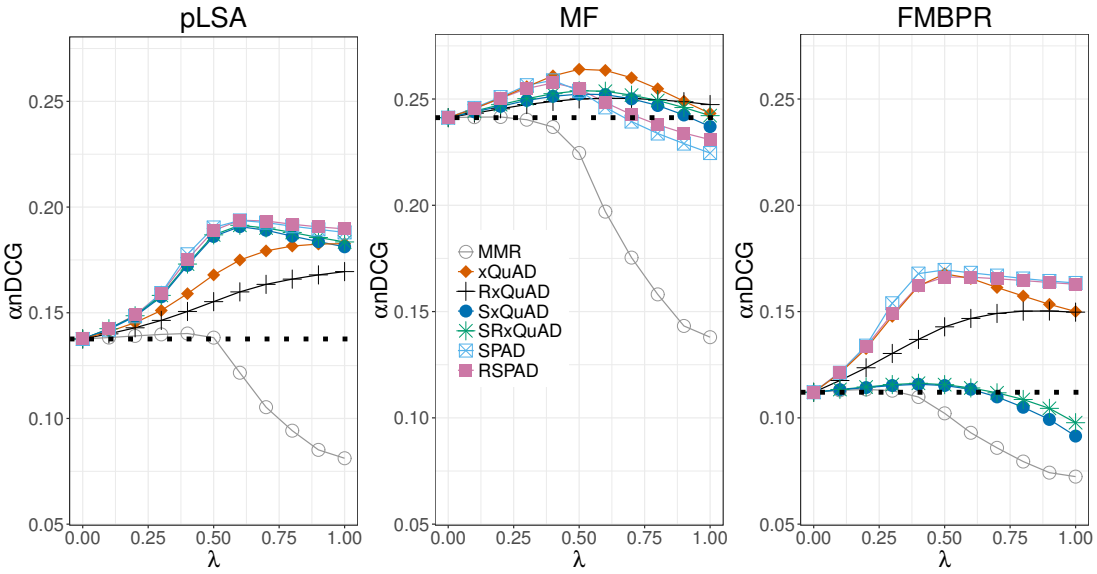



Figure 5.6: LT dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-nDCG for each baseline is shown by the dotted line.

RSPAD. For $\alpha$-nDCG, for almost all values of $\lambda$, SPAD and RSPAD perform better than all the other re-ranking approaches where pLSA and FMBPR are the baselines. But where MF is the baseline, xQuAD has higher $\alpha$-nDCG for almost all values of $\lambda$.

To summarize, the results in this subsection and the previous one indicate that SPAD and RSPAD perform the best, i.e. they are always the most accurate systems across all three datasets evaluated in this chapter, and they suffer least from the relevance/diversity trade-off. They increase relevance as well as the diversity for even more configurations than the other approaches do. We repeat the observation too that SPAD and RSPAD are at a disadvantage in the results for diversity and this may explain why the other approaches sometimes have higher diversity than SPAD and RSPAD.

## 5.2 Diversity Measured by Subprofiles

As we have mentioned before, the diversity metrics that we use to evaluate the systems in the experiments are computed with respect to the item features $\mathcal{F}$. Since SPAD and RSPAD make no use of the item features at all, they are at a disadvantage. In this section, we adapt the diversity metrics so that they use subprofiles instead of item features. We use these in the following subsections to give results in which SPAD and RSPAD are not at so much of a disadvantage.

By having both sets of results, we get a more rounded picture of the relative performance of the SPAD, RSPAD and their competitors.

We divide this section into two. First, we give the definition of the adapted diversity metrics. Next, we give the diversity results for both versions of the diversity metrics, i.e. diversity measured by item features and subprofiles.

### 5.2.1 Modified diversity metrics

We adapt the $\alpha$-nDCG, ERR-IA, S-recall, EILD and ILD metrics so that they use subprofiles instead of item features.

For $\alpha$-nDCG, the adapted version is:

$$\alpha\text{-nDCG}(L) = \frac{1}{\alpha\text{-IDCG}} \sum_{i \in L} \left[ \frac{1}{\log_2(\text{r}(i,L)+1)} \sum_{S \in \mathcal{S}_u^*} \text{rel}(i|u,S) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - \alpha\, \text{rel}(j|u,S)) \right] \tag{5.1}$$

where we modify the original $\alpha$-nDCG metric by writing $rel(i|u,S)$ instead of $rel(i|u,f)$ such that $rel(i|u,S)$ is 1 if item $i$ is *related to* (See Eq. 4.4) subprofile $S$ and user $u$ but 0 otherwise. $\mathcal{S}_u^*$ is the final set of subprofiles for user $u$. We will refer to $\alpha$-nDCG measured by item features as $\alpha\text{-nDCG}^{\mathcal{F}}$ and $\alpha$-nDCG measured by subprofiles as $\alpha\text{-nDCG}^{\mathcal{S}}$.

For ERR-IA:

$$\text{ERR-IA}(L) = \sum_{S \in \mathcal{S}_u^*} p(S|u) \sum_{i \in L} \frac{1}{\text{r}(i,L)} p(\text{rel}\,|i,u,S) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - p(\text{rel}\,|j,u,S)) \tag{5.2}$$

where we modify the original ERR-IA metric by writing $p(\text{rel}\,|i,u,S)$ instead of $p(\text{rel}\,|i,u,f)$, such that $p(\text{rel}\,|i,u,S)$ is the probability that user $u$ finds recommended item $i$ relevant when interested in subprofile $S$. We will refer to

ERR-IA measured by item features as ERR-IA$^{\mathcal{F}}$ and ERR-IA measured by subprofiles as ERR-IA$^{\mathcal{S}}$.

We modify S-recall as:

$$\text{S-recall}(L) = \frac{|\cup_{i \in L} \mathcal{S}_i|}{|\mathcal{S}_u^*|} \tag{5.3}$$

where $\mathcal{S}_i$ is the set of subprofiles that item $i$ is *related to* such that $\mathcal{S}_i \subseteq \mathcal{S}_u^*$ and $\mathcal{S}_u^*$ is the final set of subprofiles for user $u$. We will refer to S-recall measured by item features as S-recall$^{\mathcal{F}}$ and S-recall measured by subprofiles as S-recall$^{\mathcal{S}}$.

To adapt EILD and ILD to SPAD, instead of computing $\text{dist}(i,j)$ using item features, we compute $\text{dist}(i,j)$ as:

$$\text{dist}(i,j) = \frac{|\mathcal{S}_i \cap \mathcal{S}_j|}{|\mathcal{S}_i \cup \mathcal{S}_j|} \tag{5.4}$$

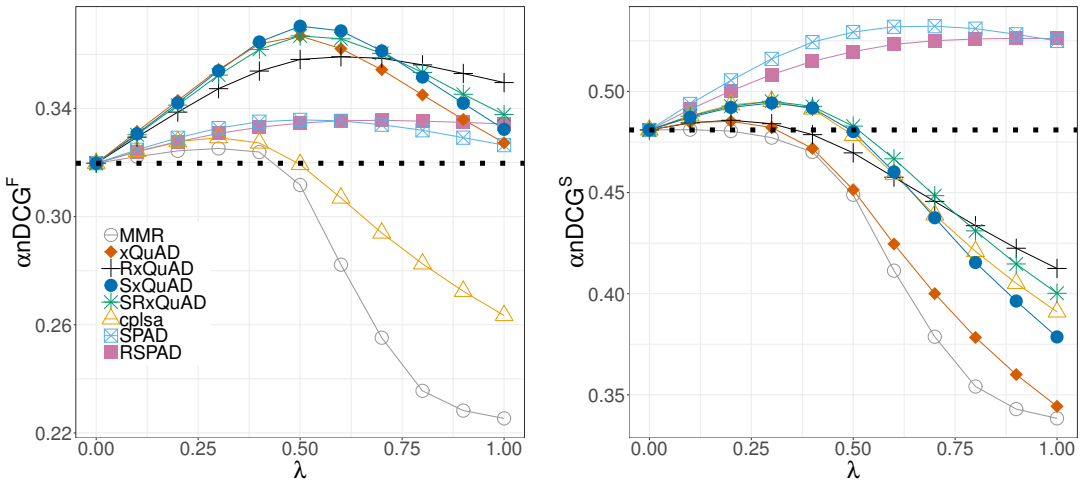
where $\mathcal{S}_i$, $\mathcal{S}_j$ are the set of subprofiles that item $i$ and $j$ are *related to* such that $\mathcal{S}_i, \mathcal{S}_j \subseteq \mathcal{S}_u^*$ and $\mathcal{S}_u^*$ is the final set of subprofiles for user $u$. Again, we refer to ILD and EILD measured by item features as ILD$^{\mathcal{F}}$ and EILD$^{\mathcal{F}}$, and ILD and EILD measured by subprofiles as ILD$^{\mathcal{S}}$ and EILD$^{\mathcal{S}}$.

### 5.2.2 Results for different values of $\lambda$

As we have mentioned earlier, SPAD and RSPAD make no use of the explicit item features. The competitor algorithms do use item features. Hence, SPAD and RSPAD are at a disadvantage in those experiments where we measure diversity, since all the diversity metrics we have used so far are defined in terms of explicit item features. In the case of ERR-IA, xQuAD has a particular advantage: the probabilities we use when computing ERR-IA are the ones computed by xQuAD, even we are evaluating an algorithm other than xQuAD.

In the previous subsection, we defined a set of diversity metrics, corresponding to the original ones, but defined in terms of subprofiles rather than item features. These metrics favour SPAD and RSPAD. By showing the values of both sets of diversity metrics side by side, we obtain a more balanced picture of the performance of the diversification algorithms.

We will not repeat all results. As baseline algorithm, we select MF, since, as



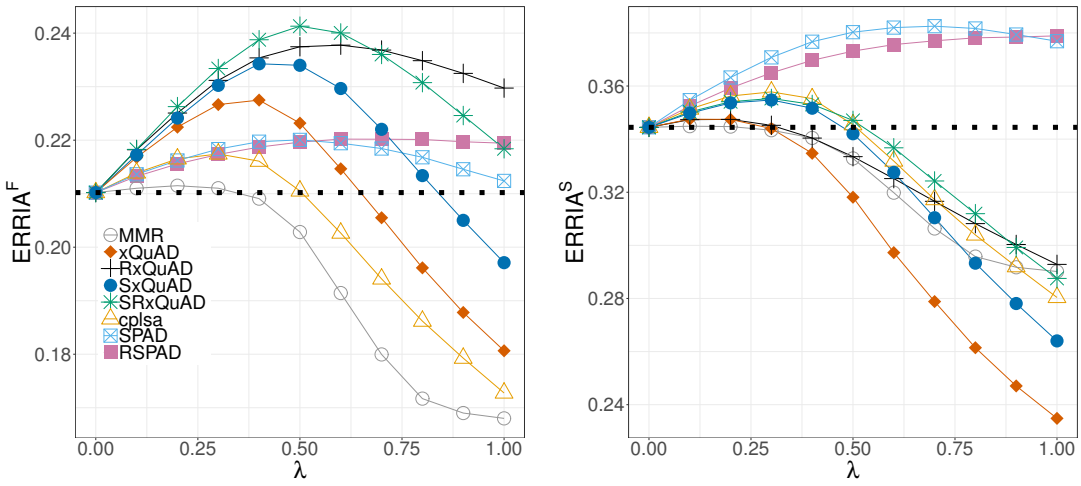

(a) $\alpha$-nDCG$^{\mathcal{F}}$ on ML1M (b) $\alpha$-nDCG$^{\mathcal{S}}$ on ML1M

Figure 5.7: ML1M dataset. $\alpha$-nDCG measured using features and subprofiles for varying $\lambda$.

mentioned before, all of the re-ranking algorithms achieve both their best precision and $\alpha$-nDCG values using MF as the baseline recommender. We use the ML1M dataset for the analysis in this section since it is the most widely used dataset in the literature. The subprofiles we use when computing the new metrics are the ones detected by NN-1. The hyperparameters we use for NN-1 are the ones that optimize $\alpha$-nDCG on the validation set for the MF baseline recommender; see Table A.2.

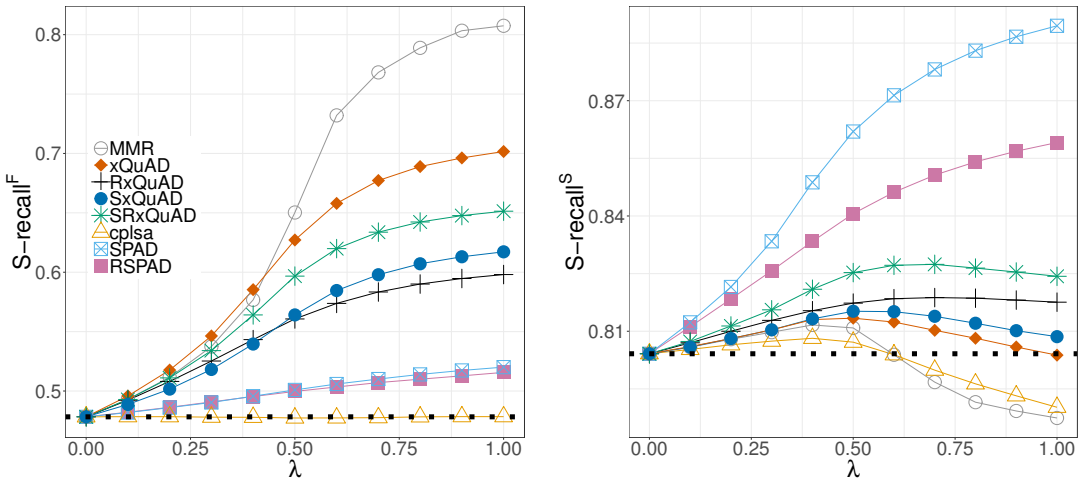
Figures 5.7, 5.8, 5.9, 5.10 and 5.11 show $\alpha$-nDCG, ERR-IA, S-recall, EILD and ILD measured by item features and measured by subprofiles. We plot the metrics computed on the test set for different values $\lambda$.

For the $\alpha$-nDCG metric, see Figure 5.7. First consider $\alpha$-nDCG measured by item features, $\alpha$-nDCG$^{\mathcal{F}}$, see Figure 5.7a. SPAD and RSPAD achieve higher values than the baseline for all values of $\lambda$. But, as expected, xQuAD and its variants have higher $\alpha$-nDCG$^{\mathcal{F}}$ than SPAD and RSPAD for many values of $\lambda$. c-pLSA and MMR, on the other hand, start to suffer from decreases in $\alpha$-nDCG$^{\mathcal{F}}$ for values of $\lambda$ from about 0.5, and for many values of $\lambda$ they perform worse than SPAD and RSPAD. Next, consider $\alpha$-nDCG measured by subprofiles, $\alpha$-nDCG$^{\mathcal{S}}$ shown in Figure 5.7b. As expected, SPAD and RSPAD perform better than all of the other re-ranking approaches and the baseline for all values of $\lambda$. All of the other approaches start to suffer from decreases in $\alpha$-nDCG$^{\mathcal{S}}$ for values of $\lambda$ from about 0.5. SPAD and RSPAD are the only methods that always increase $\alpha$-nDCG compared with the baseline, irrespective of whether it is measured




(a) ERR-IA$^{\mathcal{F}}$ on ML1M (b) ERR-IA$^{\mathcal{S}}$ on ML1M

Figure 5.8: ML1M dataset. ERR-IA measured using features and subprofiles for varying $\lambda$.



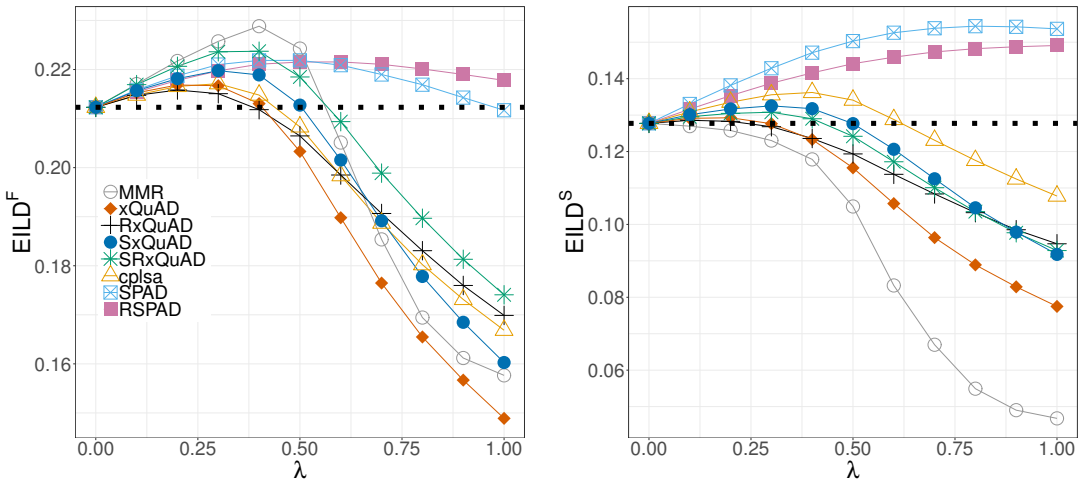

(a) S-recall$^{\mathcal{F}}$ on ML1M (b) S-recall$^{\mathcal{S}}$ on ML1M

Figure 5.9: ML1M dataset. S-recall measured using features and subprofiles for varying $\lambda$.

using item features or using subprofiles.

Figure 5.8 shows ERR-IA measured by item features and by subprofiles. The results are quite similar to those for $\alpha$-nDCG (see Figure 5.7). What differs is that, for ERR-IA$^{\mathcal{F}}$ (Figure 5.8a), xQuAD and SxQuAD start to suffer from decreases in ERR-IA$^{\mathcal{F}}$ for values of $\lambda$ from about 0.6. Again, SPAD and RSPAD are the only methods that always increase the values of ERR-IA over the baseline, irrespective of whether it is measured by item features or by subprofiles.

The results for S-recall are shown in Figure 5.9. First, consider S-recall measured



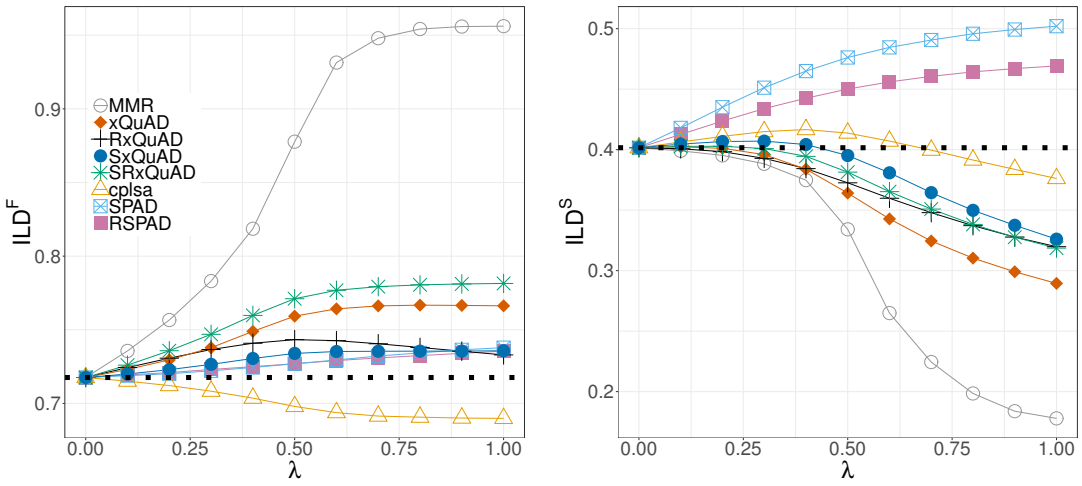

(a) $\text{EILD}^{\mathcal{F}}$ on ML1M (b) $\text{EILD}^{\mathcal{S}}$ on ML1M

Figure 5.10: ML1M dataset. EILD measured using features and subprofiles for varying $\lambda$.

by using item features, $\text{S-recall}^{\mathcal{F}}$ shown in Figure 5.9a. With respect to $\text{S-recall}^{\mathcal{F}}$, all of the re-ranking approaches, except c-pLSA, outperform the baseline for all values of $\lambda$. MMR, xQuAD and its variants perform better than SPAD and RSPAD for all values of $\lambda$. Next, consider Figure 5.9b, which shows S-recall measured by subprofiles, $\text{S-recall}^{\mathcal{S}}$. SPAD and RSPAD improve the value of $\text{S-recall}^{\mathcal{S}}$ compared with the baseline for all values of $\lambda$. MMR and c-pLSA start to suffer from decreases for values of $\lambda$ from about 0.6. xQuAD and its variants increase the values of $\text{S-recall}^{\mathcal{S}}$ with respect to the baseline for almost all values of $\lambda$.

Figure 5.10 shows the results for EILD. First, consider EILD measured by item features, $\text{EILD}^{\mathcal{F}}$, shown in Figure 5.10a. SPAD and RSPAD increase the values of EILD with respect to the baseline for almost all values of $\lambda$. All of the other re-ranking approaches start to suffer from decreases in $\text{EILD}^{\mathcal{F}}$ for values of $\lambda$ from about 0.6. Next, consider EILD measured by subprofiles, $\text{EILD}^{\mathcal{S}}$, which is shown in Figure 5.10b. Similar to $\text{EILD}^{\mathcal{F}}$, MMR, xQuAD and its variants start to suffer from decreases in EILD for values of $\lambda$ from about 0.6. SPAD and RSPAD again increase $\text{EILD}^{\mathcal{S}}$ with respect to the baseline for all values of $\lambda$, and they perform better than all of the other re-ranking approaches.

Finally, the results for ILD are shown in Figure 5.11. Looking at Figure 5.11a, we see that all of the re-ranking approaches increase $\text{ILD}^{\mathcal{F}}$ with respect to the baseline for all values of $\lambda$ with one exception only. The exception is c-pLSA, where it always suffers from decreases in $\text{ILD}^{\mathcal{F}}$. As expected, MMR is the best performing algorithm. This is not surprising: $\text{ILD}^{\mathcal{F}}$ is a metric that is very



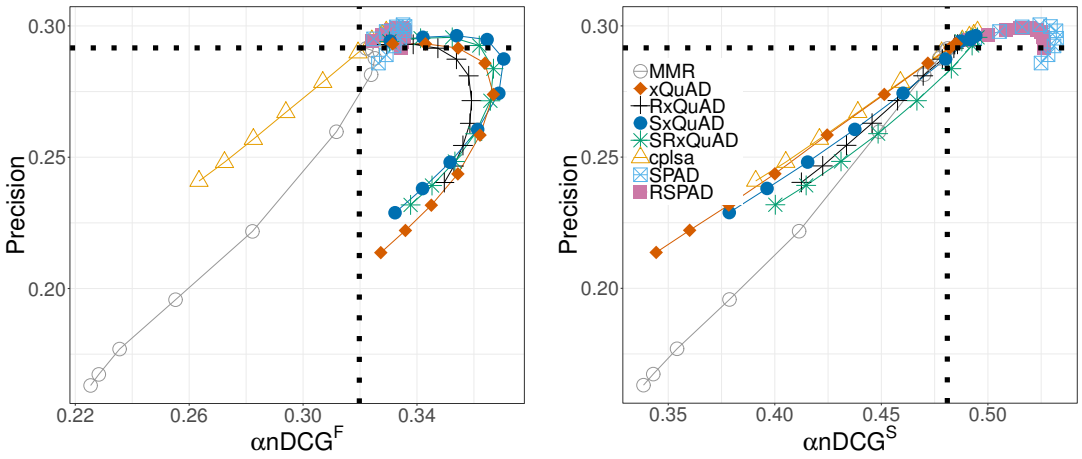

(a) $\text{ILD}^{\mathcal{F}}$ on ML1M (b) $\text{ILD}^{\mathcal{S}}$ on ML1M

Figure 5.11: ML1M dataset. ILD measured using features and subprofiles for varying $\lambda$.

similar to what is used for re-ranking in MMR. When we consider ILD measured by subprofiles, $\text{ILD}^{\mathcal{S}}$ (Figure 5.11b), the results are very similar to those for $\text{EILD}^{\mathcal{S}}$ (see Figure 5.11b). SPAD and RSPAD increase $\text{ILD}^{\mathcal{S}}$ over the baseline for all values of $\lambda$, and they perform better than the other algorithms.

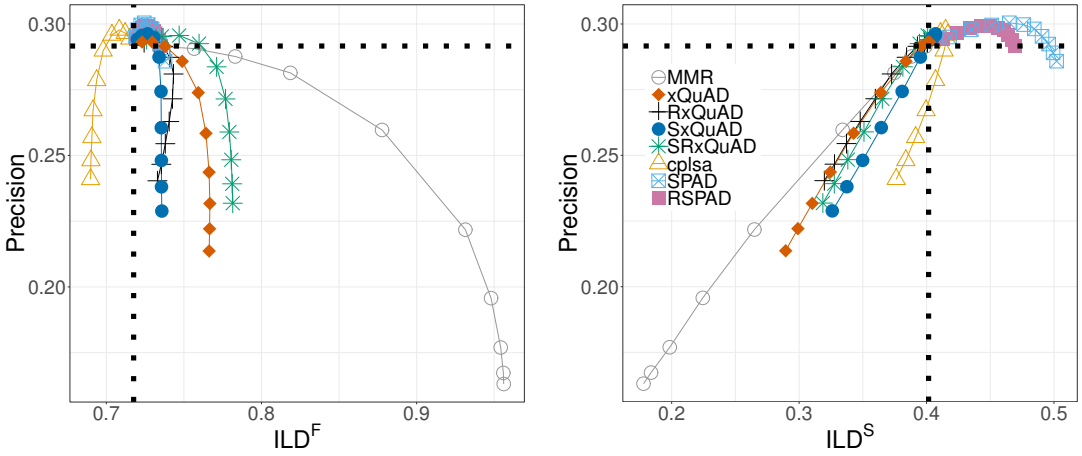
The results in this section have confirmed that SPAD and RSPAD are at a disadvantage in experiments where diversity metrics use item features. Figures 5.7, 5.8, 5.9, 5.10 and 5.11 show that SPAD and RSPAD improve the diversity metrics with respect to the baseline when measured by subprofiles for all five diversity metrics and for all values of $\lambda$, and they always perform better than competitor algorithms. This is as expected, since SPAD and RSPAD uses subprofiles for re-ranking; in this case, it is the other algorithms that are at a disadvantage since they make no use of subprofiles.

The results showing diversity metrics measured by item features are noteworthy, since, even though SPAD and RSPAD make no use of item features, for almost all five diversity metrics and for all values of $\lambda$, they improve the scores of the metrics with respect to the baseline, and there are configurations where they perform better than the competitor algorithms. We argue that, this shows the value of the additional complexity of SPAD and RSPAD's subprofile-aware approach over other re-ranking algorithms in terms of diversity.




(a) Precision/$\alpha$-nDCG$^{\mathcal{F}}$ on ML1M (b) Precision/$\alpha$-nDCG$^{\mathcal{S}}$ on ML1M

Figure 5.12: ML1M dataset. Precision vs. $\alpha$-nDCG trade-off plots.




(a) Precision/ILD$^{\mathcal{F}}$ on ML1M (b) Precision/ILD$^{\mathcal{S}}$ on ML1M

Figure 5.13: ML1M dataset. Precision vs. ILD trade-off plots.

## 5.3 A Visualization of Trade-offs

In this section, we show the trade-off between relevance and diversity for the re-ranking approaches. Figure 5.12 show precision against diversity, where diversity is measured by two versions of $\alpha$-nDCG, $\alpha$-nDCG$^{\mathcal{F}}$ and $\alpha$-nDCG$^{\mathcal{S}}$. The dotted lines show the precision and diversity of the MF baseline, dividing each subfigure into four quadrants. The 'sweet spot' is the top-right quadrant, where both precision and diversity scores are higher than the baseline. We can see that, for both $\alpha$-nDCG$^{\mathcal{F}}$ and $\alpha$-nDCG$^{\mathcal{S}}$, SPAD and RSPAD more often occupy this 'sweet spot' because they are less likely to trade-off precision for diversity. For $\alpha$-nDCG$^{\mathcal{S}}$, this is as expected. But, for $\alpha$-nDCG$^{\mathcal{F}}$, it is noteworthy, since

SPAD and RSPAD are not using item features.

Figure 5.13 shows the same trade-off, this time by measuring diversity using two versions of ILD, $\mathrm{ILD}^{\mathcal{F}}$ (Figure 5.13a) and $\mathrm{ILD}^{\mathcal{S}}$ (Figure 5.13b). It is worth showing these results because ILD is in some sense a 'purer' measure of diversity compared with $\alpha$-nDCG, which mixes ranking and diversity. The results are similar to those shown for $\alpha$-nDCG in Figure 5.12, in the sense that SPAD and RSPAD are less prone to the relevance/diversity trade-off.

## 5.4 Conclusions

In this chapter, we compared SPAD and RSPAD with several existing diversification techniques in the literature. We adapted several diversity metrics so that they are defined in terms of subprofiles instead of item features, and analyzed the performances of different re-ranking approaches with respect to both versions of the diversity metrics. We also showed the relevance/diversity trade-off.

All of the results in this chapter indicate that SPAD and RSPAD perform the best, i.e. they are always produce the most relevant recommendations across all three datasets evaluated in this chapter; they increase relevance as well as the diversity (no matter how diversity is measured) for more configurations than the other approaches do; and they suffer least from the relevance/diversity trade-off.

In the next chapter, we propose Community-Aware Diversification (CAD), in which aspects are again subprofiles but are detected indirectly through users who are similar to the active user.

# Chapter 6

# Community-Aware Diversification

Intent-aware methods for recommendation diversification seek to ensure that the recommended items cover so-called *aspects*, which are assumed to define the user's tastes and interests, to a certain extent. Most typically, aspects are item features. In chapter 4, we presented a novel intent-aware diversification method, called Subprofile-Aware Diversification (SPAD). It does not use item features at all. Instead, aspects are subprofiles of a user's profile, and subprofiles are defined in terms of item-item similarities on the items' ratings vectors, rather than on item features. The main contribution of SPAD is that, across multiple datasets, it can improve both accuracy and diversity, and it suffers least from the relevance/diversity trade-off (see Chapter 5). This is notable because early approaches to diversification sacrifice accuracy for diversity; even other intent-aware approaches sometimes sacrifice accuracy for diversity.

In this chapter, we propose Community-Aware Diversification (CAD), in which aspects are again subprofiles but are detected indirectly through users who are similar to the active user. It uses user-user similarities on user ratings vectors, but it still results in subprofiles, each of which is a subset of the *items* in the user's profile. Thus it explores the idea that a user's community (similar users) correlates with her tastes or interests (subprofiles).

The rest of this chapter is structured as follows. In the next section, we give the details of the Community-Aware diversity. Then, in Section 6.2, we report empirical results of CAD using offline experiments.

## 6.1 Community-Aware Diversity

In this section, we explain our new approach to diversification in recommender systems, which we call Community-Aware Diversification (CAD). It is a greedy re-ranking approach; it is intent-aware; like SPAD, it uses subprofiles of the user's liked-items-set $I_u^+$ as aspects; but it identifies subprofiles within $I_u^+$ by using her nearest-neighbours, i.e. other *users* similar to her, not by using the item's neighbours.

We define a candidate subprofile for each item $i \in I_u^+$. Hence, the number of candidate subprofiles is $|I_u^+|$. To simplify, consider user $u$ and one of her $k$-nearest-neighbours, $v$. If item $i$ (which we know to be a member of $u$'s liked-item-set) is also a member of $v$'s liked-item-set ($I_v^+$), then we take the intersection of $u$'s and $v$'s liked-item-sets, $I_u^+ \cap I_v^+$. Inevitably, this intersection will contain item $i$, but it may contain other items too.

We compute $I_u^+ \cap I_v^+$ for all of $u$'s neighbours who like $i$ ($v \in \mathrm{KNN}(u), i \in I_v^+$). So now we have up to $k$ sets, one for each $v \in \mathrm{KNN}(u)$ provided $i \in I_v^+$. We aggregate these sets to give a candidate subprofile. More formally, we have that, for user $u$, the candidate subprofile that corresponds to item $i$, $S_u^i$, is given by:

$$S_u^i = \bigstar_{v \in \mathrm{KNN}(u), i \in I_v^+} I_u^+ \cap I_v^+ \tag{6.1}$$

But this leaves open how to do the aggregation of the intersections, $\bigstar$.

One possibility it to take their intersection (an intersection of intersection): an item $j$ is in $i$'s candidate subprofile if it is liked by *all* of the users in $\mathrm{KNN}(u)$ who liked $i$:

$$S_u^i = \bigcap_{v \in \mathrm{KNN}(u), i \in I_v^+} I_u^+ \cap I_v^+ \tag{6.2}$$

Instead of using intersection, another possibility is to take the union of the intersections, as follows:

$$S_u^i = \bigcup_{v \in \mathrm{KNN}(u), i \in I_v^+} I_u^+ \cap I_v^+ \tag{6.3}$$

in which case an item $j$ is in $i$'s candidate subprofile if it is liked by *any* of the users in $\mathrm{KNN}(u)$ who liked $i$.

With both approaches there are problems. Eq. 6.2 can result in a lot of singleton subprofiles: $i$ might be the only item that the neighbours have in common.

There is nothing wrong with singleton profiles if they contain idiosyncratic items. But, in general, our goal is to try to group items into subprofiles that capture tastes and interests, which Eq. 6.2 may fail to do very often.

On the other hand, taking the union, as in Eq. 6.3, results in large subprofiles, perhaps even some for which $S_u^i = I_u^+$. It may place into the same subprofile items that represent different tastes and interests.

In CAD, we use a unified approach, the $q$-relaxed set intersection, $q \in [0, 1]$

$$S_u^i = \bigcap_{v \in \mathrm{KNN}(u), i \in I_v^+}^{\{q\}} I_u^+ \cap I_v^+ \tag{6.4}$$

where an item $j$ will be in $i$'s candidate subprofile if it is liked by a proportion of at least $q$ of the users in $\mathrm{KNN}(u)$ who liked $i$.

When $q = 1.0$, Eq. 6.4 is the same as Eq. 6.2, and when $q = 1/|\{v|v \in \mathrm{KNN}(u), i \in I_v^+\}|$, it gives the same results as Eq. 6.3.

For CAD, $q$ is a hyperparameter, whose value will be set using a validation set (methodology explained in Chapter 3 ). As we will show, it tends to pick quite high values for $q$, between 0.7 and 1.0, so it tends to be closer to Eq. 6.2 than Eq. 6.3. CAD also has another hyperparameter, number of neighbouring users (Eq. 6.1), that we refer to as $k_{UB}$.

What we have at this stage are candidate subprofiles, one per $i \in I_u^+$. As in SPAD (Eq. 4.6), the final step is to prune the candidate subprofiles: we eliminate any that are subsets of the others. The remaining subprofiles are treated as aspects in the following equation (the details are given in Section 4.2.1):

$$\mathrm{div}_{\mathrm{SPAD}}(i, RL) = \sum_{S \in \mathcal{S}_u} [p(S|u)p(i|u, S) \prod_{j \in RL} (1 - p(j|u, S))] \tag{6.5}$$

Before moving on to the experiments, we will mention two other approaches that did not work and that we discarded. In one of the rejected approaches, instead of using proportion $q$ in Eq. 6.4, we tried an integer threshold $\theta$, requiring items to be members of least $\theta$ of the set intersections. We also tested a variation of CAD in which aspects were not sets of items (subprofiles of $I_u^+$); rather, we used neighbours more directly than is done in Eq. 6.1 — aspects were sets of users, nearest-neighbours who liked $i$, $S_u^i = \{v \in \mathrm{KNN}(u) \mid i \in I_v^+\}$. Neither of these two other approaches worked well enough for us to show their results

in this chapter.

## 6.2 Experimental Results

In this section, we report our empirical investigation of CAD using offline experiments.

We use the same datasets that we used in the previous chapter, the ML1M dataset, the LFM dataset and the LT dataset. We also use the Facebook (FB) dataset in this chapter. Since we detect a user's subprofiles from the user's community (i.e. similar users), we wanted to investigate the effect of using user-based subprofiles on a social-networking dataset. For the details of the datasets see Chapter 3.

We compare CAD with SPAD and one other intent-aware diversification technique, xQuAD [VCV11], using the implementation which is available in the RankSys library. We have also compared CAD and SPAD with a number of other intent-aware diversification methods (RxQuAD [VCV12], SxQuAD & SRxQuAD [VC13], and c-pLSA [WH16]) and a more classical (non-intent-aware) method, MMR [CG98], similar to the previous chapter. In this chapter, we only show the results for xQuAD since, across all datasets, it was the most competitive of these other techniques.

All of these approaches to diversification use greedy re-ranking, therefore they need a baseline recommender, whose recommendation sets are re-ranked. We use the same baseline recommenders that we use in the chapters 4 and 5: probabilistic latent semantic analysis (pLSA) [Hof04], a fast alternative least-squares matrix factorization recommender (MF) [PZT10], and a factorization machine that uses Bayesian pairwise loss for ranking (FMBPR) [Bay15]. Three baselines paired with four approaches to re-ranking (CAD, SPAD, xQuAD and none at all) gives twelve systems to compare on each dataset.

The experimental setup and evaluation methodology is explained in Chapter 3 of the thesis. We show the hyperparameter values that we used in the Appendix to this thesis (Section A.3).

We divide this section into five: first we analyze the subprofiles that CAD and SPAD find in each dataset; second we give results that compare CAD with SPAD and xQuAD; next, we show the results for different values of $\lambda$; then, we show

Table 6.1: Subprofile statistics

| | | |
|---|---|---|
| ML1M | SPAD | avg. 51.09 ($\sigma = 61.5$) subprofiles per user<br>avg. len of subprofiles is 7.78 ($\sigma = 6.25$)<br>avg. sim of subprofiles is 0.0379 |
| | CAD | avg. 60.22 ($\sigma = 76.39$) subprofiles per user<br>avg. len of subprofiles is 34.01 ($\sigma = 25.28$)<br>avg. sim of subprofiles is 0.4007 |
| LFM | SPAD | avg. 134.37 ($\sigma = 161.41$) subprofiles per user<br>avg. len of subprofiles is 30.08 ($\sigma = 28.11$)<br>avg. sim of subprofiles is 0.1045 |
| | CAD | avg. 52.04 ($\sigma = 29.95$) subprofiles per user<br>avg. len of subprofiles is 16.62 ($\sigma = 35.81$)<br>avg. sim of subprofiles is 0.0523 |
| LT | SPAD | avg. 32.49 ($\sigma = 46.71$) subprofiles per user<br>avg. len of subprofiles is 8.8 ($\sigma = 10.72$)<br>avg. sim of subprofiles is 0.044 |
| | CAD | avg. 21.55 ($\sigma = 12.41$) subprofiles per user<br>avg. len of subprofiles is 6.1 ($\sigma = 9.85$)<br>avg. sim of subprofiles is 0.059 |
| FB | SPAD | avg. 24.29 ($\sigma = 25.83$) subprofiles per user<br>avg. len of subprofiles is 5.04 ($\sigma = 5.66$)<br>avg. sim of subprofiles is 0.0208 |
| | CAD | avg. 32.59 ($\sigma = 31.23$) subprofiles per user<br>avg. len of subprofiles is 4.85 ($\sigma = 6.63$)<br>avg. sim of subprofiles is 0.101 |

the trade-off between precision and diversity; lastly, we show diversity measures evaluated by using item features vs. subprofiles.

### 6.2.1 Analysis of the subprofiles

In this section we compare the subprofiles detected by CAD and SPAD on each dataset. Table 6.1 shows the average number of subprofiles per user and the average length of the subprofiles. We also show the average similarity of the subprofiles with each other, which, for user $u$, is an all-pairs average (see Eq. 4.13).

Consider, the ML1M dataset first. On average, SPAD extracts ∼51 subprofiles per user and they consist of ∼8 movies. CAD extracts more subprofiles (∼60) and those subprofiles have a lot more movies (∼34) compared with SPAD's subprofiles; CAD's subprofiles are also much more similar to each other.

Next, consider the LFM dataset. SPAD extracts an average of ∼134 subprofiles, having ∼30 artists, per user. CAD extracts fewer subprofiles (∼52) with a little over half as many artists (∼17). SPAD's subprofiles are more similar to each other than CAD's.

Then, consider the LT dataset. There are on average ∼32 subprofiles per user, which have ∼9 books, using SPAD. Using CAD gives fewer subprofiles (∼22), having fewer books (∼6). SPAD subprofiles are not very similar to each other, and neither are CAD subprofiles.

Finally, consider the FB dataset. Using CAD results in more subprofiles (∼33) than SPAD (∼24), and they are more similar to each other. The average length of the subprofiles are almost the same (∼5).

### 6.2.2 Results for different algorithms

In this section, we compare CAD with SPAD, xQuAD and the baseline algorithms (no re-ranking). For each of the three baselines, for all four datasets, we show precision and $\alpha$-nDCG in one table, and other diversity metrics (ERR-IA, S-recall, EILD and ILD) in another table. Note that the diversity metrics used in this section are the ones defined in Chapter 3, i.e. they are measured by using item features. Later, in Section 6.2.4, we present the results for the subprofile-based diversity metrics as well (see Section 5.2.1 for the definitions). There, we use superscripts $F$ and $S$ to refer to item-based and subprofile-based diversity metrics respectively. But in this section, since no ambiguity can arise, to simplify the notation, we do not use superscript $F$ to refer to item-based diversity metrics.

The best result for each metric is highlighted in bold for each block of the six tables. The value of $\lambda$ that optimizes $\alpha$-nDCG for each baseline and re-ranking strategy is given. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$), except those shown in italics. For CAD and SPAD, if improvements over xQuAD are statistically significant, they are highlighted with ▲; and if their improvements over each other are statistically significant, they are marked with △.

The results for the experiments that use MF as the baseline algorithm are in Table 6.2, and other diversity metrics are given in Table 6.3. In each block of the table, results for the MF baseline are presented first, and then results

Table 6.2: Results using MF as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| MF | | 0.2916 | 0.3197 | | |
| xQuAD | 0.5 | 0.2739 | **0.3668** | -6.08 | 14.72 |
| SPAD | 0.4 | **0.3005** ▲△ | 0.3351 | 3.03 | 4.81 |
| CAD | 0.3 | 0.2982 ▲ | 0.3354 | 2.27 | 4.89 |
| LFM | | | | | |
| MF | | 0.4654 | 0.4244 | | |
| xQuAD | 0.3 | 0.4701 | **0.4354** | 1.01 | 2.61 |
| SPAD | 0.2 | **0.4742** ▲ | 0.4296 △ | 1.9 | 1.24 |
| CAD | 0.3 | 0.472 | 0.4275 | 1.42 | 0.75 |
| LT | | | | | |
| MF | | 0.1733 | 0.2412 | | |
| xQuAD | 0.5 | 0.1866 | **0.264** | 7.7 | 9.44 |
| SPAD | 0.4 | 0.1896 ▲ | 0.2588 | 9.4 | 7.28 |
| CAD | 0.4 | **0.1899** ▲ | 0.2603 △ | 9.59 | 7.92 |
| FB | | | | | |
| MF | | 0.1341 | 0.1613 | | |
| xQuAD | 0.4 | 0.13 | **0.1791** | -3.0 | 10.99 |
| SPAD | 0.4 | 0.1428 ▲ | 0.1721 | 6.53 | 6.7 |
| CAD | 0.4 | **0.1436** ▲△ | 0.1758 △ | 7.74 | 8.82 |

for each of the re-ranking methods are given. For each method, we report the results using the value of $\lambda$ that gives highest $\alpha$-nDCG on the validation set.

Consider precision and $\alpha$-nDCG, that are shown in Table 6.2 first. For all four datasets, CAD has higher precision than xQuAD and the MF baseline, and it has higher precision than SPAD for the LT and FB datasets as well. CAD's advantage over the MF baseline is statistically significant in all cases, and its advantage over xQuAD is also statistically significant in all but the case of LFM. Its precision is higher than SPAD's for the LT and FB datasets but this is only statistically significant for the FB dataset. SPAD, on the other hand, has highest precision for the ML1M and LFM datasets and, unlike CAD, for all datasets its advantage over xQuAD is statistically significant. Its advantage over CAD, however, is only statistically significant for the ML1M dataset.

For diversity, xQuAD has the highest $\alpha$-nDCG for all datasets. But CAD and SPAD are at the disadvantage that we have discussed previously. Even so, CAD and SPAD have statistically significantly higher diversity than the MF baseline for all datasets. Compared with each other, CAD has higher diversity than SPAD

Table 6.3: Diversity metrics except $\alpha$-nDCG using MF baseline.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| ML1M | | | | | | | | | |
| MF | | 0.2102 | 0.4783 | 0.2123 | 0.7176 | | | | |
| xQuAD | 0.5 | **0.2232** | **0.6271** | 0.2033 | **0.7592** | 6.17 | 31.13 | -4.24 | 5.80 |
| SPAD | 0.4 | 0.2197 | 0.4957 | 0.2219▲ | 0.7244 | 4.52 | 3.63 | 4.5 | 0.95 |
| CAD | 0.3 | 0.2188 | 0.4988 | **0.2225**▲ | 0.7337△ | 4.06 | 4.29 | 4.81 | 2.25 |
| LFM | | | | | | | | | |
| MF | | 0.2012 | 6.39E-4 | 0.3671 | 0.7638 | | | | |
| xQuAD | 0.3 | **0.2067** | **6.61E-4** | **0.3779** | **0.778** | 2.74 | 3.4 | 2.94 | 1.87 |
| SPAD | 0.2 | *0.2036* | 6.36E-4 | 0.3734 | 0.762 | 1.22 | -0.53 | 1.87 | -0.23 |
| CAD | 0.3 | 0.2042 | *6.39E-4*△ | 0.3715 | *0.7642*△ | 1.51 | -0.09 | 1.2 | 0.06 |
| LT | | | | | | | | | |
| MF | | 0.1143 | 0.0097 | 0.1137 | 0.7577 | | | | |
| xQuAD | 0.5 | **0.1213** | **0.0104** | **0.1314** | **0.7884** | 6.19 | 7.17 | 15.53 | 4.05 |
| SPAD | 0.4 | 0.1183 | 0.0099△ | 0.1302△ | 0.7726△ | 3.57 | 2.18 | 14.48 | 1.97 |
| CAD | 0.4 | 0.1197△ | 0.0098 | 0.1286 | 0.7667 | 4.79 | 0.87 | 13.13 | 1.19 |
| FB | | | | | | | | | |
| MF | | 0.018 | 0.0048 | 0.1157 | 0.9708 | | | | |
| xQuAD | 0.4 | **0.0219** | **0.0059** | 0.1076 | 0.9657 | 21.8 | 22.21 | -6.94 | -0.53 |
| SPAD | 0.4 | 0.0196 | 0.0049 | 0.1254▲ | 0.9711▲ | 8.82 | 0.67 | 8.44 | 0.03 |
| CAD | 0.4 | 0.0202△ | 0.005△ | **0.1262**▲△ | **0.9718**▲△ | 12.27 | 2.4 | 9.08 | 0.11 |

for all but the LFM dataset where SPAD is statistically significantly higher than CAD.

If we look at the percentage changes, for the ML1M and FB datasets, xQuAD achieves highest diversity at the expense of a decrease in precision: it trades-off accuracy for diversity. CAD and SPAD, on the other hand, always increase both accuracy and diversity. Arguably, for the ML1M and LFM datasets SPAD gives a better trade-off between precision and diversity compared to CAD. For LT and FB, it is CAD that gives the better trade-off.

Next, consider the other diversity metrics shown in Table 6.3. xQuAD has the highest ERR-IA and S-recall for all four datasets, but it achieves this at the cost of lower precision for the ML1M and FB datasets. It also has the highest ILD for all datasets with one exception only. The exception is for FB, where CAD has the highest ILD. For EILD, CAD has the highest EILD values for the ML1M and FB datasets and xQuAD has the highest EILD for the other two datasets. CAD and SPAD, surprisingly, almost for all datasets and diversity metrics improve the performances of their baselines. It is surprising for CAD and SPAD, since we have argued that they are at a disadvantage using these diversity metrics. If we look at the relative performances of CAD and SPAD, we can see that CAD achieves higher diversity values in more cases than SPAD.

The results for the experiments that use pLSA as the baseline algorithm are in Tables 6.4 and 6.5. Consider precision and $\alpha$-nDCG first. Comparing Tables 6.4 and 6.2, we see that pLSA has lower precision and $\alpha$-nDCG than MF on all four

Table 6.4: Results using pLSA as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| pLSA | | 0.2639 | 0.2842 | | |
| xQuAD | 0.7 | 0.2456 | **0.3428** | -6.93 | 20.61 |
| SPAD | 1.0 | **0.2803** ▲△ | 0.3171 | 6.2 | 11.57 |
| CAD | 0.7 | 0.2731 ▲ | 0.316 | 3.48 | 11.19 |
| LFM | | | | | |
| pLSA | | 0.3804 | 0.3426 | | |
| xQuAD | 0.5 | 0.41 | 0.3847 | 7.78 | 12.28 |
| SPAD | 0.6 | **0.4299** ▲△ | **0.3878** ▲△ | 13.0 | 13.19 |
| CAD | 0.5 | 0.4124 | 0.3738 | 8.39 | 9.09 |
| LT | | | | | |
| pLSA | | 0.0965 | 0.1376 | | |
| xQuAD | 0.8 | 0.1233 | 0.1816 | 27.76 | 31.94 |
| SPAD | 0.6 | 0.1407 ▲ | 0.1937 ▲ | 45.8 | 40.75 |
| CAD | 0.7 | **0.1423** ▲ | **0.1983** ▲△ | 47.36 | 44.1 |
| FB | | | | | |
| pLSA | | 0.1028 | 0.1217 | | |
| xQuAD | 0.6 | 0.0996 | 0.148 | -3.19 | 21.65 |
| SPAD | 0.7 | 0.1201 ▲ | 0.1445 | 16.78 | 18.76 |
| CAD | 0.8 | **0.1209** ▲△ | **0.1501** ▲△ | 17.6 | 23.36 |

datasets, and in no case does re-ranking pLSA give a higher results than the *corresponding* result for MF. Otherwise, the story is very similar. Here, SPAD has highest precision for the ML1M and LFM datasets, and CAD the second highest. For the LT and FB datasets, CAD has the highest precision, and SPAD the second highest. However, for the LT dataset the difference between CAD and SPAD is not statistically significant. Again, despite making no use of explicit features, CAD and SPAD always increase diversity and, in fact, now for the LFM dataset SPAD gives the highest $\alpha$-nDCG and for the LT and FB datasets CAD gives the highest $\alpha$-nDCG. Only in the ML1M dataset does xQuAD achieve the highest diversity, again at the expense of a decrease in accuracy.

Next, consider the other diversity metrics, which are in Table 6.5. For ERR-IA and S-recall, for all four datasets, all of the re-ranking approaches increase the value over the pLSA baseline. SPAD has the highest EILD with one exception only. The exception is FB, where CAD has the highest EILD. For ILD, xQuAD has the highest ILD with one exception only. The exception is FB, where CAD has the highest ILD and xQuAD has lower ILD than the baseline, CAD and

Table 6.5: Diversity metrics except $\alpha$-nDCG using pLSA baseline.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| ML1M | | | | | | | | | |
| pLSA | | 0.1831 | 0.4664 | 0.1864 | 0.7023 | | | | |
| xQuAD | 0.7 | **0.2041** | **0.6589** | 0.1815 | **0.7706** | 11.46 | 41.27 | -2.62 | 9.73 |
| SPAD | 1.0 | 0.2037 | 0.5137 | **0.2055**▲△ | 0.7282 | 11.26 | 10.15 | 10.29 | 3.69 |
| CAD | 0.7 | 0.2026 | 0.52△ | 0.2026▲ | 0.7449△ | 10.66 | 11.49 | 8.74 | 6.07 |
| LFM | | | | | | | | | |
| pLSA | | 0.1629 | 6.14E-4 | 0.2863 | 0.7424 | | | | |
| xQuAD | 0.5 | 0.1829 | **6.85E-4** | 0.3296 | **0.7918** | 12.33 | 11.47 | 15.14 | 6.65 |
| SPAD | 0.6 | **0.184**△ | 6.36E-4 | **0.3328**△ | 0.7601 | 12.95 | 3.46 | 16.26 | 2.38 |
| CAD | 0.5 | 0.1799 | 6.37E-4 | 0.3168 | 0.7614 | 10.46 | 3.71 | 10.67 | 2.57 |
| LT | | | | | | | | | |
| pLSA | | 0.0672 | 0.0085 | 0.0478 | 0.698 | | | | |
| xQuAD | 0.8 | 0.088 | **0.0106** | 0.074 | **0.7955** | 30.95 | 25.11 | 54.72 | 13.96 |
| SPAD | 0.6 | 0.0912▲ | 0.0094△ | 0.0851▲ | 0.7494△ | 35.74 | 10.81 | 77.94 | 7.36 |
| CAD | 0.7 | **0.0942**▲△ | 0.0092 | **0.0854**▲ | 0.741 | 40.12 | 8.43 | 78.59 | 6.15 |
| FB | | | | | | | | | |
| pLSA | | 0.0136 | 0.0048 | 0.0822 | 0.9686 | | | | |
| xQuAD | 0.7 | **0.0186** | **0.007** | 0.0699 | 0.962 | 36.01 | 45.17 | -14.9 | -0.68 |
| SPAD | 0.7 | 0.0165 | 0.005 | 0.0999▲ | 0.9712▲ | 21.11 | 4.01 | 21.44 | 0.27 |
| CAD | 0.8 | 0.0175△ | 0.0051△ | **0.1012**▲△ | **0.9727**▲△ | 28.3 | 6.2 | 23.1 | 0.43 |

SPAD. It is noteworthy that, unlike xQuAD, CAD and SPAD improve all the diversity metrics for all four datasets. Again, similar to the MF baseline, for the pLSA baseline CAD has higher diversity in more cases than SPAD.

The results for the experiments that use FMBPR as the baseline algorithm are in Tables 6.6 and 6.7. For precision and $\alpha$-nDCG, which are shown in Table 6.6, the results are very similar to those for the pLSA baseline shown in Table 6.4. SPAD has the highest precision for all the datasets, with one exception only. The exception is FB, where CAD has the highest precision. Again, despite being at a disadvantage in terms of diversity, CAD and SPAD always increase $\alpha$-nDCG over the baseline algorithm and, in fact, for LFM, SPAD gives the highest $\alpha$-nDCG and for the LT and FB datasets CAD gives the highest $\alpha$-nDCG. Only in the ML1M dataset, at the expense of a decrease in precision, xQuAD achieves the highest $\alpha$-nDCG. If we look at the relative performances of CAD and SPAD, arguably, SPAD gives better improvement over the FMBPR baseline for both precision and $\alpha$-nDCG for all datasets but FB.

For the other diversity metrics, see Table 6.7. All of the re-ranking algorithms improve ERR-IA over the baseline. SPAD has the highest ERR-IA for all datasets with one exception only. The exception is FB, where xQuAD has the highest ERR-IA. xQuAD has the highest S-recall for all datasets. SPAD has the highest EILD for all datasets with the exception of FB, where CAD has the highest EILD. For ILD results are mixed. For the relative performances of CAD and SPAD, similar to those for the MF and pLSA baselines, CAD has higher diversity

Table 6.6: Results using FMBPR as the baseline.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision | $\alpha$-nDCG | Precision | $\alpha$-nDCG |
| ML1M | | | | | |
| FMBPR | | 0.2655 | 0.3025 | | |
| xQuAD | 0.4 | 0.2534 | **0.3376** | -4.56 | 11.61 |
| SPAD | 0.4 | **0.2829**▲△ | 0.3219△ | 6.58 | 6.43 |
| CAD | 0.4 | 0.2715▲ | 0.3205 | 2.27 | 5.96 |
| LFM | | | | | |
| FMBPR | | 0.3737 | 0.3409 | | |
| xQuAD | 0.4 | 0.3972 | 0.3758 | 6.28 | 10.23 |
| SPAD | 0.4 | **0.4231**▲△ | **0.3864**▲△ | 13.21 | 13.35 |
| CAD | 0.5 | 0.397 | 0.3617 | 6.24 | 6.12 |
| LT | | | | | |
| FMBPR | | 0.0829 | 0.112 | | |
| xQuAD | 0.5 | 0.1161 | 0.1677 | 40.04 | 49.71 |
| SPAD | 0.5 | **0.1275**▲△ | 0.1696▲ | 53.77 | 51.38 |
| CAD | 0.6 | 0.1239 ▲ | **0.1726**▲△ | 49.47 | 54.01 |
| FB | | | | | |
| FMBPR | | 0.107 | 0.1213 | | |
| xQuAD | 0.4 | 0.1092 | 0.1512 | 2.04 | 24.63 |
| SPAD | 0.4 | 0.1253▲ | 0.1472 | 17.12 | 21.34 |
| CAD | 0.4 | **0.1288**▲△ | **0.1547**▲△ | 21.75 | 28.65 |

values in more cases than SPAD.

### 6.2.3 Results for different values of $\lambda$

In this section, we look at the effect of parameter $\lambda$, which controls the balance between relevance and diversity in Eq. 2.2. The results we have shown so far use whichever values for $\lambda$ give highest $\alpha$-nDCG. Instead, here we plot precision and $\alpha$-nDCG on the test set for different values of $\lambda$; see Figures 6.1 and 6.2.

In this section and in the following sections of this chapter, we only show the results for the MF baseline, since all of the re-ranking approaches achieve both their best precision and $\alpha$-nDCG values using MF as the baseline recommender.

We see that the results that we discussed in Section 6.2.2 are fairly robust over different values for $\lambda$. For example, for the ML1M dataset (Figure 6.1a), SPAD's precision is comparable with, or higher than, CAD's and xQuAD's, only becom-

Table 6.7: Diversity metrics except $\alpha$-nDCG using FMBPR baseline.

| | | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | ERR-IA | S-recall | EILD | ILD | ERR-IA | S-recall | EILD | ILD |
| ML1M | | | | | | | | | |
| FMBPR | | 0.1982 | 0.5119 | 0.1854 | 0.7166 | | | | |
| xQuAD | 0.4 | 0.206 | **0.6215** | 0.1797 | 0.7404 | 3.92 | 21.41 | -3.05 | 3.36 |
| SPAD | 0.4 | **0.2107**▲△ | 0.5208 | **0.2055**▲△ | 0.7337 | 6.29 | 1.73 | 10.84 | 2.39 |
| CAD | 0.4 | 0.2081▲ | 0.5337△ | 0.1981▲ | **0.7462**▲△ | 4.99 | 4.26 | 6.87 | 4.12 |
| LFM | | | | | | | | | |
| FMBPR | | 0.1614 | 6.48E-4 | 0.2868 | 0.7724 | | | | |
| xQuAD | 0.4 | 0.1742 | **6.95E-4** | 0.3216 | **0.8016** | 7.94 | 7.18 | 12.15 | 3.78 |
| SPAD | 0.4 | **0.1835**▲△ | 6.51E-4△ | **0.3314**▲△ | 0.7736△ | 13.66 | 0.42 | 15.56 | 0.15 |
| CAD | 0.5 | 0.1756 | 6.42E-4 | 0.3055 | 0.7672 | 8.77 | -0.9 | 6.52 | -0.68 |
| LT | | | | | | | | | |
| FMBPR | | 0.0546 | 0.009 | 0.0394 | 0.7179 | | | | |
| xQuAD | 0.5 | 0.0801 | **0.0107** | 0.0683 | **0.7938** | 46.72 | 18.52 | 73.55 | 10.57 |
| SPAD | 0.5 | 0.0803 | 0.0094△ | **0.0742**▲△ | 0.7401△ | 46.98 | 3.98 | 88.6 | 3.1 |
| CAD | 0.6 | **0.0838**▲△ | 0.0091 | 0.07 | 0.7289 | 53.48 | 0.75 | 77.89 | 1.54 |
| FB | | | | | | | | | |
| FMBPR | | 0.0134 | 0.0053 | 0.0863 | **0.9722** | | | | |
| xQuAD | 0.4 | **0.019** | **0.0067** | *0.0857* | 0.9637 | 42.19 | 26.21 | -0.74 | -0.87 |
| SPAD | 0.4 | 0.0168 | 0.0052 | 0.1058▲ | 0.9699▲ | 25.41 | -0.97 | 22.57 | -0.23 |
| CAD | 0.4 | 0.0182▲△ | *0.0052* | **0.1126**▲△ | *0.9719*▲△ | 36.29 | -1.16 | 30.47 | -0.04 |

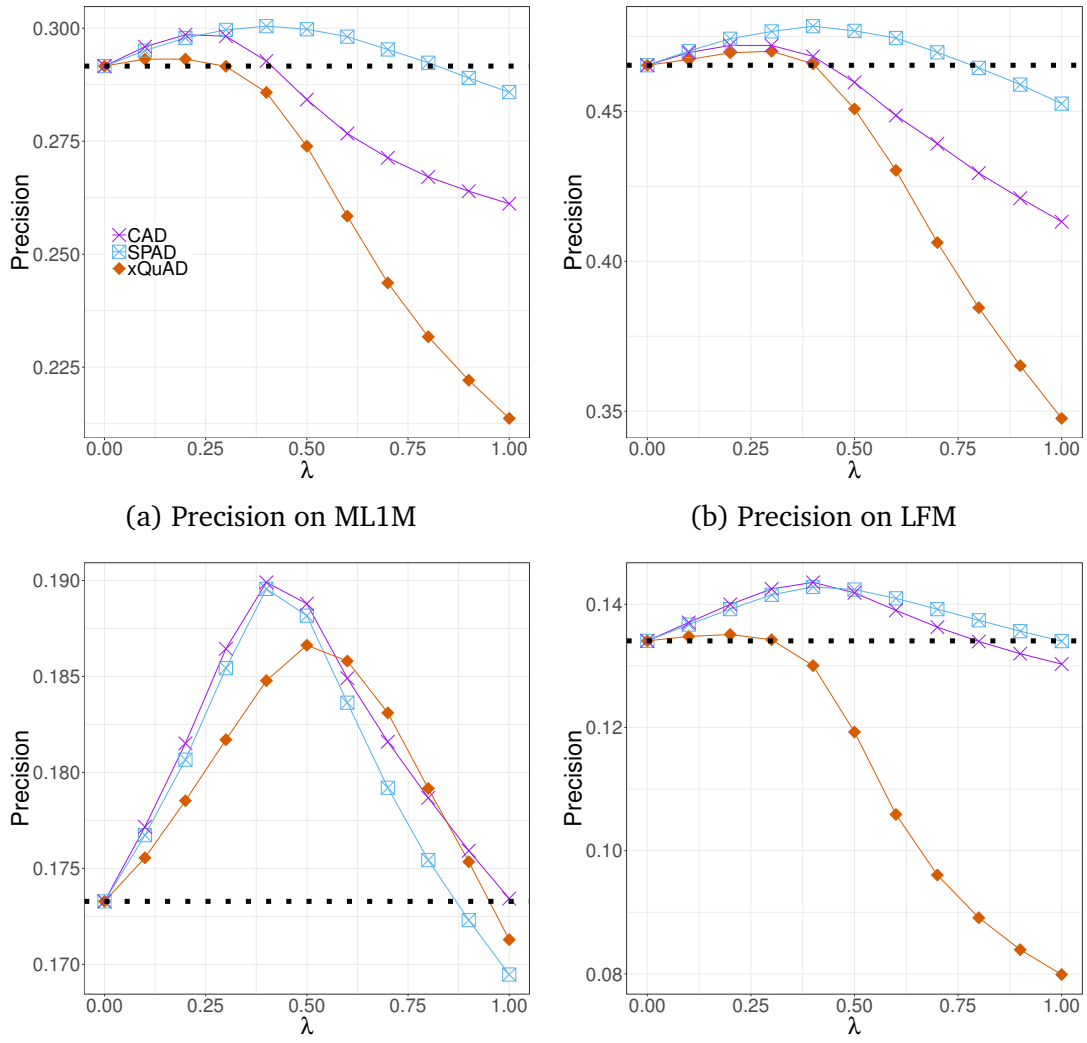
ing worse than the baseline for very high values of $\lambda$. For $\alpha$-nDCG (Figure 6.2a), SPAD is competitive with CAD for low values of $\lambda$ and then outperforms CAD. xQuAD has high $\alpha$-nDCG for many values of $\lambda$ but soon suffers from decreases in precision. Results for LFM (Figures 6.1a and 6.2a) are similar but less marked.

For LT (Figures 6.1c and 6.2a), SPAD and CAD switch places and so it is CAD that does best across different values of $\lambda$, and for higher values of $\lambda$, xQuAD is competitive with CAD since, for these values, it has higher $\alpha$-nDCG and competitive precision. For FB (Figures 6.1d and 6.2d), xQuAD is not competitive for any values of $\lambda$: even where its $\alpha$-nDCG is a little higher, this comes at the cost of large drops in precision. Across different values of $\lambda$, SPAD and CAD perform quite closely on this dataset, although we know from Section 6.2.2 that, at $\lambda = 0.4$, CAD outperforms SPAD on both metrics.

### 6.2.4 Diversity measured by subprofiles

Subprofile-Aware Diversification methods make no use of explicit item features. Therefore, they are at a disadvantage in experiments where we measure diversity using existing metrics, since all those metrics are defined in terms of explicit item features. CAD is as much at a disadvantage as SPAD in such experiments.

In the previous chapter, we adapted the diversity metrics to create ones that are




(a) Precision on ML1M

(b) Precision on LFM

(c) Precision on LT

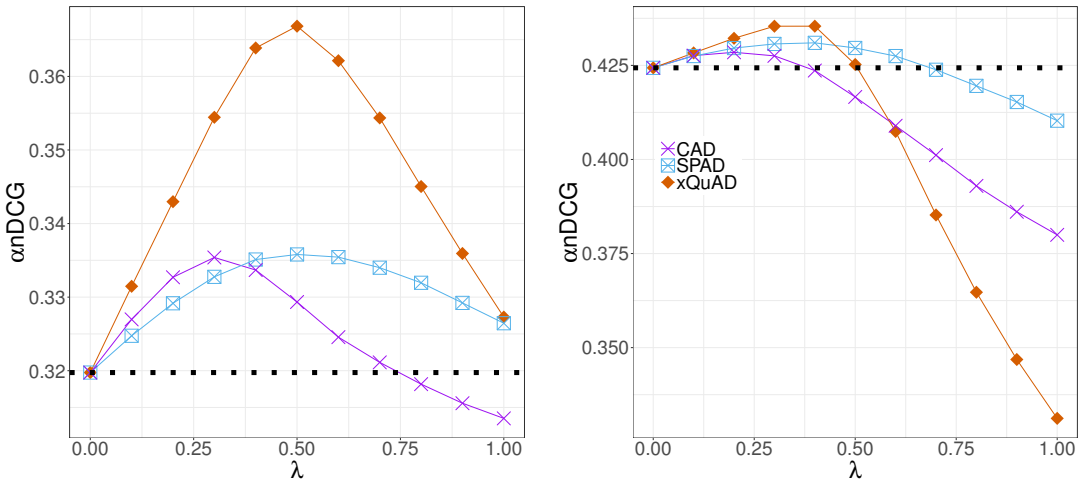
(d) Precision on FB

Figure 6.1: ML1M, LFM, LT and FB datasets. Precision values for varying $\lambda$ using MF as baseline. Values for MF are shown by dotted lines.

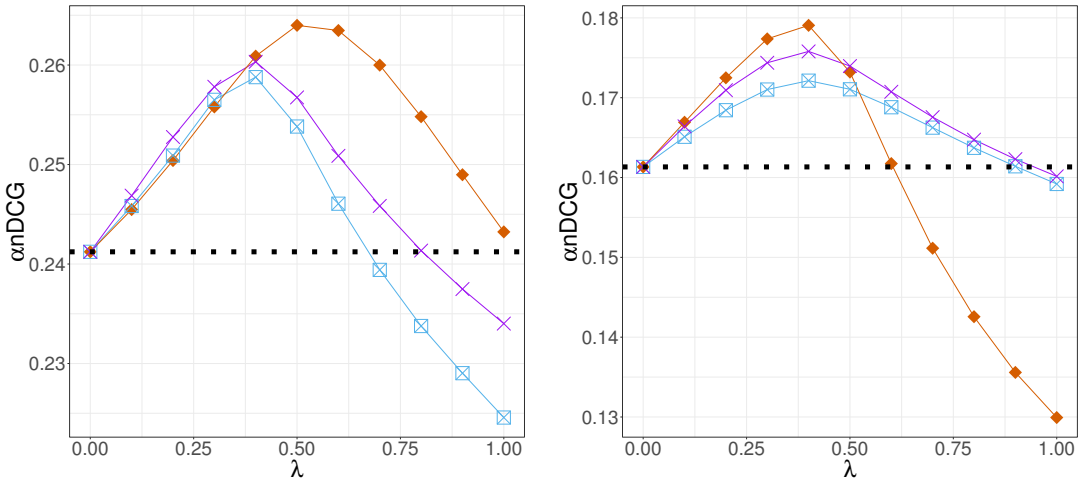
defined in terms of subprofiles rather than item features (see Section 5.2). In this section, we give a more balanced picture of the performances of the diversification algorithms, including CAD, by showing both sets of diversity metrics side by side.

Similarly to Section 5.2.2, we use MF as the baseline recommender and we use the ML1M dataset for the analysis in this section. The hyperparameters we use for CAD are the ones that optimize $\alpha$-nDCG on the validation set for the MF baseline recommender; see section A.3 in the Appendix.

The subprofiles we use when computing the adapted metrics are the ones detected by the subprofile detection part of CAD (see Section 6.1). Note that




(a) $\alpha$-nDCG on ML1M (b) $\alpha$-nDCG on LFM

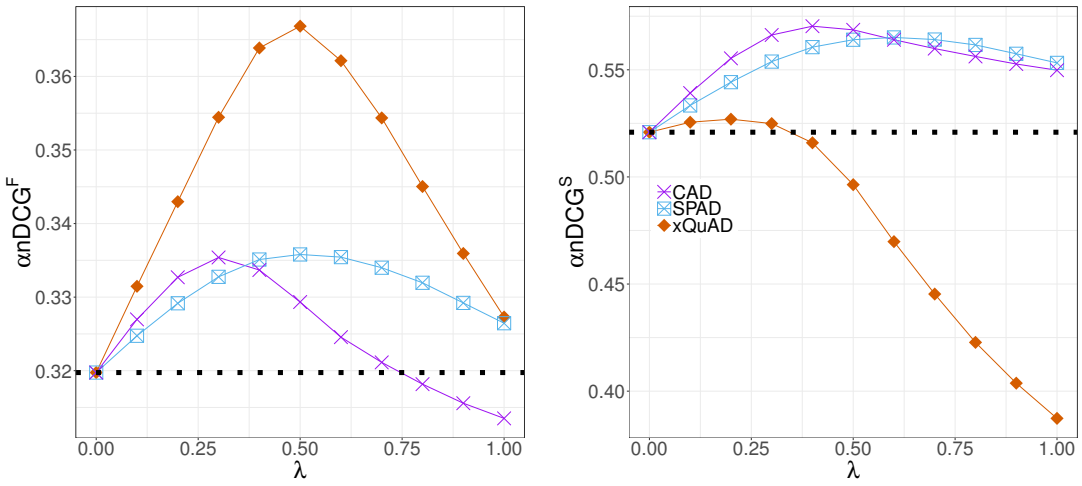



(c) $\alpha$-nDCG on LT (d) $\alpha$-nDCG on FB

Figure 6.2: ML1M, LFM, LT and FB datasets. $\alpha$-nDCG values for varying $\lambda$ using MF as baseline. Values for MF are shown by dotted lines.

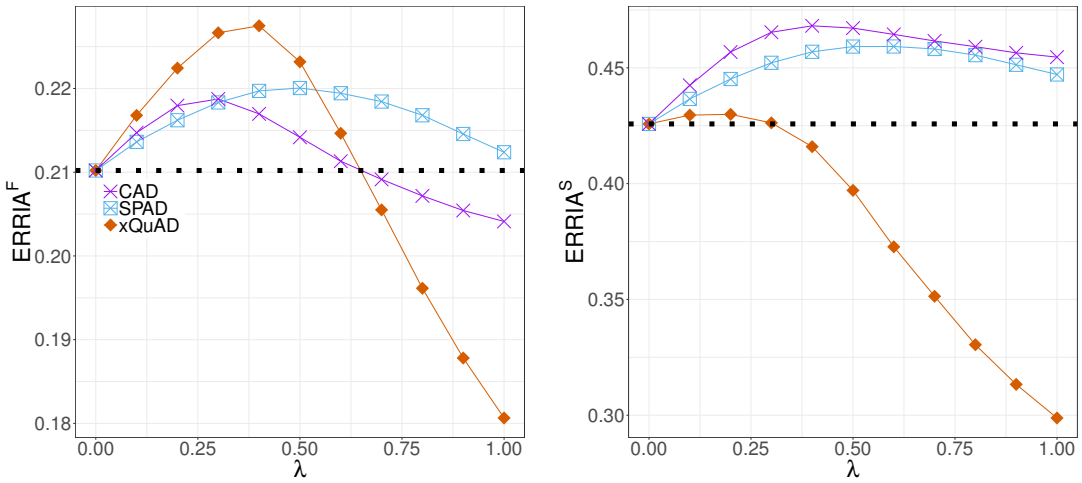
SPAD is at a disadvantage when we measure subprofile-based diversity metrics by using subprofiles detected by CAD. We did plot adapted diversity metrics by using subprofiles of SPAD as well. We find that the results are really similar to the ones that we show in this section by using CAD's subprofiles. The main difference is the one we would expect: CAD mostly has better results when the metric uses CAD subprofiles; SPAD mostly has better results when the metric uses SPAD subprofiles. Since this difference is to be expected, for simplicity and clarity, we do not show the results where the adapted metrics uses SPAD's subprofiles.

Figures 6.3, 6.4, 6.5, 6.6 and 6.7 show $\alpha$-nDCG, ERR-IA, S-recall, EILD and ILD measured by item features and measured by subprofiles. We plot the met-




(a) $\alpha$-nDCG$^{\mathcal{F}}$ on ML1M (b) $\alpha$-nDCG$^{\mathcal{S}}$ on ML1M

Figure 6.3: ML1M dataset. $\alpha$-nDCG measured using features and subprofiles for varying $\lambda$.

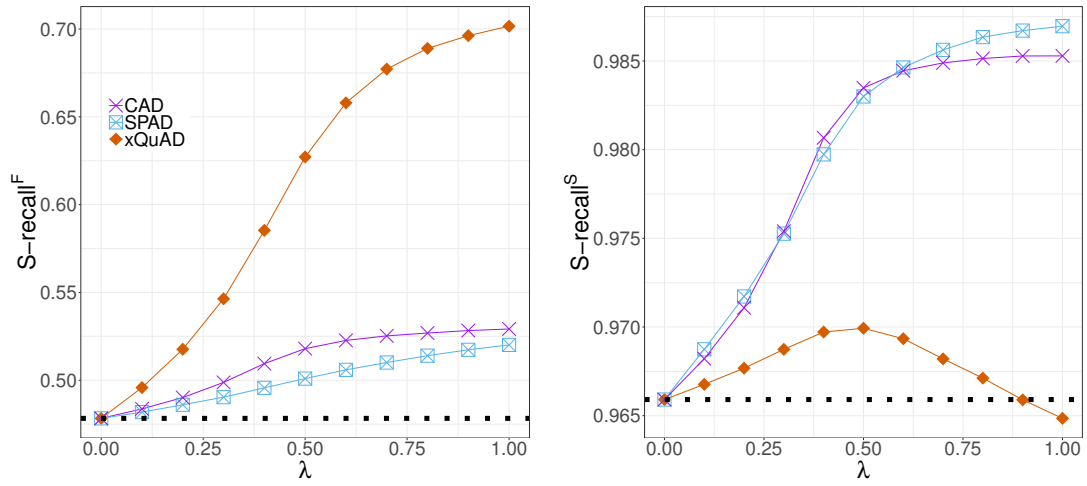



(a) ERR-IA$^{\mathcal{F}}$ on ML1M (b) ERR-IA$^{\mathcal{S}}$ on ML1M

Figure 6.4: ML1M dataset. ERR-IA measured using features and subprofiles for varying $\lambda$.

rics computed on the test set for different values of $\lambda$.

For the $\alpha$-nDCG metric, see Figure 6.3. For $\alpha$-nDCG$^{\mathcal{F}}$ (Figure 6.3a), xQuAD and SPAD have higher values than the baseline for all values of $\lambda$. But, as expected, xQuAD has higher $\alpha$-nDCG$^{\mathcal{F}}$ than SPAD and CAD for all values of $\lambda$. For larger values of $\lambda$, CAD suffers from decreases in $\alpha$-nDCG$^{\mathcal{F}}$. For $\alpha$-nDCG$^{\mathcal{S}}$ on the other hand (Figure 6.3b), as expected CAD and SPAD perform better than xQuAD and the baseline for all values of $\lambda$. xQuAD suffer from decreases in $\alpha$-nDCG$^{\mathcal{S}}$ for values of $\lambda$ from about 0.4.




(a) S-recall$^{\mathcal{F}}$ on ML1M (b) S-recall$^{\mathcal{S}}$ on ML1M

Figure 6.5: ML1M dataset. S-recall measured using features and subprofiles for varying $\lambda$.

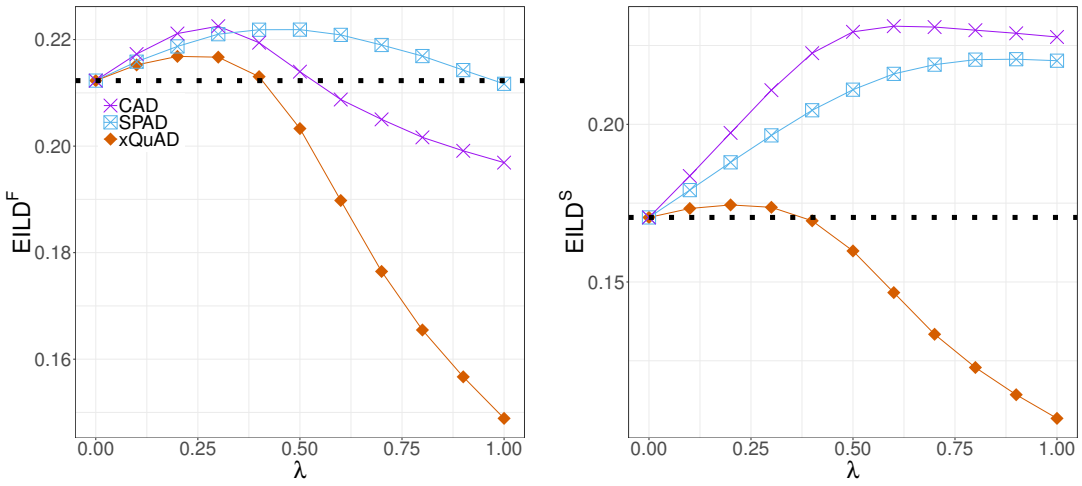
For the ERR-IA metric, see Figure 6.4. The results are quite similar to those for $\alpha$-nDCG (Figure 6.3). What differs is that, for ERR-IA$^{\mathcal{F}}$ (Figure 6.4a), xQuAD starts to suffer from decreases in ERR-IA$^{\mathcal{F}}$ for values of $\lambda$ from about 0.7.

The results for S-recall are shown in Figure 6.5. Consider S-recall$^{\mathcal{F}}$ (Figure 6.5$a$) first. All of the algorithms increase the values of S-recall$^{\mathcal{F}}$ for all values of $\lambda$. xQuAD, as expected, performs better than both CAD and SPAD, and CAD performs better than SPAD for all values of $\lambda$. Next, consider S-recall$^{\mathcal{S}}$ (Figure 6.5$b$). CAD and SPAD increase the value of this metric over the baseline for all values of $\lambda$. xQuAD performs better than the baseline for all but large values of $\lambda$, but does not perform as well as CAD and SPAD.

Figure 6.6 shows the results for EILD. First, consider Figure 6.6a, which shows EILD$^{\mathcal{F}}$. Surprisingly, SPAD and CAD, for all values of $\lambda$, perform better than xQuAD. Next, consider EILD$^{\mathcal{S}}$ (Figure 6.6a). For all values of $\lambda$, CAD and SPAD, as expected, perform better than xQuAD and the baseline. xQuAD, on the other hand, soon suffers from decreases in EILD$^{\mathcal{S}}$.

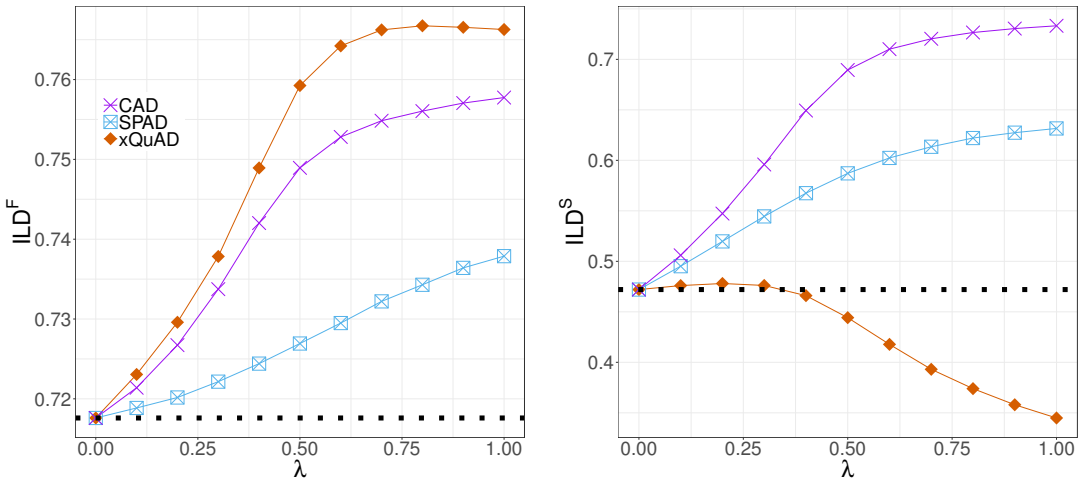
The results for ILD are shown in Figure 6.7; ILD$^{\mathcal{F}}$ in Figure 6.7a, and ILD$^{\mathcal{S}}$ in Figure 6.7b. All of the algorithms increase ILD$^{\mathcal{F}}$ over the baseline for all values of $\lambda$. When we consider ILD$^{\mathcal{S}}$, the results are very similar to those for EILD$^{\mathcal{S}}$ (see Figure 6.6b). CAD and SPAD increase ILD$^{\mathcal{S}}$ over the baseline for all values of $\lambda$, and they perform better than xQuAD.

The results in this section have confirmed that, similar to SPAD, CAD is at a




(a) EILD$^{\mathcal{F}}$ on ML1M (b) EILD$^{\mathcal{S}}$ on ML1M

Figure 6.6: ML1M dataset. EILD measured using features and subprofiles for varying $\lambda$.

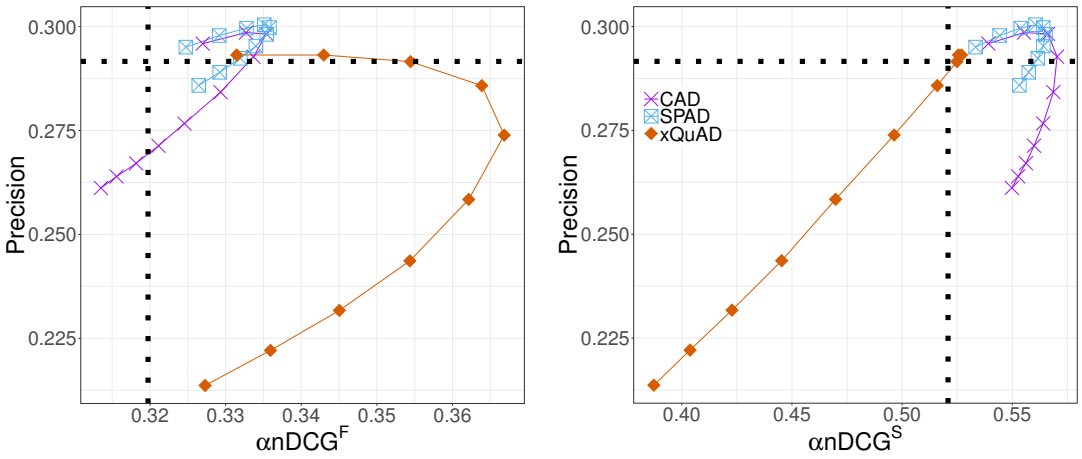



(a) ILD$^{\mathcal{F}}$ on ML1M (b) ILD$^{\mathcal{S}}$ on ML1M

Figure 6.7: ML1M dataset. ILD measured using features and subprofiles for varying $\lambda$.

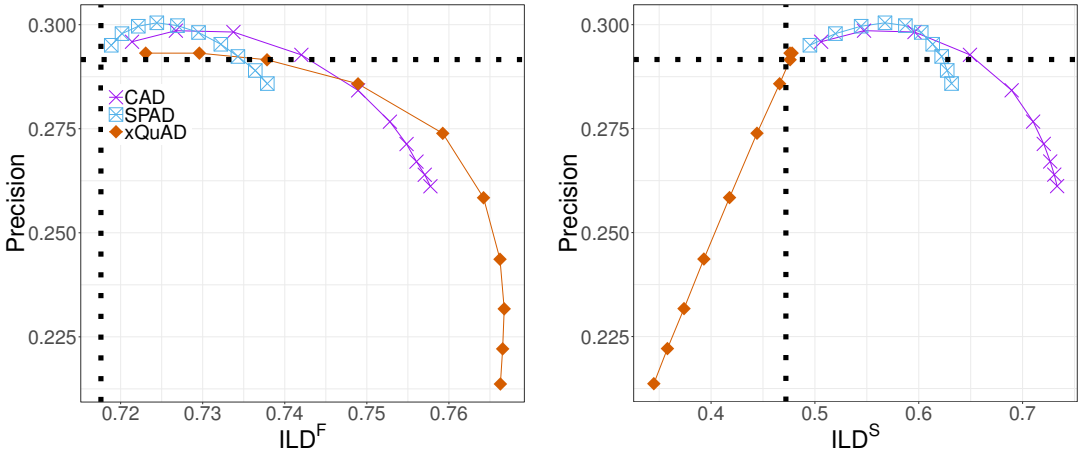
disadvantage in experiments where diversity metrics use item features. They also show that CAD and SPAD increase the values of the diversity metrics over the baseline when measured by subprofiles for all five diversity metrics and for all values of $\lambda$, and they always perform better than xQuAD. This is as expected, since CAD and SPAD use subprofiles for re-ranking; in this case, it is xQuAD that is at a disadvantage, since it makes no use of subprofiles.

The results showing diversity metrics measured by item features are noteworthy, since, even though CAD and SPAD make no use of item features, there are




(a) Precision/$\alpha$-nDCG$^{\mathcal{F}}$ on ML1M (b) Precision/$\alpha$-nDCG$^{\mathcal{S}}$ on ML1M

Figure 6.8: ML1M dataset. Precision vs. $\alpha$-nDCG trade-off plots.




(a) Precision/ILD$^{\mathcal{F}}$ on ML1M (b) Precision/ILD$^{\mathcal{S}}$ on ML1M

Figure 6.9: ML1M dataset. Precision vs. ILD trade-off plots.

configurations where they perform better than xQuAD. This supports our claim in Section 5.2.2 that using subprofile-aware approaches to recommendation diversification allows a finer-grained representation of tastes and interests and this additional complexity allows them to perform better than other re-ranking algorithms.

### 6.2.5 A visualization of trade-offs

In this section, similar to Section 5.3 of the previous chapter, we show the trade-off between relevance and diversity for the re-ranking methods CAD, SPAD and xQuAD. Figure 6.8 and 6.9 plots their precision against diversity, where diver-

sity is measured by two versions of $\alpha$-nDCG and ILD respectively. The dotted lines show the precision and diversity of the MF baseline, dividing each subfigure into four quadrants. Top-right parts of the plots means improved precision and diversity are over the baseline.

Consider first Figure 6.8. We can see that, for both $\alpha\text{-nDCG}^{\mathcal{F}}$ and $\alpha\text{-nDCG}^{\mathcal{S}}$, CAD and SPAD more often improve precision and diversity over the baseline, because they are less likely to trade off precision for diversity. If we look at the relative performances of CAD and SPAD, we can see that SPAD more often improves precision and diversity compared to CAD. Next, consider Figure 6.9, which shows precision against diversity using two versions of ILD, $\text{ILD}^{\mathcal{F}}$ and $\text{ILD}^{\mathcal{S}}$. The results are similar to those in Figure 6.8, with CAD and SPAD more likely to increase both precision and diversity. For both versions of ILD, we can see that CAD improves diversity over baseline more than SPAD, but it does so sometimes by trading off precision.

## 6.3 Conclusions

In this chapter, we have presented a new approach to recommendation set diversification, which we call Community-Aware Diversification (CAD). It is an intent-aware approach and uses subprofiles of the items that the user likes as its aspects, as does Subprofile-Aware Diversification (SPAD). CAD detects subprofiles using a user-user similarity approach, unlike SPAD, which uses an item-item similarity approach. We compared the performance of CAD to SPAD and to xQuAD on four datasets. We found that CAD and SPAD produce recommendations that are always the most accurate. We also showed that CAD and SPAD are less prone to trading-offs accuracy for diversity. In some cases using CAD subprofiles performs better than using SPAD subprofiles; in other cases, the reverse is the case. We conclude that, for some domains, it is meaningful and useful to define subprofiles indirectly through a user's community (her nearest neighbours).

In the next chapter, we compare intent-aware diversification approaches to calibrated recommendation.

# Chapter 7

# A Comparison of Calibrated and Intent-Aware Recommendations

Calibrated and intent-aware recommendation are recent approaches that have apparent similarities. Both try, to a certain extent, to cover the user's tastes and interests, as revealed by her user profile. The main difference lies in their objective. Aside from recommendation relevance, the main goal of a calibrated recommender system is to produce calibrated recommendations, reflecting user's interests in the right proportions. A calibrated recommendation set might be diverse, but diversity is not an explicit goal.[1] By contrast, diversity is the main goal of intent-aware recommender systems. This is achieved by something similar to calibration and so a set of recommendations might be calibrated to some extent. But, intent-aware methods define their equivalent to calibration in a relevance-based way. We gave detailed explanations of both calibrated and intent-aware recommendation in Chapter 2.

In this chapter, we compare them in detail. First, we compare them by giving an informal, motivating example from the music domain. Second, since Steck has shown the optimality of calibrated recommendation [Ste18], we want to confirm the same for intent-aware diversification by showing that the greedy approach to intent-aware diversification used in xQuAD [Var15], and inherited by SPAD and RSPAD, give an approximate optimality guarantee. Third, we define a new variant of Steck's calibrated recommender systems, one which calibrates with respect to subprofiles, rather than item features. Then, we define a new variant of the calibration metric, one that measures calibration in

[1] In fact, Steck defines a diversity-enhanced calibrated recommender system, which includes diversity as an explicit objective, alongside calibration [Ste18]. We will discuss it in Section 7.4.

terms of subprofiles, rather than item features. Finally, we present an empirical comparison of calibrated and intent-aware recommendation.

## 7.1 Intent-Aware vs. Calibrated

As we have mentioned, there is an apparent similarity between intent-aware and calibrated recommendation. Both try to cover the user's different tastes and interests, as revealed by her profile. Indeed, for the latter, covering the tastes and interests in the same proportion as they occur in the user's profile is the main goal. Intent-aware recommendation may result in calibrated recommendations, but it does not directly aim to cover the user's interests in the same proportion as they occur in the user's profile. As we saw, it modulates coverage by recommendation relevance. We illustrate this difference with an informal example.

Consider a user who listens to jazz 70% of the time and to rock music 30% of the time. Suppose that the goal is to recommend a list of top-$10$ recommendations. Calibrated recommendation tries to generate 10 recommendations such that seven (70%) are jazz and the remaining three (30%) are rock. An intent-aware approach, such as xQuAD, considers how much an item satisfies a given aspect (genre in this case) through $p(i|u, f)$. Suppose the first four songs that the recommender includes in the recommendation list are jazz songs which this user is likely to choose from a recommendation list, i.e. $p(i|u, f = jazz)$ is high for each of these songs, $i$. Now consider adding a fifth piece of jazz to the recommendation list. It will be penalized because each of the four existing items $j$ in the recommendation list have high $p(j|u, f = jazz)$: see the factor $\prod_{j \in RL}(1 - p(j|u, f))$ in Eq. 2.8. A fifth jazz song might only be included if it can overcome the 'penalty' imposed by the songs that have been added to the recommendation list already. The final top-$10$ might not have seven jazz songs; it may even have more rock than jazz. On the other hand, if the first seven songs to be added to the recommendation list are (informally speaking) not jazzy enough for this user's tastes (more precisely, if they have very low values for $p(i|u, f = jazz)$), then it is possible that more jazz songs will be added to the recommendation list. The final top-$10$ might not have three rock songs; it may even have no rock at all.

## 7.2 The Optimality of Intent-Aware Diversification

In this section we analyze the greedy re-ranking approach to intent-aware diversification to show that its objective function is monotone and submodular and, therefore, although greedy, it has a $(1 - \frac{1}{e})$ optimality guarantee. Doing this confirms that, on this criterion at least, intent-aware and calibrated recommendation are on a par, since Steck has already shown that calibrated recommendation enjoys this guarantee.

Our analysis closely follows Agrawal and Gollapodi's analysis of IA-Select's search result diversification objective, which is to maximize the probability that the average user finds at least one useful result within the top-$N$ result [AGHI09]. Agrawal and Gollapudi prove the submodularity of their search result diversification objective, IA-SELECT. We can adapt their proof because intent-aware diversification is a generalization of IA-Select [Var15].

First, we give some definitions:

**Definition 7.2.1.** **MONOTONICITY** *Given a finite ground set $I$, a function $f : 2^I \mapsto \mathbb{L}$ from sets drawn from $I$ to reals is monotonic if and only if for all sets $X \subseteq I$, and $i \in I \setminus X$, it satisfies the following: $f(X \cup \{i\}) - f(X) \geq 0$.*

**Definition 7.2.2.** **SUBMODULARITY** *Given a finite ground set $I$, a function $f : 2^I \mapsto \mathbb{L}$ from sets drawn from $I$ to reals is submodular if and only if for all sets $X \subseteq Y \subseteq I$, and $i \in I \setminus Y$, it satisfies the following: $f(X \cup \{i\}) - f(X) \geq f(Y \cup \{i\}) - f(Y)$.*

The inequality in the definition of submodularity means that adding an item $i$ to a smaller set $X$ brings a gain that is no smalller than adding it to a larger set $Y$.

**Definition 7.2.3.** **MODULAR** *Given a finite ground set $I$, a function $f : 2^I \mapsto \mathbb{L}$ from sets drawn from $I$ to reals is modular if $f(X) = \sum_{i \in X} w_i$ for some weights $w : I \mapsto \mathbb{L}$. Such functions are also referred to as* additive *or* linear. *Modular functions are also submodular. If $w_i \geq 0$ for all $w_i$, then $f$ is also monotone.*

In intent-aware approaches to diversification, such as xQuAD and SPAD, the goal, for a given user $u$, is to determine the optimal recommendation set, denoted as $RL^*$, of size $N$ items, such that $RL^*$ maximizes the following objective function:

$$RL^* = \underset{RL, |RL|=N}{\arg\max} (1 - \lambda) s(RL) + \lambda \operatorname{div}_{\text{IA}}(RL) \tag{7.1}$$

where $s(RL) = \sum_{i \in RL} s(u, i)$, is a modular function.

When $\lambda = 1$ the objective function in 7.1 reduces to finding the optimal set of items that maximizes the following function:

$$\mathrm{div}_{\mathrm{IA}}(RL) = \sum_{a \in \mathcal{A}} p(a|u) \left(1 - \prod_{i \in RL} (1 - p(i|u,a))\right) \tag{7.2}$$

which is NP-hard to optimize.

**Lemma 7.2.4.** *Finding the optimal set that maximizes* $\mathrm{div}_{\mathrm{IA}}(RL)$ *is NP-hard.*

*Proof.* This follows from a reduction from Max $k$-Coverage, an NP-hard problem related to SET COVER [Hoc97]. In the Max k-Coverage problem, one is given a universe of elements $\mathcal{E}$, a collection $\mathcal{C}$ of subsets of $\mathcal{E}$, and an integer $k$. Each element $e \in \mathcal{E}$ has a corresponding weight $w(e)$. The objective is to find a set of subsets $E = \{E_1, E_2, \ldots, E_k\} \subseteq \mathcal{C}, |E| = k$, such that the sum of the weights of $\bigcup_{l=1}^{k} E_l$ is maximized.

We create a mapping between $\mathcal{E}$ and a set of aspects $\mathcal{A}$. Then, the collection of subsets $\mathcal{C}$ is mapped to the set of items $I$, since each item $i$ in our case is represented as set of aspects $i = \{a_1, a_2, \ldots, a_m\}$. The weight of an aspect $a$, $w(a)$ (analogous to $w(e)$) becomes:

$$w(a) = p(a|u) \left(1 - \prod_{i \in RL} (1 - p(i|u,a))\right)$$

Since we try to maximize $\mathrm{div}_{\mathrm{IA}}(RL)$, which can be rewritten as $\mathrm{div}_{\mathrm{IA}}(RL) = \sum_a w(a)$, the optimal solution that maximizes $\mathrm{div}_{\mathrm{IA}}(RL)$ is optimal for Max $k$-Coverage.

□

**Lemma 7.2.5.** $\mathrm{div}_{\mathrm{IA}}(RL)$ *is a submodular function.*

*Proof.* Let $X, Y$ be two arbitrary sets of items related by $X \subseteq Y$. Let $i$ be an item not in $Y$. Let us denote $X \cup \{i\}$ by $X'$ and similarly $Y \cup \{i\}$ as $Y'$.

$$\mathrm{div}_{\mathrm{IA}}(X') - \mathrm{div}_{\mathrm{IA}}(X) = \tag{7.3}$$

$$= \sum_{a \in \mathcal{A}} p(a|u) \left( (1 - \prod_{j \in X'} (1 - p(j|u,a))) - (1 - \prod_{j \in X} (1 - p(j|u,a))) \right) \tag{7.4}$$

$$= \sum_{a \in \mathcal{A}} p(a|u) \left( \prod_{j \in X} (1 - p(j|u,a)) - \prod_{j \in X'} (1 - p(j|u,a)) \right) \tag{7.5}$$

$$= \sum_{a \in \mathcal{A}} p(a|u) \left( \prod_{j \in X} (1 - p(j|u,a)) \right) p(i|u,a) \tag{7.6}$$

Similarly, we can establish that:

$$\mathrm{div}_{\mathrm{IA}}(Y') - \mathrm{div}_{\mathrm{IA}}(Y) = \sum_{a \in \mathcal{A}} p(a|u) \left( \prod_{j \in Y} (1 - p(j|u,a)) \right) p(i|u,a) \tag{7.7}$$

Note that for all $a \in \mathcal{A}$ given that $X \subseteq Y$,

$$\prod_{j \in Y} (1 - p(j|u,a)) \leq \prod_{j \in X} (1 - p(j|u,a))$$

Therefore, we conclude that

$$\mathrm{div}_{\mathrm{IA}}(X') - \mathrm{div}_{\mathrm{IA}}(X) \geq \mathrm{div}_{\mathrm{IA}}(Y') - \mathrm{div}_{\mathrm{IA}}(Y)$$

as desired, i.e. the function $\mathrm{div}_{\mathrm{IA}}(RL)$ is submodular. □

**Lemma 7.2.6.** $\mathrm{div}_{\mathrm{IA}}(RL)$ *is a monotonic function.*

*Proof.* Let $X$ be an arbitrary set of items such that $X \subseteq I$. Let $i$ be an item not in $X$. Let us denote $X \cup \{i\}$ by $X'$. Following Eq. 7.3, we know the following:

$$\mathrm{div}_{\mathrm{IA}}(X') - \mathrm{div}_{\mathrm{IA}}(X) = \sum_{a \in \mathcal{A}} p(a|u) \left( \prod_{j \in X} (1 - p(j|u,a)) \right) p(i|u,a)$$

Knowing that probabilities $p(a|u)$ and $p(i|u,a)$ cannot be negative, we conclude that:

$$\mathrm{div}_{\mathrm{IA}}(X') - \mathrm{div}_{\mathrm{IA}}(X) \geq 0$$

as desired, i.e., the function $\mathrm{div}_{\mathrm{IA}}(RL)$ is monotonic.

□

Now we appeal to the following theorem [NWF78]:

**Theorem 7.2.7.** *For a monotone and submodular function $f$ from sets to reals, where $f(\emptyset) = 0$, let $S^*$ be the optimal set of $k$ elements that maximizes $f$. Let $S'$ be the $k$-element set constructed by greedily selecting elements one at a time that gives the largest marginal increase in $f$. Then $f(S') \geq (1 - \frac{1}{e}) f(S^*)$*

We already mentioned that, intent-aware diversification approaches use greedy re-ranking. We have given the proof of submodularity and monotonocity of $\mathrm{div}_{\mathrm{IA}}(RL)$, which is the second term of Eq. 7.1. We have also explained that the first term of Eq. 7.1, $s(RL)$, is modular (and hence also submodular).

This leads to the submodularity of the objective function given in Eq. 7.1, because it is the sum of a modular function (which is also submodular), $s(RL) = \sum_{i \in RL} s(u,i)$, and a submodular function $\mathrm{div}_{\mathrm{IA}}(RL)$. Since linear combinations of submodular functions are also submodular, the objective function given in Eq. 7.1 is submodular.

We can conclude that the greedy approach to intent-aware diversification approaches is a $(1 - \frac{1}{e})$ approximation of the optimal solution, if $s(RL)$ (Eq. 7.1) is monotonic, i.e. $s(u,i) \geq 0$ for all $(u,i)$ pairs. Note that, when $s(RL)$ is not monotonic (if there exist some $(u,i)$ pairs for which $s(u,i) < 0$), one can easily use a monotone transform, i.e. map all $s(u,i) < 0$ to 0.

Note that the proof given in this section is a general proof for the intent-aware approaches using $\mathrm{div}_{\mathrm{IA}}(RL)$ given in Eq. 7.2: it applies to xQuAD, SPAD and

their variants.

In this section, we have given the proof that intent-aware approaches using greedy re-ranking guarantee a $(1 - \frac{1}{e})$ approximation of the optimal solution.

## 7.3 Calibrated Recommendation using Subprofiles

Calibrated recommendations are ones that reflect the user's tastes and interests, as revealed by the user's profile, and Steck defines these in terms of item features (see Section 2.4 and also [Ste18]). But the subprofile idea that we present in the previous chapters of this thesis opens an opportunity to define a new variant of calibrated recommendation, one which uses subprofiles instead of features, much as SPAD uses subprofiles where xQuAD uses features. We refer to the Steck's calibrated recommendation, which uses item features, as $\text{CR}_{\mathcal{F}}$; we refer to our variant, which uses subprofiles instead of item features, as $\text{CR}_{\mathcal{S}}$.

In $\text{CR}_{\mathcal{S}}$, the distributions $p$ and $q$ (Eqs. 2.14 and 2.15) are defined in the same way, writing $S$ in place of $f$:

$$p(S|u) = \frac{\sum_{i \in I_u^+} w_{u,i} p(S|i)}{\sum_{i \in I_u^+} w_{u,i}} \tag{7.8}$$

$$q(S|u) = \frac{\sum_{i \in RL} w_{r(i)} p(S|i)}{\sum_{i \in RL} w_{r(i)}} \tag{7.9}$$

The question is how to define $p(S|i)$, which replaces $p(f|i)$ in these equations. We use $p(S|i) = \frac{1}{|\mathcal{S}_i|}$, where $\mathcal{S}_i$ is the set of user $u$'s final subprofiles that item $i$ is *related to* (using Eq. 4.4).

We modify the $\text{C}_{\text{KL}}$ metric to use subprofiles instead of item features by replacing the distributions $p$ and $q$ over features in Eq. 2.16 with distributions over subprofiles, much as we did when we defined $\text{CR}_{\mathcal{S}}$ above. We will refer to $\text{C}_{\text{KL}}$ measured by item features as ${\text{C}_{\text{KL}}}^{\mathcal{F}}$ and $\text{C}_{\text{KL}}$ measured by subprofiles as ${\text{C}_{\text{KL}}}^{\mathcal{S}}$.

## 7.4 Experimental Results

In this section, we compare intent-aware and calibrated recommendations empirically. We want to reveal the extent to which intent-aware approaches do pro-

duce calibrated recommendations. We also want to evaluate calibrated recommendation more thoroughly than was done in [Ste18]. Steck's goal in [Ste18] was just "to illustrate that the proposed approach [i.e. calibrated recommendation] works as expected". Hence, he used just one dataset. He did not compare calibrated recommendation with any different recommenders: he simply compared calibrated recommendation (with different values of $\lambda$) with just its baseline recommender. While he did measure recall (as defined in [SG11]), his focus was on measuring the calibration metric, $\mathrm{C_{KL}}$. Here, we will use two datasets; we will compare two forms of calibrated recommendation ($\mathrm{CR}_{\mathcal{F}}$ and $\mathrm{CR}_{\mathcal{S}}$) to two forms of intent-aware recommendation (xQuAD and SPAD); and we will measure $\mathrm{C_{KL}}$ but also precision and diversity. In the case of diversity, we will show results for four different metrics, and we will explore the trade-off the recommenders make between precision and diversity.

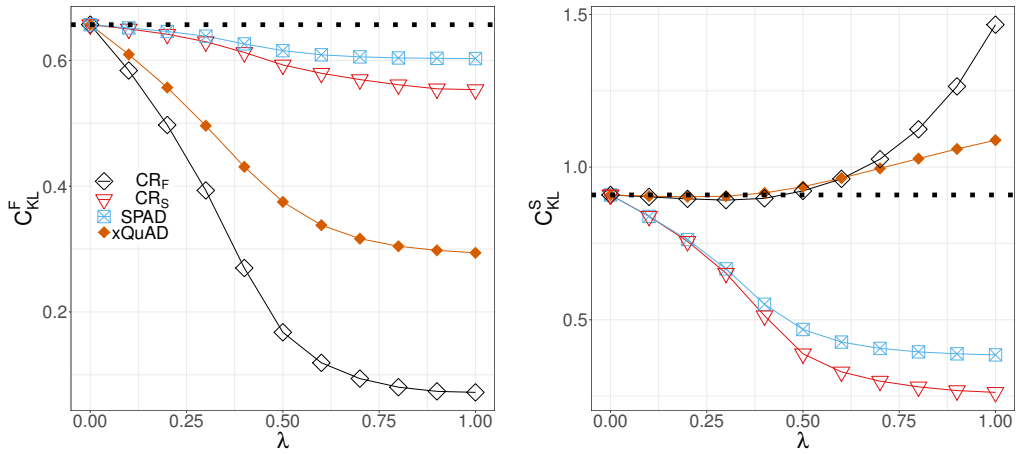
We report our empirical comparison of $\mathrm{CR}_{\mathcal{F}}$, $\mathrm{CR}_{\mathcal{S}}$, xQuAD and SPAD on the ML20M and TasteProfile datasets (see Chapter 3 for the details of the datasets). Note that the ML20M and TasteProfiles datasets are different from the datasets used in the experiments of the previous chapters. We use ML20M, which is a larger dataset than those used in Chapters 4, 5 and 6, since it is the same dataset used in [Ste18]. We also use TasteProfile to test on another publicly available larger dataset. Note also that, as we explain in Chapter 3, for ML20M and TasteProfile datasets, we use only one data split to create train, validation and test sets, unlike for the other datasets for which we use 5 data splits. We show the hyperparameter values that we used in the Appendix to this thesis.

First, we compare the performances of $\mathrm{CR}_{\mathcal{F}}$, $\mathrm{CR}_{\mathcal{S}}$, xQuAD and SPAD on the two versions of the calibration metric, $\mathrm{C_{KL}}^{\mathcal{F}}$ and $\mathrm{C_{KL}}^{\mathcal{S}}$. Next, we see how they affect the relevance of the baseline recommender by measuring Precision. Then, we look at their effect on 'pure' diversity metrics, $\mathrm{ILD}^{\mathcal{F}}$ and $\mathrm{ILD}^{\mathcal{S}}$, and relevance-aware diversity metrics, $\alpha\text{-nDCG}^{\mathcal{F}}$ and $\alpha\text{-nDCG}^{\mathcal{S}}$. Finally, we look at the trade-off between precision and diversity measured by the $\mathrm{ILD}$ metrics.

### 7.4.1 Calibration results

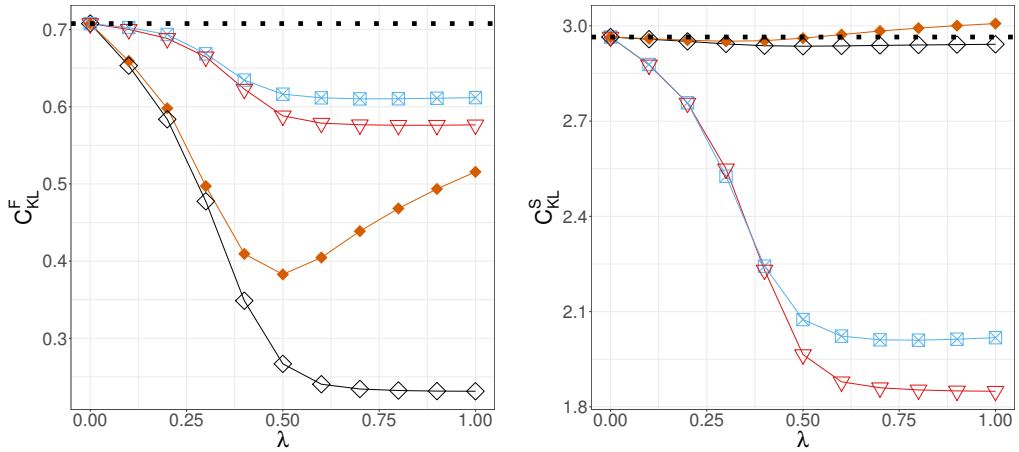
The calibration results are shown in Figure 7.1. It is important to keep in mind that for $\mathrm{C_{KL}}$ (unlike other results in this thesis) smaller values are better: smaller values mean better coverage of the user's interests.

Figure 7.1a shows results on the ML20M dataset when calibration is measured




(a) $C_{KL}{}^{\mathcal{F}}$ on ML20M

(b) $C_{KL}{}^{\mathcal{S}}$ on ML20M



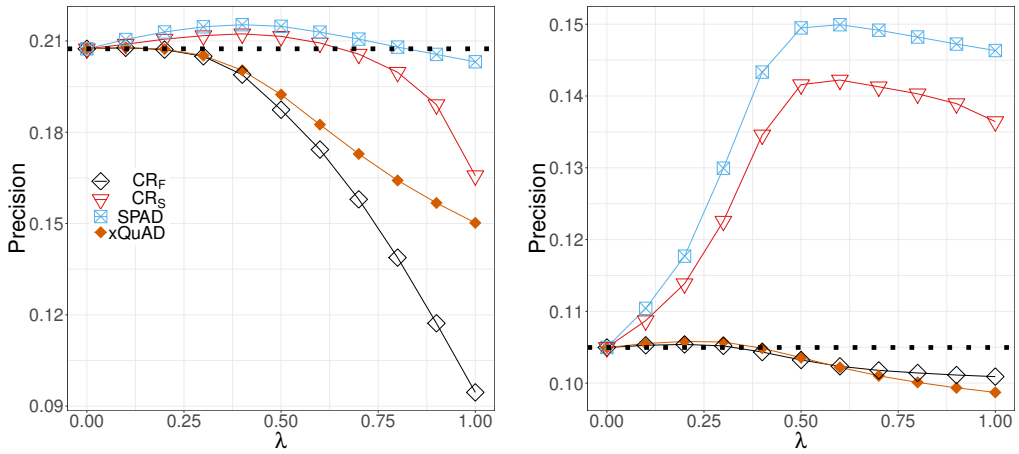

(c) $C_{KL}{}^{\mathcal{F}}$ on TasteProfile

(d) $C_{KL}{}^{\mathcal{S}}$ on TasteProfile

Figure 7.1: ML20M and TasteProfile datasets. $C_{KL}$ measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines.

using item features, $C_{KL}{}^{\mathcal{F}}$. CR$_{\mathcal{F}}$ performs best: it has the smallest values of $C_{KL}{}^{\mathcal{F}}$ for all values of $\lambda$. This result is to be expected: CR$_{\mathcal{F}}$ re-ranks baseline recommendations using the $C_{KL}{}^{\mathcal{F}}$ metric.

We hypothesized that intent-aware approaches would result in calibrated recommendations to a certain extent. xQuAD tries to cover different user tastes and interests defined by item features, and so it should do well for this metric. Indeed, it can be seen in Figure 7.1a that, for all values of $\lambda$, xQuAD results in good calibration: not as good as CR$_{\mathcal{F}}$, of course, but better than CR$_{\mathcal{S}}$ and SPAD. However, for all values of $\lambda$, even CR$_{\mathcal{S}}$ and SPAD recommendations are more calibrated than the baseline. This is noteworthy, since CR$_{\mathcal{S}}$ and SPAD make no use of item features, hence they are at a disadvantage.




(a) Precision on ML20M (b) Precision on TasteProfile

Figure 7.2: ML20M and TasteProfile datasets. Precision values for different values of $\lambda$. Values for MF are shown by dotted lines.

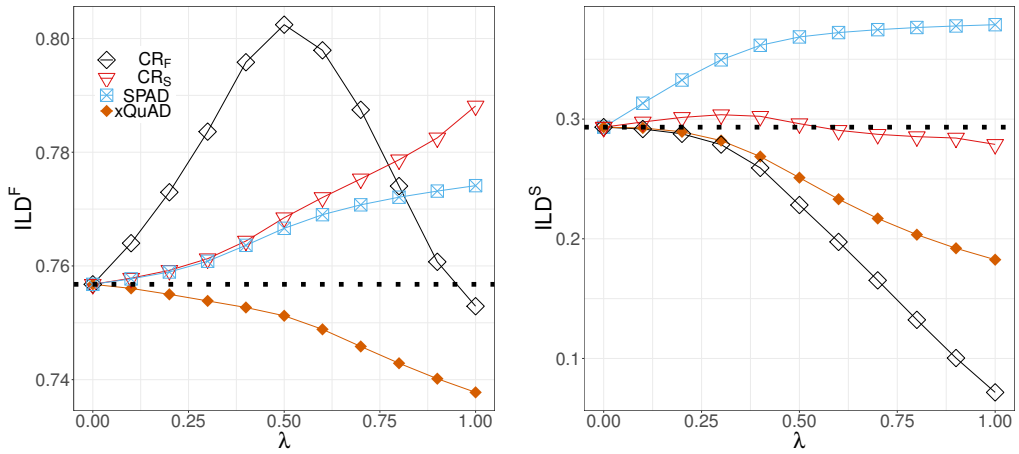
Figure 7.1b shows results on ML20M when calibration is measured using subprofiles, ${C_{KL}}^{\mathcal{S}}$. This time, it is xQuAD and $CR_{\mathcal{F}}$ that are at a disadvantage. We can see that, $CR_{\mathcal{S}}$ performs best, as expected. But, for all values of $\lambda$, SPAD performs better than the baseline too. For smaller values of $\lambda$, xQuAD and $CR_{\mathcal{F}}$ perform close to the baseline. But for larger values of $\lambda$, they perform worse than the baseline.

Figures 7.1c and 7.1d show calibration results on TasteProfile. The results are quite similar to those for ML20M. The difference is, when calibration is measured by subprofiles (Figure 7.1d), $CR_{\mathcal{F}}$ only performs close to the baseline for all values of $\lambda$.

Ideally, calibrated recommendations must not harm recommendation relevance. In the next subsection, we see how the four different re-ranking algorithms affect the precision of the baseline recommender.

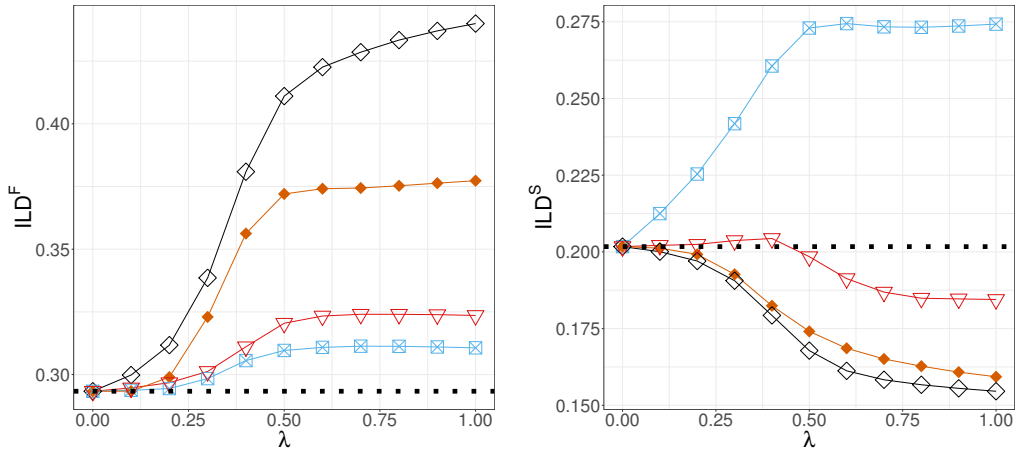
### 7.4.2 Precision

Figure 7.2a and 7.2b plot precision for different values of $\lambda$ on the ML20M and TasteProfile test sets, respectively. Consider ML20M first. $CR_{\mathcal{S}}$ and SPAD perform well. Precision only falls below the baseline for high values of $\lambda$: from 0.7 for $CR_{\mathcal{S}}$ and from 0.8 for SPAD. $CR_{\mathcal{F}}$ and xQuAD do not do so well: for many values of $\lambda$, their precision is lower than that of the baseline's original recommended set. $CR_{\mathcal{F}}$ suffers even more than xQuAD: its precision falls even




(a) $\text{ILD}^{\mathcal{F}}$ on ML20M

(b) $\text{ILD}^{\mathcal{S}}$ on ML20M

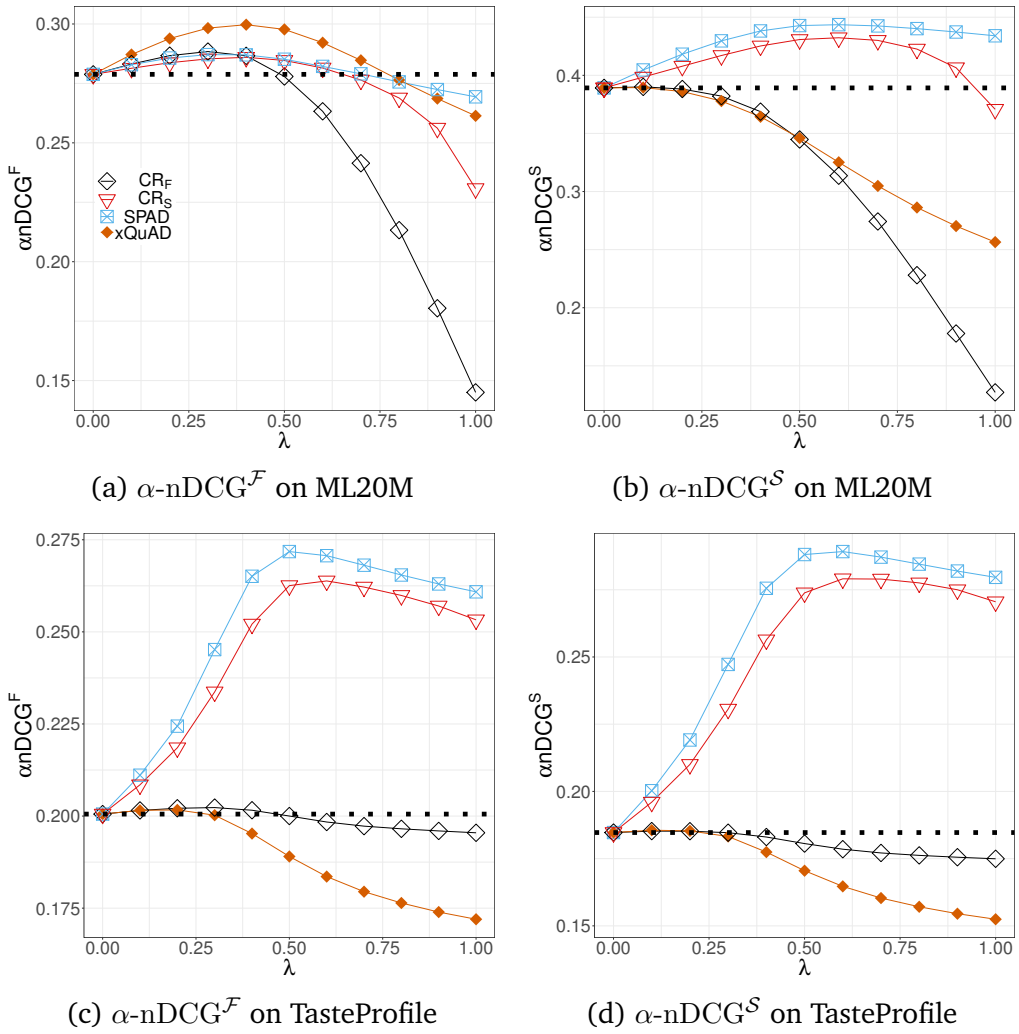



(c) $\text{ILD}^{\mathcal{F}}$ on TasteProfile

(d) $\text{ILD}^{\mathcal{S}}$ on TasteProfile

Figure 7.3: ML20M and TasteProfile datasets. ILD measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines.

more sharply as $\lambda$ grows. These $\text{CR}_{\mathcal{F}}$ results confirm those reported by Steck: he found that, for larger values of $\lambda$, $\text{CR}_{\mathcal{F}}$'s accuracy drops quickly (although he measured recall rather than precision) [Ste18]. We see similar results when we turn to the TasteProfile dataset. The main differences are that, $\text{CR}_{\mathcal{S}}$ and SPAD now achieve higher precision than the baseline for all values of $\lambda$; and $\text{CR}_{\mathcal{F}}$ and xQuAD suffer smaller decreases in precision relative to the baseline than they did on ML20M. The results presented in this subsection show a clear preference for approaches that use subprofiles, rather than item features.




(a) $\alpha$-nDCG$^{\mathcal{F}}$ on ML20M

(b) $\alpha$-nDCG$^{\mathcal{S}}$ on ML20M

(c) $\alpha$-nDCG$^{\mathcal{F}}$ on TasteProfile

(d) $\alpha$-nDCG$^{\mathcal{S}}$ on TasteProfile

Figure 7.4: ML20M and TasteProfile datasets. $\alpha$-nDCG measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines.

### 7.4.3 Diversity results

In this subsection, we measure diversity in different ways, enabling us to see the extent to which calibrated recommendations are diverse recommendations.

Figure 7.3 shows results for ILD, which is a 'pure' diversity metric. Figure 7.3a shows results on the ML20M dataset when ILD is measured using item features, ILD$^{\mathcal{F}}$. Surprisingly, CR$_{\mathcal{F}}$, which is not an algorithm that explicitly seeks to diversify result sets, achieves the highest values of ILD$^{\mathcal{F}}$. CR$_{\mathcal{S}}$ and SPAD are at their usual disadvantage when a metric uses features. But, for all values of $\lambda$, their ILD$^{\mathcal{F}}$ exceeds the baseline. xQuAD is another surprise. For all values of $\lambda$, it performs worse than the baseline. This appears to an idiosyncratic result,

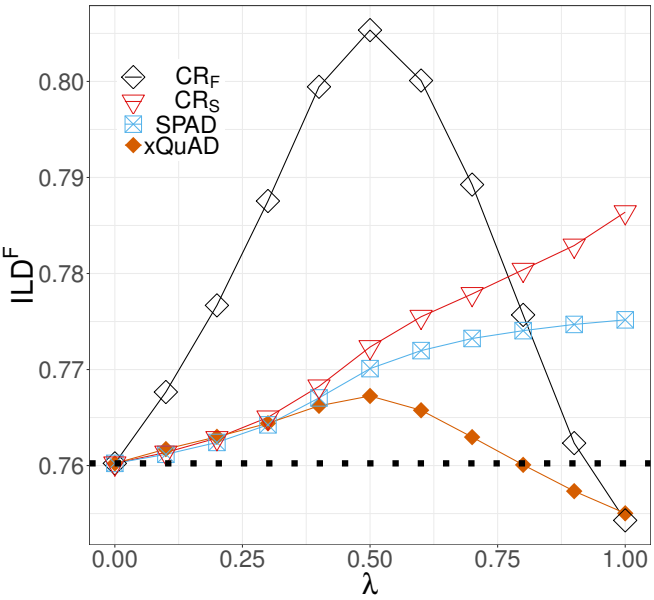



Figure 7.5: ML20M dataset without binarization. ILD values for varying $\lambda$. Values for MF are shown by dotted lines.

specific to this dataset. xQuAD's $\text{ILD}^{\mathcal{F}}$ on a number of datasets that contain explicit ratings, for example, usually exceeds the baseline (e.g. see [Var15] and also earlier in this thesis), and it exceeds the baseline by a small amount for most values of $\lambda$ when we run it on ML20M without binarization (see Figure 7.5).

When ILD is measured using subprofiles, $\text{ILD}^{\mathcal{S}}$ (Figure 7.3b), $\text{CR}_{\mathcal{F}}$ and xQuAD are the ones at a disadvantage. Sure enough, their diversity according to this metric always falls below the baseline, with $\text{CR}_{\mathcal{F}}$ worse than xQuAD. On the other hand, $\text{CR}_{\mathcal{F}}$ and SPAD are at an advantage. SPAD, the algorithm that actually seeks to diversify, produces the most diverse results sets, better than the baseline recommender for all values of $\lambda$; $\text{CR}_{\mathcal{F}}$, which only seeks to calibrate its recommendations, produces result sets that are quite similar in diversity to those produced by the baseline.

Figures 7.3c and 7.3d show ILD results on TasteProfile. Results for $\text{ILD}^{\mathcal{S}}$ (Figure 7.3d) are similar to those for ML20M. The difference is $\text{ILD}^{\mathcal{F}}$ (Figure 7.3c). For all values of $\lambda$, all of the re-ranking algorithms, including xQuAD, have higher values of $\text{ILD}^{\mathcal{F}}$ than the baseline.

While ILD is a 'pure' measure of diversity, $\alpha$-nDCG is a relevance-aware measure of diversity. The $\alpha$-nDCG results are in Figure 7.4.

Figure 7.4a shows results on the ML20M dataset when $\alpha$-nDCG is measured us-

ing item features, $\alpha\text{-nDCG}^{\mathcal{F}}$. For almost all values of $\lambda$, xQuAD performs better than the other re-ranking algorithms. For small values of $\lambda$, $\text{CR}_{\mathcal{F}}$ is competitive with $\text{CR}_{\mathcal{S}}$ and SPAD, but soon suffers from the largest decreases in $\alpha\text{-nDCG}^{\mathcal{F}}$. $\text{CR}_{\mathcal{S}}$ and SPAD perform similarly to each other. Even though they are at a disadvantage, they achieve higher $\alpha\text{-nDCG}^{\mathcal{F}}$ than the baseline for all but large values of $\lambda$.

Figure 7.4b shows ML20M results for $\alpha\text{-nDCG}^{\mathcal{S}}$, where xQuAD and $\text{CR}_{\mathcal{F}}$ are the algorithms that are at a disadvantage. SPAD always has higher $\alpha\text{-nDCG}^{\mathcal{S}}$ than all the other algorithms, including the baseline. $\text{CR}_{\mathcal{S}}$ has higher $\alpha\text{-nDCG}^{\mathcal{S}}$ than the baseline, except when $\lambda$ is large. xQuAD and $\text{CR}_{\mathcal{F}}$ are never better than the baseline and perform particularly poorly as $\lambda$ grows.

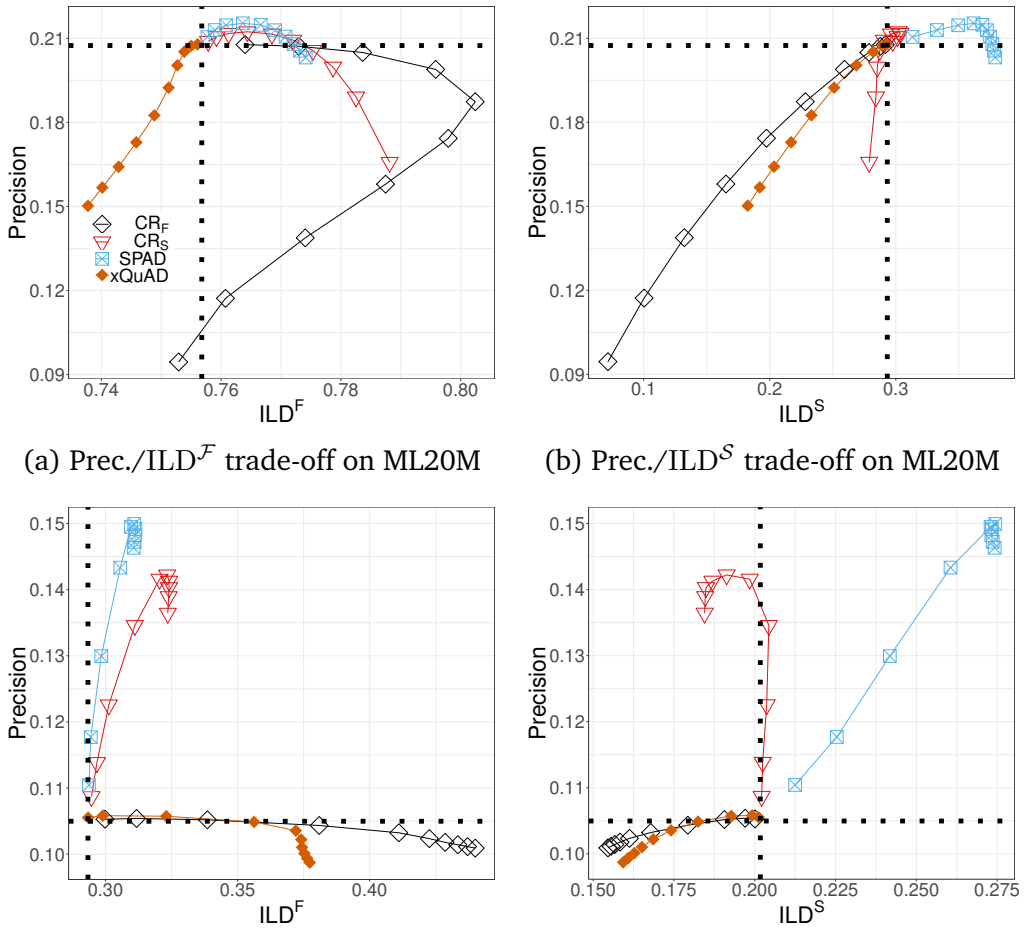
Figures 7.4c and 7.4d show $\alpha\text{-nDCG}$ results on TasteProfile. For $\alpha\text{-nDCG}^{\mathcal{S}}$ (Figure 7.4d), results are similar to those for ML20M. The difference is $\alpha\text{-nDCG}^{\mathcal{F}}$ (Figure 7.4c). Now, SPAD and $\text{CR}_{\mathcal{S}}$ are always higher than the baseline, while xQuAD and $\text{CR}_{\mathcal{F}}$ are similar to the baseline for small values of $\lambda$ and fall a little below the baseline for large values of $\lambda$.

The diversity results presented in this subsection show that a set of calibrated recommendations can be a diverse set of recommendations as well. Re-ranking approaches that use subprofiles as aspects, $\text{CR}_{\mathcal{S}}$ and SPAD, perform particularly well according to the relevance-aware diversity metric, $\alpha\text{-nDCG}$. In fact, SPAD increases diversity, no matter how it is measured, on both datasets for almost all values of $\lambda$. To aid visualization of the relevance/diversity trade-off better, the next subsection plots precision and $\text{ILD}$ together.

### 7.4.4 A visualization of trade-offs

Each subfigure in Figure 7.6 is divided into four parts by the dotted lines that plot the precision and $\text{ILD}$ values of the MF baseline. When, for a given value of $\lambda$, a re-ranking algorithm improves both precision and $\text{ILD}$ over the baseline, for example, it appears as a point in the top-right quadrant of the plot.

We can see that across all four subfigures (i.e. for both datasets and both version of $\text{ILD}$), assuming that we regard precision and diversity as equally important, SPAD is best by far. It most often increases both precision and diversity. $\text{CR}_{\mathcal{S}}$ is second best according to these visualizations. $\text{CR}_{\mathcal{F}}$ and xQuAD are not competitive.




(a) Prec./ILD$^{\mathcal{F}}$ trade-off on ML20M

(b) Prec./ILD$^{\mathcal{S}}$ trade-off on ML20M

(c) Prec./ILD$^{\mathcal{F}}$ trade-off on TasteProfile

(d) Prec./ILD$^{\mathcal{S}}$ trade-off on TasteProfile

Figure 7.6: ML20M and TasteProfile datasets. Precision vs. ILD trade-off plots measured using features and subprofiles for different values of $\lambda$. Values for MF are shown by dotted lines.

### 7.4.5 Other results

For completeness, we should mention two other approaches to diversification whose results we have computed but which we do not show because they clutter the plots but are not competitive.

The first is the traditional approach to diversification, namely Maximal Marginal Relevance (MMR) [CG98] and its variants, e.g. [SM01, ZMKL05].

While MMR has some of the highest values for ILD, as one would expect, since this is close to what it optimizes, it performs poorly on all other metrics, almost always having the worst values for precision, for example.

The second approach is one proposed by Steck in [Ste18], which we will re-

fer to here as a *diversity-enhanced calibrated recommender*, whose goals was to bring extra diversity to calibrated recommendations [Ste18]. Steck does this by introducing a diversity-promoting prior, $p_0(f)$, to recommend from features that are not in the user's profile. Specifically, probability distribution $p(f|u)$ is replaced by $\tilde{p}(f|u) = \beta p_0(f) + (1-\beta)p(f|u)$. There are no experimental results in [Ste18]. But we have implemented and evaluated this version of calibrated recommendation as well. We take $p_0(f)$ to be the average $p(f|u)$ over the all the users [Ste18]. The most useful results were obtained when we optimized $\beta$ for $\alpha$-nDCG rather than for precision or ILD. In this case, we found that, although the diversity-extended calibrated recommender does improves diversity a little (measured by ILD), it harms precision and the $\mathrm{C_{KL}}$ metrics. Overall, it was not competitive with the simpler forms of calibrated recommendation.

## 7.5 Conclusions

In this chapter, we compared calibrated and intent-aware recommendation. They have apparent similarities in that both try to cover the user's interests to a certain extent. First, we analyzed that the greedy approach to intent-aware recommendation, including xQuAD, SPAD and RSPAD, enjoy a $1 - \frac{1}{e}$ approximate optimality guarantee, similar to Steck's calibrated recommendations. We also defined a new instantiation of calibrated recommendation that uses subprofiles ($\mathrm{CR}_{\mathcal{S}}$) in place of features ($\mathrm{CR}_{\mathcal{F}}$).

We find that intent-aware recommendation results in calibrated recommendations to a certain extent, and calibrated recommendation results in diverse recommendations to a certain extent. We also see that re-ranking approaches using features, xQuAD and $\mathrm{CR}_{\mathcal{F}}$, harm precision a lot. The re-ranking approaches that use subprofiles, SPAD and $\mathrm{CR}_{\mathcal{S}}$, achieve the highest precision, achieve good calibration according to both calibration metrics, and achieve good diversity according to both $\alpha$-nDCG metrics. SPAD also achieves good diversity according to both ILD metrics and suffers least from the relevance/diversity trade-off.

In the next chapter, we apply SPAD to the task of Automatic Playlist Continuation. By applying SPAD to a different task on a larger dataset, we show the generality of SPAD and its variants.

# Chapter 8

# Automatic Playlist Continuation using SPAD

In this chapter, we take the idea of SPAD and apply it to a task that is different from the one in the previous chapters of this thesis, namely Automatic Playlist Continuation (APC). In the previous chapters of the thesis, the main goal was to improve diversity along with relevancy. In the APC task, the details of which are explained in the next section, the main goal is not diversity but to accurately predict tracks that are missing from the playlists.

The emergence of online music streaming services like Spotify, Pandora, Deezer, Apple Music and Amazon Music has increased the value of research related to music recommendation. Although music recommender systems often successfully recommend songs that satisfy users, in the sense of fitting the users' preferences, there are still a lot of challenges to be tackled [SZC+18]. Automatic playlist continuation (APC) is one such challenge. The aim in APC is to help users to create and extend their own playlists.

In the previous chapters, we show that on several datasets, using subprofiles as aspects instead of item features results in more accurate and diverse recommendations. In this chapter, we show that, even applied to the task of APC, the idea of using subprofiles result in more accurate and diverse recommendations.

The *ACM RecSys Challenge 2018*, organized by Spotify, the University of Massachusetts and Johannes Kepler University is all about APC. Using datasets of playlists made available by Spotify, participants build systems to automatically predict tracks that are missing from test playlists.

In this chapter, we take the idea of SPAD and apply it to the task of the Challenge. Our approach to APC is twofold: Cold-Start-APC for short playlists and SPAD-APC for other playlists. Cold-Start-APC is a rudimentary popularity-based recommender. SPAD-APC treats playlists as if they were user profiles. It builds an implicit matrix factorization model to generate initial recommendations. But it re-ranks those recommendations using SPAD's intent-aware diversification method. The SPAD re-ranking method aims to ensure that the final set of recommendations covers different interests or tastes in the playlists of the users, which we refer to as subprofiles. We show that such subprofiles do exist within playlists and we show that the SPAD method achieves higher precision than matrix factorization alone.

In the rest of this chapter, we give an overview of the Challenge, then describe our approach to the Challenge in detail. We explain the resources we used and our experimental methodology, and then give some results.

## 8.1 Challenge Overview

The Challenge focuses on music recommendation, specifically automatic playlist continuation (APC) [CLSZ18]. The task is to recommend appropriate tracks to add to a playlist.

For this task, Spotify released the Million Playlist Dataset[1] (MPD), containing 1,000,000 playlists created by Spotify users. Some statistics about the MPD can be found in Table 8.1. Each playlist is represented by playlist metadata (the title, description if available, the number of unique tracks, etc.) and the tracks in the playlist, together with metadata about the tracks (such as the name of the track, album and artist).

Spotify also released the Challenge Set, comprising 10,000 incomplete playlists (i.e. some of the tracks are hidden). Specifically, in the Challenge Set, the number of seed tracks in a playlist has values from the set $\{0, 1, 5, 10, 25, 100\}$. Furthermore, the way in which the seed tracks were chosen (e.g. initial tracks or random tracks) and the availability of a playlist title mean that there are 10 different categories of playlists, with 1,000 playlists per category. This is summarized in Table 8.2.

[1] https://recsys-challenge.spotify.com/

The task is to recommend 500 tracks to each of the playlists in the Challenge Set. These recommendations are evaluated by Spotify against the ground-truth $G$, i.e. the tracks that were actually hidden. The evaluation metrics are precision@$|G|$, NDCG and a bespoke measure called Clicks that is based on rank within the ground-truth, combined by a Borda count.

However, this RecSys Challenge is split into two sub-challenges. In the Main Challenge, participants must train their prediction models exclusively on the MPD. In the Creative Challenge, participants are allowed to use public external data sources. Although we do not use any external data, we build our model using the union of the MPD and the Challenge Set. This means that our system is obliged to compete in the Creative Challenge.[2]

The Spotify dataset contains no explicit item features (e.g. there are no track genres). To allow us to use the conventional diversity metrics in the experiments (which, as we have seen, require item features), we crawled music genre metadata by using Spotipy[3], a lightweight Python library for the Spotify Web API. However, this API does not provide track genres, only artist genres. Therefore, we indirectly assigned genres to tracks through the genres of their artists. We check whether the artist that a track belongs to has a genre or not. If the artist has genre information, we assign the genre information of the artist to the track that belongs to the corresponding artist. There are 295,860 unique artists in MPD. For 93,790 of those artists we were able to crawl genre information, i.e. we have crawled genre information for 31.7% of the artists in MPD. As a result, 2048 different genres are assigned to the tracks.

As we explain in Section 8.4, due to resource limitations we eliminated all tracks that appear in only one MPD playlist, cutting the number of unique tracks from 2,262,292 to 1,189,252. 944,973 of those 1,189,252 have been assigned with genre information, i.e. 79.46% of the tracks used in the experiments have genre information.

[2]On the RecSys Challenge forum, in reply to the question "[is it] allowed to include the information in the challenge set for the model training?", the reply was: "The rules say that for the main track you can only use the MPD. The challenge set is not part of the MPD, so for the main track, the answer is 'no'. For the creative track you can use publicly available data and the challenge set qualifies as that — so for the creative track, the answer is 'yes'." `https://groups.google.com/forum/#!topic/recsyschallenge-2018/1F-QCl2se4E`

[3]https://spotipy.readthedocs.io/en/latest/

Table 8.1: MPD statistics

| | |
|---|---|
| # of playlists | 1,000,000 |
| # of tracks | 66,346,428 |
| # of unique tracks | 2,262,292 |
| # of unique albums | 734,684 |
| # of unique artists | 295,860 |
| # of unique titles | 92,944 |
| # of unique normalized titles | 17,381 |
| # of playlists with description | 18,760 |
| avg. playlist length | 66.3 |

Table 8.2: Challenge dataset statistics

| # of playlists | title | tracks | cold-start |
|---|---|---|---|
| 1000 | ✓ | none | ✓ |
| 1000 | ✓ | first track | ✓ |
| 1000 | ✓ | first 5 tracks | X |
| 1000 | X | first 5 tracks | X |
| 1000 | ✓ | first 10 tracks | X |
| 1000 | X | first 10 tracks | X |
| 1000 | ✓ | first 25 tracks | X |
| 1000 | ✓ | random 25 tracks | X |
| 1000 | ✓ | first 100 tracks | X |
| 1000 | ✓ | random 100 tracks | X |

## 8.2 Other Approaches to the Challenge

In this section, we summarize some of the approaches taken by the top-performing teams who entered the Creative Track of the Challenge (see Section 8.6.3 for the final leaderboard). For a detailed analysis of the approaches for both the Main and Creative Tracks, refer to the paper by Zamani et al. [ZSLC18].

Almost all of the top performing teams use the Spotify API to crawl for audio features as an external data source. One team computed their own audio features from the downloaded 30-second samples of the tracks [FBY+18]. Some of the teams use pre-trained word embeddings as external data to create track embeddings [KMI18]. In our approach, which we present in next section, unlike the other teams, we do not use external data such as audio features or pre-trained word embeddings.

The winning team in both the Main and Creative Tracks used a two-stage approach to the task [VRC+18]. For the Creative Track, first, they learn lower dimensional representations of the playlists and tracks by using matrix fac-

torization. Then they use those representations together with other features in a gradient-boosting learning-to-rank model, XGBoost [CG16]. Some of the other features that they use are playlist embeddings learnt by neural-networks, nearest-neighbours based collaborative filtering models and content-based music descriptors of the tracks.

The team that came second in the Creative Track [ABC+18] uses an ensemble of different techniques, including a popularity-based recommender, track & playlist based collaborative filtering models, and track & playlist based content-based filtering models. They also use natural language processing techniques to enhance the playlist titles to be used in the ensemble model. They use track features acquired from the Spotify API as additional features.

The team that came third in the Creative Track [LKLJ18] use a less complex approach. They combine nearest-neighbourhood collaborative filtering techniques with a matrix factorization algorithm. They also use audio features as external data.

Our approach is quite different from all of these. We explain it in detail in the next section.

## 8.3 Our Approach

Let $I$ be the set of all items (i.e. tracks) in the union of the MPD Set and Challenge Set. A playlist, which we will designate by $u$, is a *set* of items, $u \subseteq I$. Let $\mathrm{title}(u)$ be the title of playlist $u$, if it has one (or $\perp$ if it does not have a title). Let $U_{MPD}$ be the set of playlists in the MPD and $U_{CS}$ be the set of playlists in the Challenge Set; then $U = U_{MPD} \cup U_{CS}$. The *candidate* items that can be added to a given playlist $u \in U_{CS}$ are all the items in $I$ less those that are already in $u$: $I \setminus u$. We compute recommendations in one of two ways, depending on the length of $u$. The first way we refer to as the Cold-Start-APC; the second we refer to as SPAD-APC. Our implementation is publicly available.[4]

We consider playlists $u \in U_{CS}$ to be cold-start playlists if they have a title and either zero or one track; see the last column in Table 8.2. We present Cold-Start-APC in Section 8.3.1. For the remaining 8,000 playlists in $U_{CS}$, candidates are scored by a matrix factorization algorithm and then re-ranked using SPAD

[4] https://github.com/mesutkaya/SpotifyRecSysChallenge2018

(see Section 8.3.2).

Before presenting details of the two approaches, we make a few additional observations. First, we do not, in fact, use the full set of items $I$. As we explain in Section 8.4, we exclude some items from $I$ to improve compute-times. Second, it follows from the formulation given above, that we are not using any metadata, except for the title of the playlist, where available (and, in fact, this is only used by Cold-Start-APC, not by SPAD-APC). Third, since we treat playlists as *sets* of items, for this prediction task we are ignoring the ordering of the items in the playlist. There is some debate about the significance of ordering in playlists. For example, Schedl et al. think that it is important [SZC+18] and there are approaches that take it into account (e.g. [BP06]). But according to Tintarev et al., there is actually little evidence that the exact order of the tracks matters to users [TLL17]. In any case, in our solution we are not using the ordering. Fourth, we are, in effect, treating playlists in the way that a regular recommender would treat a user profile (and this explain why we designate them by $u$). In a regular recommender, a positive-feedback-only implicit ratings profile would just be a set of items: the ones the user likes; in our recommender, a playlist is just a set of items.

### 8.3.1 Cold-Start-APC

We use Cold-Start-APC for the 1000 playlists in the Challenge Set that have title only and the 1000 playlists that have a title and one track (their first).

For each candidate $i \in I \setminus u$, Cold-Start-APC computes a score, $s(u, i)$, based on the popularity of $i$ across the playlists in $U_{MPD}$, and recommends the 500 candidate items that have the highest scores. To improve the scoring, we use a 'normalization' function provided by Spotify which converts track titles to lowercase and removes punctuation symbols. As shown in Table 8.1, 92,944 different titles become 17,318 unique titles after normalization.

For a cold-start playlist $u$, the predicted score $s(u, i)$ for a candidate track $i \in I \setminus u$ is computed as follows:

$$s(u, i) = \sum_{v \in U_{MPD}} \mathbb{1}(u, i, v) \tag{8.1}$$

In the case where $u$ has a title but no tracks:

$$\mathbb{1}(u,i,v) = \begin{cases} 1 & \text{if } i \in v \text{ and } \text{title}(u) = \text{title}(v) \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

In the case where $u$ has a title and one track:

$$\mathbb{1}(u,i,v) = \begin{cases} 2 & \text{if } i \in v \text{ and } u \subseteq v \text{ and } \text{title}(u) = \text{title}(v) \\ 1 & \text{if } i \in v \text{ and } (\text{title}(u) = \text{title}(v) \text{ xor } u \subseteq v) \\ 0 & \text{otherwise} \end{cases} \tag{8.3}$$

Note that, for some of the playlists with title only, it can be the case that the recommender cannot recommend 500 tracks with non-zero scores. In this case, we fill the rest of the recommendations with the most popular tracks in $U_{MPD}$.

There is no doubt that our cold-start solution is rudimentary, and there are many ways it could be improved, perhaps especially by using external data sources.

### 8.3.2 SPAD-APC

Our approach for the remaining 8000 playlists in the Challenge Set, having at least 5 tracks each, is based on the idea of SPAD. The goal of the SPAD and its variants is to generate a set of recommendations where each recommendation is relevant but the set of recommendations is diverse. In the APC task, at least in the way it is formulated by Spotify in the Challenge, relevance is the only criterion; diversity is not important. This raises the question of why we should apply the SPAD diversification technique. The answer is that, in Chapter 5 and 6, across multiple datasets, we have found that SPAD and its variants most of the time improve both precision and diversity. This is notable because many approaches to diversification trade-off accuracy for diversity.

It is also worth asking: are there reasons for thinking that diversity will be helpful for APC in general? Lee et al. report the results of a user study into playlists that were generated automatically using content-based similarity [LBM11]. A common concern among the participants in the user study was that consecutive tracks in the playlists were too similar; they also complained about a lack of variety in the playlists. Interpretations of 'variety' differed from user to user,

e.g. variety in terms of genres, styles, artists, etc. A tentative implication is that users like playlists to have some diversity and, if creating their own playlists (like the ones provided by Spotify), they might aim to give them a level of diversity. If this tentative implication is correct, then there are reasons to try a diversification technique in APC.

As usual, we use NN-1 (see Section 4.3.1.1 for the details) as the subprofile detection algorithm.

We take the idea of SPAD and apply it to APC. As already mentioned, we treat each playlist $u \in U$ as if it were a user's profile. Our baseline recommender (whose recommendations get re-ranked by SPAD) is a fast alternating-least-squares (ALS) implementation of matrix factorization for implicit and explicit datasets [PZT10]. We chose this as our baseline because in the previous chapters, for different datasets, it is the most accurate baseline recommender.

Ordinarily, SPAD diversifies a set of recommendations to cover the different tastes (subprofiles) that we extract from a user's profile. It is not obvious that a playlist will similarly contain different tastes and therefore not obvious that SPAD-style diversification will be of benefit to the APC task. Evidence of the benefit is given in Section 8.6.

Our approach differs from the other teams' approaches to the Creative Track (see Section 8.2). First, unlike the other teams, we do not use any external data such as audio features at all. Second, as we mentioned before, although relevance is the only criterion in the challenge, we use a diversification approach.

## 8.4 Resources

Some of our decisions were constrained by available resources, which are explained here.

By the rules of the Challenge, participants could submit only one set of predictions for evaluation per day. We started working (part-time) on the Challenge in its last three weeks, so this gave us a very small number of opportunities to test the performance of our approach on the Challenge Set. However, this did motivate us to create a validation set (see next section) so that we could test algorithm variants and find good values for hyperparameters. Not only was this

expedient, we hope that it helped us to avoid overfitting our solution to the Challenge Set.

We ran our algorithm on a personal laptop with a 2.5 Ghz Intel Core i7 and 24 GiB memory. Running experiments using a laptop for this large dataset was challenging, especially for the matrix factorization algorithm. It took some time to prepare a daily submission to the public leaderboard. This in turn constrained the size of validation set that we were able to work with (next section).

It also resulted in us making a major decision. Before applying our approach, we eliminated all tracks that appear in only one MPD playlist. The number of unique tracks was cut from 2,262,292 to 1,189,252. Of course, this reduction in the item space improved run-times considerably. But it has a negative effect on recommendation accuracy: since we can never recommend the eliminated tracks, we lose out (for resource reasons, rather than algorithmic reasons) if those tracks ever appear in the ground truth.

## 8.5 Methodology

The methodology that we follow in this chapter is different from the methodology explained in Chapter 3. The main reasons are that the MPD dataset is larger than other datasets used in the experiments throughout this thesis and the task of the challenge is a different task. We explain the methodology that we use for APC in this section.

The baseline matrix factorization algorithm that we use has two hyperparameters: the number of latent factors $d$ and a confidence level $\alpha$. SPAD also has its own hyperparameters, which are the number of neighbours while detecting the subprofiles ($k_{nn}$) and the number of neighbours in Eq. 4.4 ($k_{ind}$). For the re-ranking in Eq. 2.2, there is $\lambda$, controlling the balance between accuracy and diversity.

To find good values for these hyperparameters, we used a validation set. We randomly selected 10,000 playlists. We kept 80% of their tracks as part of the training data and held out 20% as validation data. This is quite a small validation set (10,000 playlists compared with 1,000,000); it may not be representative enough to optimize hyperparameter values. Its size was determined by the resources we had available (previous section).

We used a grid search. For $d$, $\alpha$, $k_{nn}$, and $k_{ind}$, we tested values from the set $\{10, 20, 30, \ldots, 100\}$. For $\lambda$, we tested values from $\{0.1, 0.2, 0.3, \ldots, 1.0\}$. We selected the values that maximize precision, which were as follows: $d = 100$, $\alpha = 50$, $k_{nn} = 30$, $k_{ind} = 70$ and $\lambda = 0.4$.

Notice that, in the other chapters, we had a two-step hyperparameter optimization process. When choosing hyperparameter values for baseline recommenders, we optimized precision. But when choosing hyperparameter values for the re-ranking algorithms (such as SPAD), we optimized $\alpha$-nDCG. In this chapter, because the challenge evaluates the playlists only for accuracy, not for diversity, we use precision to optimize both the hyperparameters of the baseline recommender (MF) and of SPAD. Hence, we have only a one-step hyperparameter optimization process here.

For the results presented in Section 8.6.2, we train the baseline using its selected hyperparameter values on the 80% of the tracks in the 10,000 playlists selected as validation set, and, for each of the 10,000 playlists, we generate a recommendation set RS, where |RS| = 500. Then, we re-rank each RS using SPAD with its selected hyperparameter values to produce the final set of recommendations. Finally, we measure the evaluation metrics on the remaining 20% of the tracks of each playlist.

## 8.6 Experimental Results

We divide this section into three: first we analyze the subprofiles that we found in the playlists; then we give some experimental results on the validation set; finally, we use data from the public leaderboards to compare our team's performance with other teams.

### 8.6.1 Subprofile analysis

We begin by asking: do playlists actually contain sub-tastes? Or, in other words, do playlists, when treated as if they were user profiles, contain subprofiles? If they do not, then there is little prospect that SPAD will work well on the APC task. We show some data here that suggests that playlists do contain sub-tastes (subprofiles).

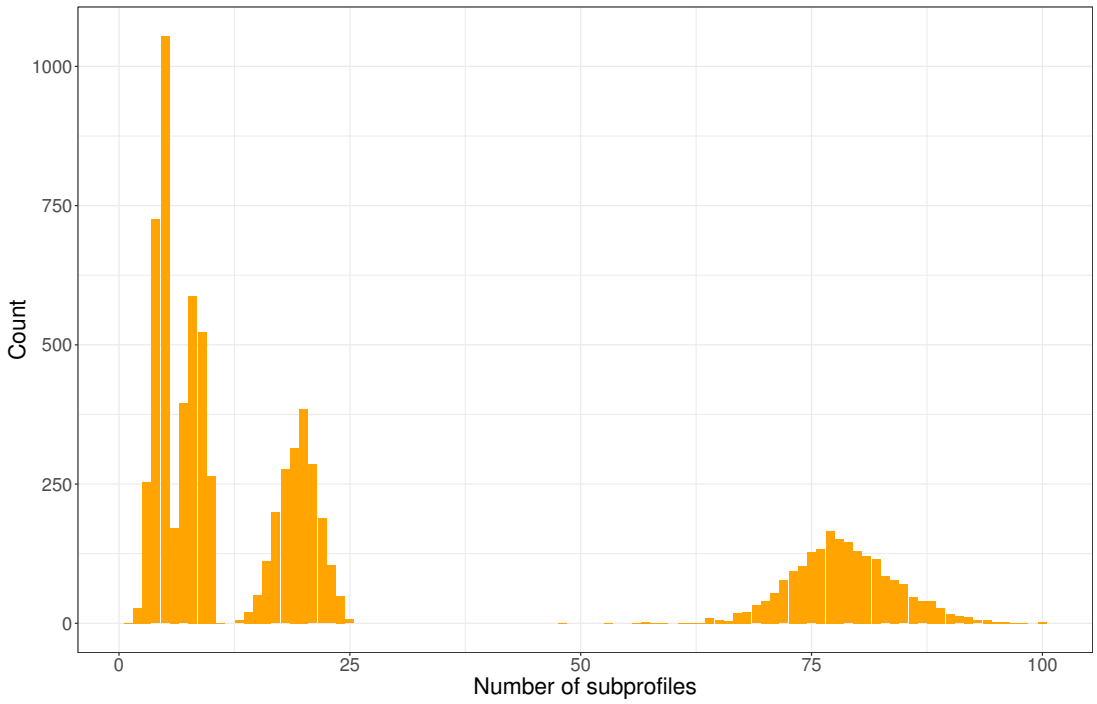



Figure 8.1: Number of subprofiles.

We applied SPAD's subprofile detection method (explained in Section 8.3.2) to the 8,000 playlists in the Challenge dataset that have at least 5 tracks. Figure 8.1 is a histogram showing the frequencies of different numbers of subprofiles. In other words, it shows how many of the 8,000 playlists contain just one subprofile, how many contain two subprofiles, how many contain three, and so on.

If a playlist has just one subprofile, then that subprofile comprises the whole playlist, and this would be a playlist with no sub-tastes. From the histogram, we see that the number of playlists that fall into this category (i.e. ones with just one subprofile) is very small. Perhaps surprisingly, playlists do contain sub-tastes. Four subprofiles is most frequent, but some playlists have as many as 100.

What also matters is how long these subprofiles are, and this is shown in Figure 8.2.

From the histogram, we see that some are very short: even a single song where that song is not a good enough neighbour to other songs in the playlist for it to join their subprofile. But most subprofiles comprise two or more tracks.

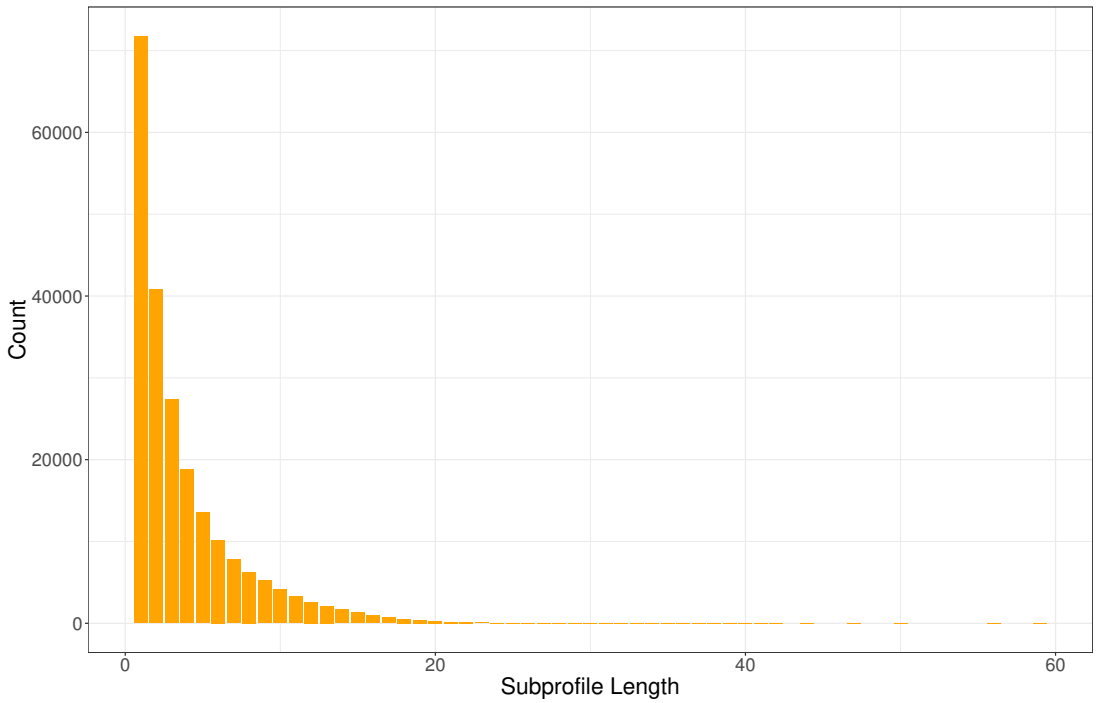



Figure 8.2: Subprofile length.

### 8.6.2 Validation set results

Here we show a few validation set results that confirm that SPAD re-ranking provides an advantage over using just its baseline matrix factorization algorithm.

In Figure 8.3, for all 10,000 playlists in the validation set, we plot Precision (Figure 8.3a), and diversity measured as $\alpha$-nDCG (Figure 8.3b), ERR-IA (Figure 8.3c), S-recall (Figure 8.3d), ILD (Figure 8.3e) and EILD (Figure 8.3f) for different values of $N$ for top-$N$ recommendations.

It can be seen that re-ranking the matrix factorization recommendations using SPAD always increases precision over matrix factorization. From the point of view of the Challenge, this is all that matters. But we will look at the diversity results too.

We can see in the figures that using SPAD always increases diversity but with one exception. The exception is the ILD metric: for large values of $N$, the MF baseline performs better than SPAD. Interestingly, even when diversity is measured by S-recall and ILD, which are not relevance-aware metrics, SPAD performs well. For S-recall, SPAD always performs slightly better than the MF baseline. For ILD, especially for smaller values of $N$, SPAD outperforms the MF

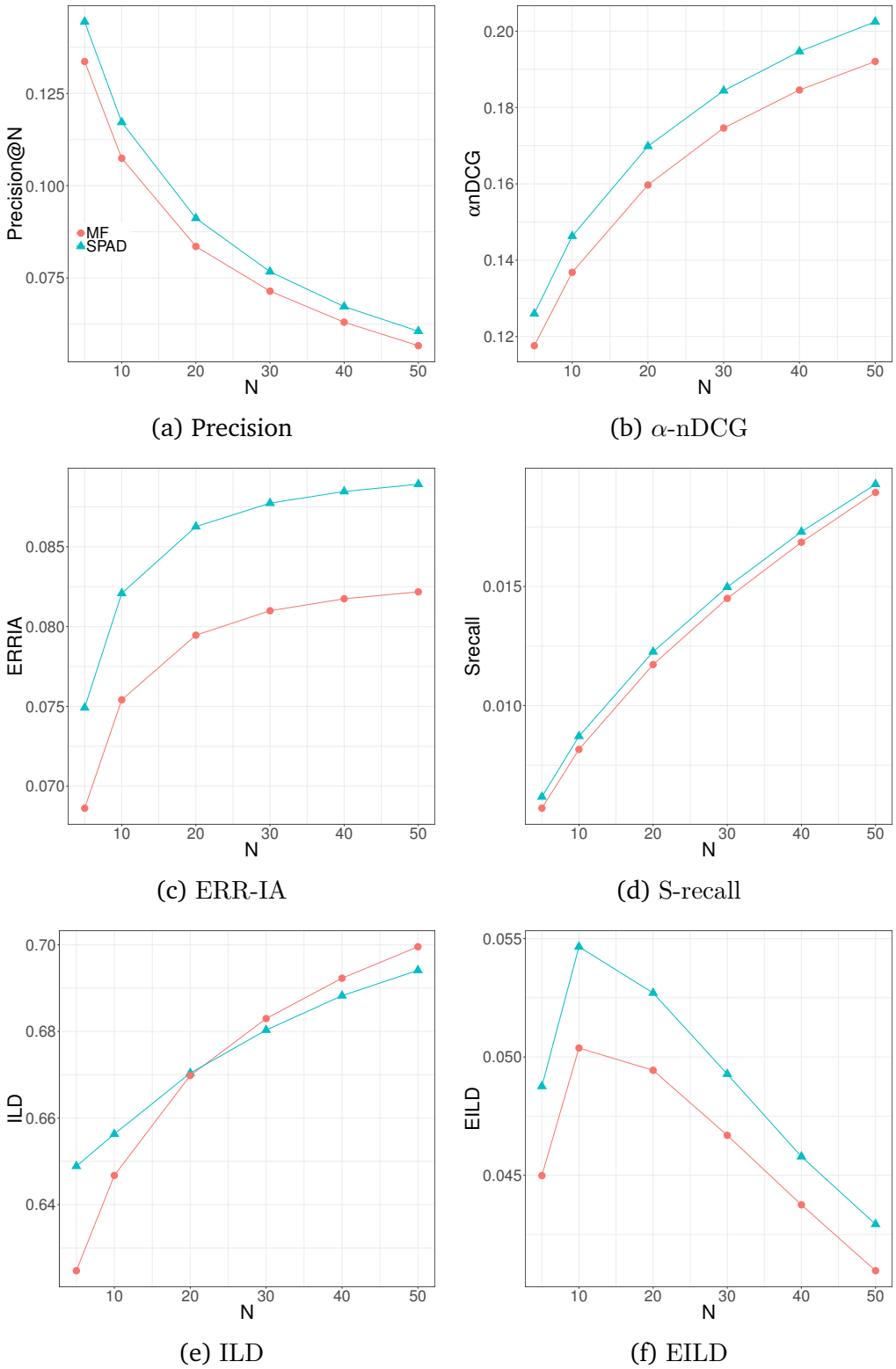



Figure 8.3: Validation set results against the number of recommendations, $N$.

baseline. It is also noteworthy that SPAD always results in more relevant and diverse recommendations for higher ranks in the recommendation list, top-5

and top-10 for example. In Spotify playlists, users are interacting with top-10 recommendations first.

We also look at the values of the metrics for different playlist sizes and for different numbers of subprofiles. For the 10,000 playlists in the validation set, Figure 8.4 shows the precision and diversity of the MF baseline and of SPAD for playlists of up to 5 tracks, from 6 to 10 tracks, 11 to 25 tracks, 26 to 50 tracks, 51 to 100 tracks, and 101 to 200 tracks.

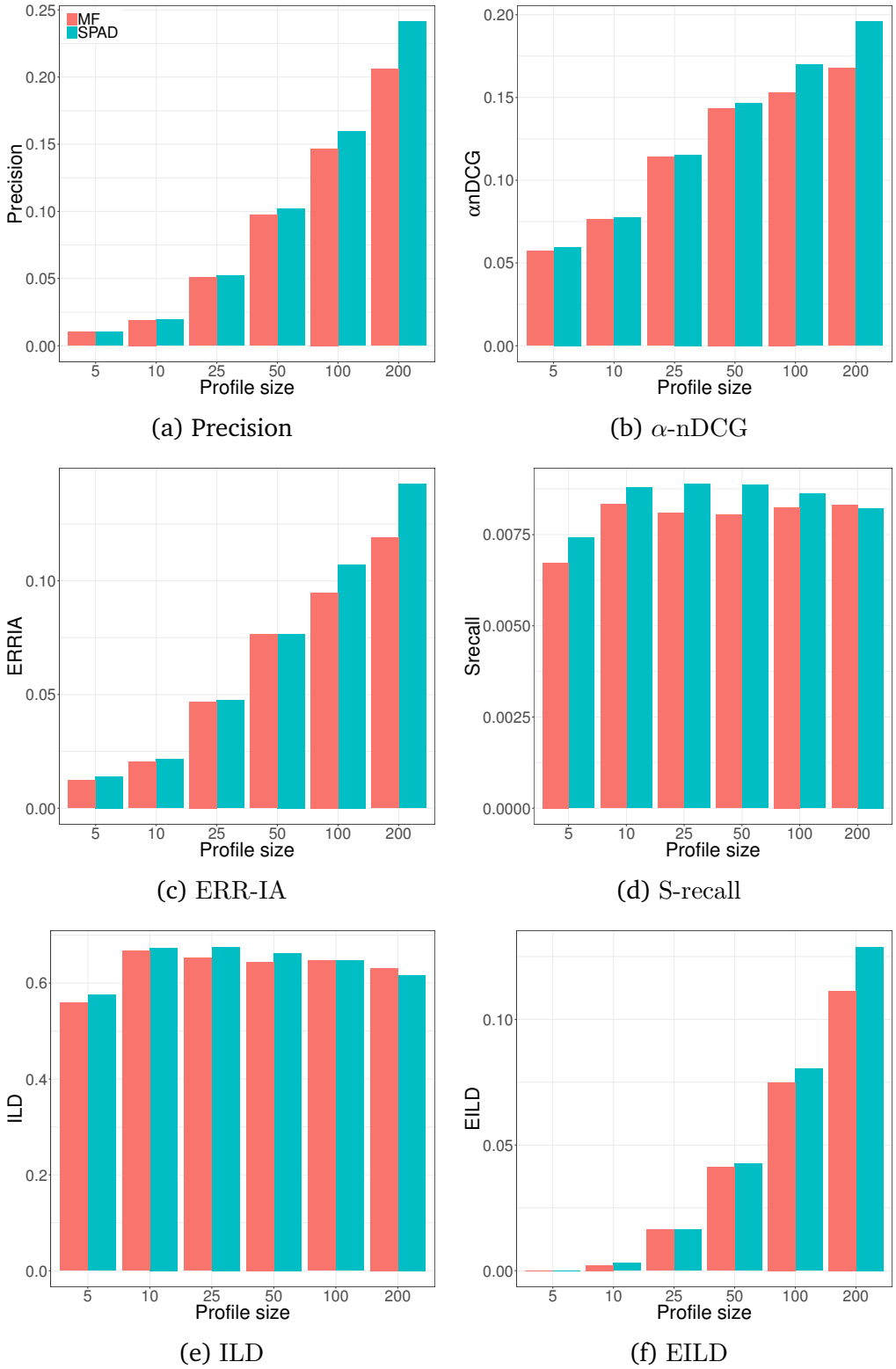
Similarly, Figure 8.5 shows the precision and diversity for the validation set but this time showing results for playlists that have up to 5 subprofiles, from 6 to 10 subprofiles, 11 to 25, 26 to 50, 51 to 100 and 101 to 200.

The figures show that, usually as the playlist length and the number of subprofiles increases, the amount by which SPAD outperforms matrix factorization also increases. The more songs there are in a playlist, the more subprofiles there are, but also the more precision and diversity measured by $\alpha$-nDCG, ERR-IA and EILD benefits from SPAD re-ranking. ILD and S-recall do not follow the same pattern. They usually benefit from SPAD re-ranking, but when there are more than 100 tracks in the playlists and playlists have more than 100 subprofiles SPAD starts to trade-off diversity for precision.

Considering the task of the challenge, which aims to maximize the relevancy of the recommended tracks to be added to playlists, arguably the figures also show that for the shortest playlists, precision for both SPAD and matrix factorization is so low that we might have benefited instead from applying our cold-start strategy (or a more sophisticated cold-start strategy) to more playlists than we did.

### 8.6.3 Leaderboard results

Among 32 teams, our team (*teamrozik*) came seventh in the Creative Challenge. Table 8.3 shows the final leaderboard published by the Challenge organizers, which evaluates the teams on all of the Challenge Set.

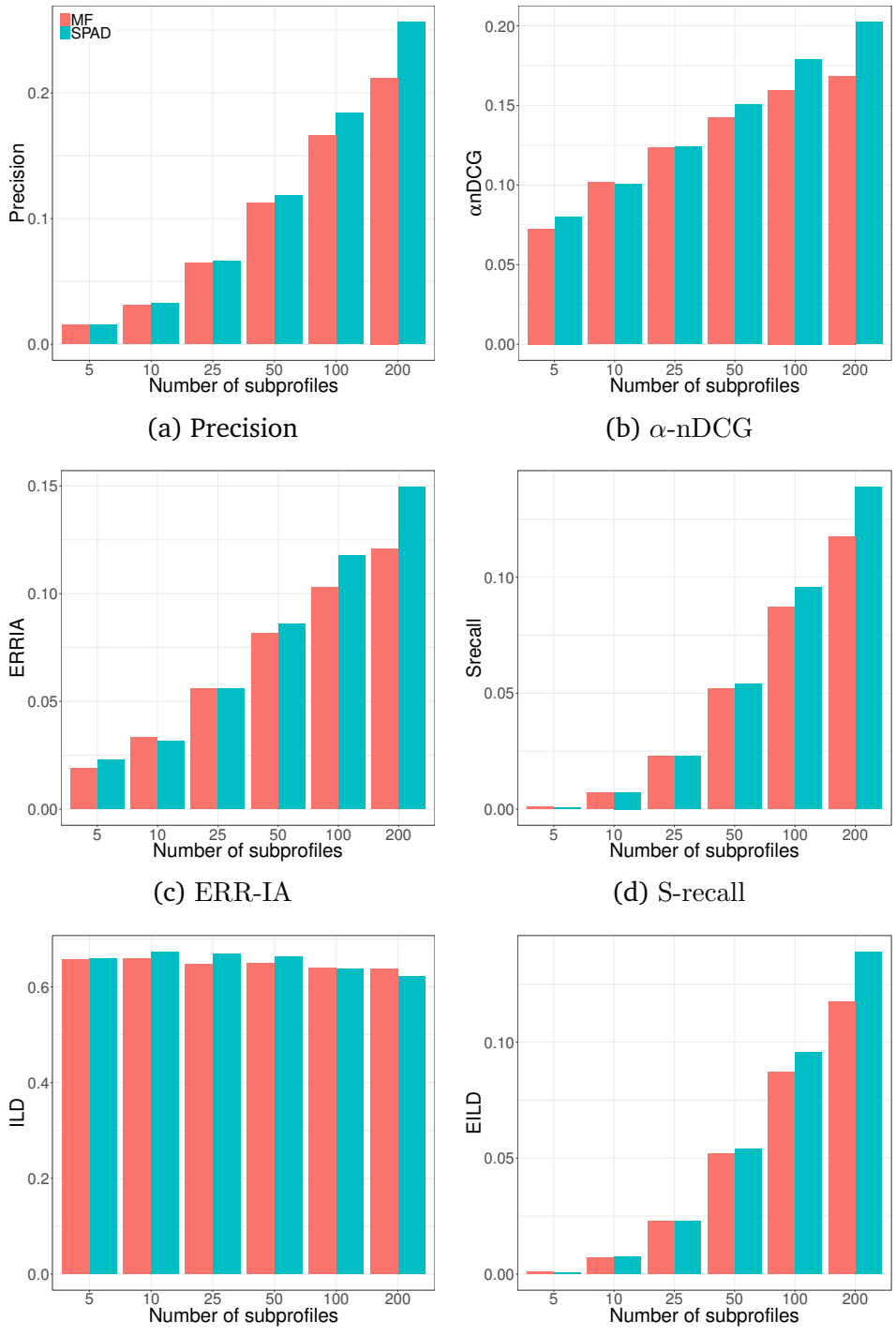



(a) Precision (b) $\alpha$-nDCG

(c) ERR-IA (d) S-recall

(e) ILD (f) EILD

Figure 8.4: Profile length plots.




(a) Precision (b) $\alpha$-nDCG

(c) ERR-IA (d) S-recall

(e) ILD (f) EILD

Figure 8.5: Number of subprofiles plots.

## 8.7 Conclusions

In this chapter, we presented our approach to the *ACM RecSys Challenge 2018*, in which the task is automatic playlist continuation (APC). For cold-start playlists,

Table 8.3: Creative Challenge leaderboard

| rank | team name | RPREC | RPREC rank | NDCG | NDCG rank | Clicks | Click rank | Borda |
|---|---|---|---|---|---|---|---|---|
| 1 | vl6 | 0.2234 | 1 | 0.3939 | 1 | 1.7845 | 1 | 90 |
| 2 | Creamy Fireflies | 0.2197 | 2 | 0.3846 | 2 | 1.9252 | 4 | 85 |
| 3 | KAENEN | 0.209 | 3 | 0.3746 | 3 | 2.0482 | 6 | 81 |
| 4 | cocoplaya | 0.2022 | 7 | 0.3656 | 6 | 1.8377 | 2 | 78 |
| 5 | BachPropagate | 0.2024 | 6 | 0.3659 | 5 | 2.0029 | 5 | 77 |
| 6 | Trailmix | 0.2059 | 4 | 0.3703 | 4 | 2.2589 | 9 | 76 |
| 7 | **teamrozik** | 0.2055 | 5 | 0.3609 | 7 | 2.1636 | 8 | 73 |

we used a popularity recommender. For the remainder of the playlists, we built a model using matrix factorization to generate sets of recommendations. But our contribution is that we re-ranked those recommendations to increase their diversity using SPAD. Using a validation set, we showed that SPAD re-ranking results in more accurate and mostly more diverse recommendations. Our analysis supports the claim that user-generated playlists do contain subprofiles corresponding to different interests or tastes, and trying to cover those subprofiles in the final set of recommendations produces more accurate recommendations.

Due to resource limitations (Section 8.4), we excluded a large number of songs from the dataset on which we built our model. Being unable to recommend those songs will have negatively affected our precision results. Furthermore, in the time that we allowed ourselves for working on the Challenge, we were not able to improve the method we use for cold-start playlists. Our method is quite rudimentary and easily improved, e.g. to use string similarity instead of exact matching of playlist titles.

While SPAD has proven successful for the APC task, it is designed for making recommendations in general. A music streaming service, such as Spotify, could apply it to whole user profiles (instead of playlists), in which case it should produce accurate and diverse sets of recommendations, covering different tastes and interests within the user's profile.

In the next chapter, we conclude the thesis by giving a brief overview of the contributions that we have made and offering ideas for future work.

# Chapter 9

# Conclusions & Future Work

In this thesis, we have addressed the question of *how best to produce a relevant but diverse set of recommendations*. First, we have proposed a form of intent-aware diversification approach, Subprofile-Aware Diversification (SPAD), that diversifies with respect to subprofiles that we mine from the user's profile, rather than with respect to item features.

Throughout this thesis, using different datasets, we found that SPAD and its variants almost always result in relevant and diverse recommendations, i.e. they suffer less from the relevance/diversity trade-off than other state-of-the-art approaches to recommendation diversification. This is true even when SPAD is applied to a completely different task, Automatic Playlist Continuation (Chapter 8). This is noteworthy, since it supports the generality of SPAD. Our findings addresses the main goal of this thesis: "producing relevant but diverse set of recommendations."

In this chapter, we present the conclusions derived from the main contributions of this thesis and we give some ideas for future work.

## 9.1 Conclusions

In this section, we summarize the main conclusions of this thesis.

### 9.1.1 Subprofile detection methods

In Chapter 4, we have presented and compared eight different ways to extract subprofiles from a user's profile. We grouped them into three: methods that use the nearest-neighbours of liked items; methods that use the explanations of top-$n$ recommendations; and methods that consider profile coverage. The methods that use the nearest-neighbours of liked items have several advantages over the others, and the empirical comparison on three datasets shows that one of these methods (designated NN-1) also most often performs better than the others in terms of recommendation accuracy and diversity. We have analyzed the subprofiles extracted by NN-1 for each of the three datasets with descriptive statistics and distribution graphs to better understand how extracted subprofiles differ from dataset to dataset.

### 9.1.2 SPAD & RSPAD

In Chapter 4, we have proposed a new approach to recommendation diversification that we refer to as SPAD, and its variant RSPAD. Unlike other approaches to recommendation diversification, which use item features to model user tastes and interests, in SPAD and RSPAD we have modeled user tastes and interests by using subprofiles of the user's profile. By doing so, we obtain a more fine-grained representation of the user tastes and interests than those obtained by item features.

In chapter 5, using subprofiles detected by NN-1, we have compared SPAD and RSPAD against several existing intent-aware diversification methods, ones that use item features as aspects. Empirical results on three datasets show that SPAD and RSPAD always result in the highest precision; they increase both precision and diversity in almost all settings; and they suffer even less from the relevance/diversity trade-off. These are noteworthy results because all existing intent-aware methods may have an advantage with respect to our measures of diversity since both the methods and the metrics uses item features.

### 9.1.3 Adaptation of diversity metrics to subprofiles

SPAD and its variants make no use of item features at all in their objective functions. Since all of the diversity metrics that we use in the experiments

are computed with respect to the item features, SPAD and its variants are at a disadvantage. In Chapter 5, we have adapted the diversity metrics $\alpha$-nDCG, ERR-IA, S-recall, EILD and ILD so that they use subprofiles instead of item features. Experimental results showing diversity metrics by item features and subprofiles side-by-side confirm that, indeed, SPAD and its variants are at a disadvantage. They also show the additional complexity of SPAD and its variants over other re-ranking algorithms in terms of diversity, because they always have the highest diversity values when measured by subprofiles, and they also usually improve the score of the diversity metrics, measured by item features, with respect to their baselines.

### 9.1.4 Community-Aware Diversification

In Chapter 6, we defined an alternative way to detect subprofiles, that we refer to as Community-Aware Diversification (CAD). CAD detects subprofiles using a user-user similarity approach, unlike SPAD, which uses an item-item similarity approach.

We find that, similar to SPAD, CAD produces recommendations that are usually the most accurate. We show that, again similar to SPAD, CAD is less prone to trading-offs accuracy for diversity. We also find that in some cases using CAD subprofiles performs better than using SPAD subprofiles; in other cases, the reverse is the case. We conclude that, for some domains, it is meaningful and useful to define subprofiles indirectly through a user's community (her nearest neighbours).

### 9.1.5 Intent-aware vs. calibrated recommendation

In Chapter 7, we have given a comparison between intent-aware and calibrated recommendation. Calibrated recommendation is strongly related to intent-aware recommendation in that both try to cover the user's interests to a certain extent.

We adapted the calibration metric, which uses item features, to give a variant that it is defined with respect to subprofiles instead. We have also defined a new instantiation of calibrated recommendation that aims to cover subprofiles, instead of item features.

We have found that intent-aware recommendation results in calibrated recommendations to a certain extent, and calibrated recommendation results in diverse recommendations to a certain extent. We have also found that calibrated recommendation, which aims to cover user interests with their exact proportions in the user's profile, harms relevancy (measured by precision) a lot. Besides, re-ranking approaches using subprofiles achieve the highest precision, and good calibration according to both calibration metrics, and usually achieve good diversity. They also suffer less from the relevance/diversity trade-off.

### 9.1.6 Submodularity of intent-aware approaches

By giving the proof that the main goal of intent-aware approaches, to find an optimal recommendation set $RL^*$ of size $N$ items, is NP-hard, and giving the proof of submodularity and monotonicity of the objective function, we have confirmed the proof of $(1 - \frac{1}{e})$ optimality guarantee of intent-aware approaches that use xQuAD's objective function (See Eq. 7.1), including SPAD and its variants.

### 9.1.7 Applying SPAD to Automatic Playlist Continuation

In this thesis, the main goal of SPAD and its variants is diversification. As we already mentioned, the main advantage of SPAD and variants is that they increase both precision and diversity. In chapter 8, we apply SPAD to the task of Automatic Playlist Continuation (APC), where precision, rather than diversity, is the main goal. We showed that SPAD re-ranking results in more accurate recommendations. We confirm that user-generated playlists do contain subprofiles corresponding to different interests or tastes, and trying to cover those subprofiles in the final set of recommendations produces more accurate and diverse recommendations.

## 9.2 Future Work

Our contributions in this thesis have introduced new ways to diversify recommendations. Mainly, we have focused on developing approaches that can produce relevant but diverse recommendations. We believe that we have covered

many angles on the main research topic. But our work is open to further extensions. In this section, we try to give a brief overview of some of these possible extensions.

### 9.2.1 Interpretability of SPAD

In the future, we could explore the interpretability of SPAD's recommendations: we might be able to explain SPAD's recommendations in terms of subprofiles. Since subprofiles are just sets of items, we can take inspiration from the work on item-based explanations (as used, for example, in amazon.com), which has been proven to produce effective explanations [BD14, BM05].

### 9.2.2 Perceived diversity

Nearly all work on recommender set diversification, including our own, is evaluated using offline experiments. But there is the question of how users *perceive* diversity [GDBJ10]. It would be valuable to conduct user studies to measure how users perceive the recommendations generated, e.g. by SPAD.

### 9.2.3 Using temporal aspects in SPAD

We could explore temporal aspects in detecting the subprofiles. In this thesis, subprofile detection treats the user profile and accordingly user tastes and interests as a static set. It does not consider the order, nor the actual timestamps, of the items in the user's profile. It might be beneficial to take this into account. For instance, we could take the idea of Evolutionary Collaborative Filtering [LZXY10], which assumes that older preferences of the users are generally less correlated with a user's current interests.

### 9.2.4 Using subprofiles in the cost functions

It might be possible to use subprofiles directly in the cost function that is optimized by a recommender system, instead of using them for re-ranking. Indicative of how this might work are set-oriented approaches, e.g. [SYCY13].

### 9.2.5 SPAD for task of recommendation to shared accounts

We could use the idea of using subprofiles for recommendation diversification for the task of recommending to shared accounts. We have used algorithms that were originally developed to recommending to shared accounts [VG15] as subprofile detection approaches. Our motivation to do so was the fact that DAMIB and DAMIB-COVER recommend to different subprofiles within a single account. Now that we have a wider range of subprofile detection methods (developed in this thesis), we could see which of these is most competitive in the task of recommending to shared accounts.

### 9.2.6 Further subprofile detection methods

In Chapter 4, we presented eight different subprofile detection methods. We also present alternative subprofile detection methods in chapter 6. We could develop yet further methods. For example, since subprofile detection can be viewed as a clustering task, perhaps existing clustering algorithms could be used to detect subprofiles as well. Notice that, usually, clustering algorithms *partition* a set of objects into clusters, hence each object is assigned to exactly one cluster (e.g. see $k$-means clustering [KMN$^+$02]). Our current subprofile detection methods allow subprofiles to overlap: an item can belong to more than one subprofile. It is possible that overlapping clustering algorithms would be a better choice to use to detect the subprofiles (e.g $k$-means for overlapping clustering [Cle08]).

### 9.2.7 SPAD for cold-start users

Intent-aware approaches diversify a set of recommended items produced by a baseline recommender with respect to the tastes or interests revealed by a user's profile. The user's tastes or interests are defined as a probability distribution over a set of *aspects*. In xQuAD, for example, aspects are item features, whereas in SPAD, aspects are the items in the user's profile, referred to as subprofiles. Intent-aware diversification approaches may fail to give diverse results to cold-start users, since their profile may not yet reflect all their tastes or interests. Only once their profiles have matured to include a wider range of tastes or interests will these techniques be effective.

Indeed, in Chapter 8, albeit for the APC task, we show that, usually, as the playlist length (user's profile size) and the number of subprofiles increases, the amount by which SPAD outperforms the baseline recommender also increases. For the smaller playlists (users with smaller profile size), the performance of SPAD and the baseline recommender are really close to each other.

These findings open a line of future work for cold-start users. In part, this may be about how to diversify recommendations for cold-start users, despite their small profiles. On the other hand, the solution may not be about directly about diversification. The solution may be getting a user to build her profile using active learning or extending her tastes or interests through serendipitous recommendations.

# Appendix A

# Hyperparameter Values

## A.1 Hyperparameter values for baseline recommenders

Table A.1 shows optimized hyperparameter values of each baseline recommendation algorithm on each dataset. Unless explicitly indicated, we use the values shown in Table A.1 to train the baseline recommenders in the following chapters.

## A.2 Hyperparameter values for different subprofile detection methods

Here, we show the hyperparameter values for different subprofile detection methods that are used in the experiments of Chapter 4. We select the values of $k_{ind}$, $k_{nn}$ and $k_{IB}$ from $V = \{10, 30, 50, \ldots, 290, 310\}$ and we select the value of $cp$ from the set $[0.5, 0.6, \ldots, 1.0]$. See Table A.2 for the hyperparameter values.

Table A.1: Hyperparameter values for each baseline recommender.

| | **MF** | | **pLSA** | **FMBPR** | | | |
|---|---|---|---|---|---|---|---|
| | d | $\alpha$ | d | d | lr | regW | regM |
| **ML1M** | 30 | 1 | 50 | 190 | 0.01 | 0.01 | 0.01 |
| **LFM** | 30 | 1 | 30 | 10 | 0.01 | 0.01 | 0.001 |
| **LT** | 330 | 1 | 270 | 270 | 0.01 | 0.01 | 0.01 |
| **FB** | 50 | 10 | 30 | 190 | 0.01 | 0.01 | 0.01 |
| **ML20M** | 40 | 6 | - | - | - | - | - |
| **TasteProfile** | 100 | 20 | - | - | - | - | - |

Table A.2: hyperparameters values for different subprofile detection approaches for all three datasets and all three baselines. These are optimized using validation sets.

| | pLSA | | | | | | MF | | | | | | FMBPR | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ |
| ML1M | | | | | | | | | | | | | | | | | | |
| NN-1 (for SPAD) | 1.0 | - | 10 | - | 50 | - | 0.4 | - | 10 | - | 30 | - | 0.4 | - | 10 | - | 10 | - |
| NN-1 (for RSPAD) | 1.0 | - | 10 | - | 30 | - | 0.7 | - | 10 | - | 50 | - | 0.4 | - | 10 | - | 10 | - |
| NN-2 | 1.0 | - | 10 | - | 30 | - | 0.5 | - | 10 | - | 30 | - | 0.4 | - | 10 | - | 10 | - |
| IB+ | 1.0 | 50 | - | 10 | 50 | - | 0.6 | 50 | - | 10 | 50 | - | 0.4 | 30 | - | 10 | 10 | - |
| DAMIB | 1.0 | 70 | - | 10 | 30 | - | 0.6 | 70 | - | 10 | 50 | - | 0.4 | 30 | - | 10 | 10 | - |
| DAMIB-COVER | 0.6 | 90 | - | 10 | 30 | - | 0.6 | 110 | - | 10 | 50 | - | 0.4 | 30 | - | 10 | 10 | - |
| IB+$_{\text{cp}}$ | 0.7 | - | - | 10 | 10 | 1.0 | 0.4 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{\text{cp}}$ | 1.0 | - | - | 10 | 30 | 0.9 | 0.5 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 0.9 |
| DAMIB-COVER$_{\text{cp}}$ | 1.0 | - | - | 10 | 30 | 0.9 | 0.5 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 0.9 |
| LFM | | | | | | | | | | | | | | | | | | |
| NN-1 (for SPAD) | 0.6 | - | 30 | - | 10 | - | 0.2 | - | 50 | - | 10 | - | 0.4 | - | 10 | - | 10 | - |
| NN-1 (for RSPAD) | 0.6 | - | 10 | - | 10 | - | 0.4 | - | 70 | - | 10 | - | 0.5 | - | 10 | - | 10 | - |
| NN-2 | 0.5 | - | 30 | - | 10 | - | 0.2 | - | 50 | - | 10 | - | 0.5 | - | 30 | - | 10 | - |
| IB+ | 0.5 | 110 | - | 30 | 10 | - | 0.3 | 110 | - | 10 | 30 | - | 0.5 | 110 | - | 30 | 10 | - |
| DAMIB | 0.5 | 110 | - | 30 | 10 | - | 0.2 | 70 | - | 50 | 10 | - | 0.5 | 110 | - | 30 | 10 | - |
| DAMIB-COVER | 0.6 | 110 | - | 70 | 10 | - | 0.3 | 50 | - | 10 | 30 | - | 0.6 | 110 | - | 70 | 10 | - |
| IB+$_{\text{cp}}$ | 0.5 | - | - | 10 | 30 | 1.0 | 0.2 | - | - | 50 | 10 | 0.5 | 0.5 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{\text{cp}}$ | 0.6 | - | - | 50 | 10 | 1.0 | 0.2 | - | - | 70 | 10 | 1.0 | 0.6 | - | - | 50 | 10 | 1.0 |
| DAMIB-COVER$_{\text{cp}}$ | 0.6 | - | - | 50 | 10 | 1.0 | 0.2 | - | - | 70 | 10 | 1.0 | 0.6 | - | - | 50 | 10 | 1.0 |
| LT | | | | | | | | | | | | | | | | | | |
| NN-1 (for SPAD) | 0.6 | - | 10 | - | 10 | - | 0.4 | - | 30 | - | 10 | - | 0.5 | - | 10 | - | 10 | - |
| NN-1 (for RSPAD) | 0.7 | - | 10 | - | 10 | - | 0.4 | - | 10 | - | 10 | - | 0.6 | - | 10 | - | 10 | - |
| NN-2 | 0.6 | - | 10 | - | 10 | - | 0.4 | - | 10 | - | 10 | - | 0.5 | - | 10 | - | 10 | - |
| IB+ | 0.5 | 10 | - | 10 | 10 | - | 0.5 | 10 | - | 10 | 10 | - | 0.6 | 50 | - | 50 | 10 | - |
| DAMIB | 0.5 | 30 | - | 10 | 10 | - | 0.4 | 30 | - | 50 | 10 | - | 0.6 | 30 | - | 10 | 10 | - |
| DAMIB-COVER | 0.5 | 30 | - | 10 | 10 | - | 0.4 | 10 | - | 50 | 10 | - | 0.5 | 30 | - | 10 | 10 | - |
| IB+$_{\text{cp}}$ | 0.5 | - | - | 10 | 10 | 1.0 | 0.4 | - | - | 10 | 10 | 1.0 | 0.5 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{\text{cp}}$ | 0.6 | - | - | 30 | 10 | 1.0 | 0.4 | - | - | 30 | 10 | 0.8 | 0.5 | - | - | 10 | 10 | 0.9 |
| DAMIB-COVER$_{\text{cp}}$ | 0.6 | - | - | 30 | 10 | 0.5 | 0.5 | - | - | 10 | 10 | 1.0 | 0.5 | - | - | 10 | 10 | 1.0 |

## A.3 Hyperparameter values for different diversification algorithms

This section presents the hyperparameter values we use in Chapters 5, 6 and 7.

We select the values of $k_{ind}$, $k_{nn}$ and $k_{UB}$ from the set $V$ that we show in the previous section. However, for the FB dataset, we found we needed a greater range of candidate values and so we tested with values up to 1500. For CAD's additional hyperparameter $q$, used in relaxed intersection, we select its values from the set $[0.1, 0.2, \ldots, 1.0]$.

The values selected for the MovieLens dataset are:

- pLSA: $k_{nn} = 10$, $k_{ind} = 50$ for SPAD; $k_{UB} = 150$, $k_{ind} = 10$, $q = 0.7$ for CAD.
- MF: $k_{nn} = 10$, $k_{ind} = 30$ for SPAD; $k_{UB} = 170$, $k_{ind} = 10$, $q = 0.7$ for CAD.
- FMBPR: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 150$, $k_{ind} = 10$, $q = 0.8$ for CAD.

The values selected for the LastFM dataset are:

- pLSA: $k_{nn} = 30$, $k_{ind} = 10$ for SPAD; $k_{UB} = 130$, $k_{ind} = 10$, $q = 0.9$ for CAD.
- MF: $k_{nn} = 50$, $k_{ind} = 10$ for SPAD; $k_{UB} = 130$, $k_{ind} = 10$, $q = 1.0$ for CAD.
- FMBPR: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 90$, $k_{ind} = 10$, $q = 0.8$ for CAD.

The values selected for the LibraryThing dataset are:

- pLSA: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 150$, $k_{ind} = 10$, $q = 1.0$ for CAD.
- MF: $k_{nn} = 30$, $k_{ind} = 10$ for SPAD; $k_{UB} = 170$, $k_{ind} = 10$, $q = 0.8$ for CAD.
- FMBPR: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 150$, $k_{ind} = 10$, $q = 1.0$ for CAD.

The values selected for the Facebook dataset are:

- pLSA: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 1500$, $k_{ind} = 10$, $q = 0.8$ for CAD.
- MF: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 1500$, $k_{ind} = 10$, $q = 0.8$ for CAD.
- FMBPR: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD; $k_{UB} = 1500$, $k_{ind} = 10$, $q = 0.8$ for CAD.

The values selected for the ML20M and TasteProfile datasets are:

- MF: $k_{nn} = 10$, $k_{ind} = 10$ for SPAD and $\text{CR}_{\mathcal{S}}$.

# References


[ABC+18] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio Ferrari Dacrema. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 4:1–4:6. ACM, 2018.

[ABDN+17] Vito W Anelli, Vito Bellini, Tommaso Di Noia, Wanda La Bruna, Paolo Tomeo, and Eugenio Di Sciascio. An Analysis on Time- and Session-aware Diversification in Recommender Systems. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 270–274. ACM, 2017.

[ABM19] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555*, 2019.

[AGHI09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 5–14, 2009.

[Aio13] Fabio Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 273–280. ACM, 2013.

[AK08] Gediminas Adomavicius and Young Ok Kwon. Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach. In *Proceedings of the 2008 Workshop on Information Technologies and Systems*, pages 151–156, 2008.

[AK09] Gediminas Adomavicius and YoungOk Kwon. Toward more diverse recommendations: Item re-ranking methods for recommender systems. In *Proceedings of the 19th Workshop on Information Technologies and Systems*, pages 79–84, 2009.

[AK12] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2012.

[AKBW15] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. Optimal greedy diversity for recommendation. In *Proceedings of the 24th International Conference on Artificial Inteeligence*, pages 1742–1748. AAAI Press, 2015.

[Bay15] Immanuel Bayer. Fastfm: a library for factorization machines. *arXiv preprint arXiv:1505.00641*, 2015.

[BD14] Derek Bridge and Kevin Dunleavy. If you liked Herlocker et al.'s explanations paper, then you might like this paper too. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, pages 22–27. ACM, 2014.

[BM05] Mustafa Bilgic and Raymond J Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization, a Workshop on the Next Stage of Recommender Systems Research at the International Conference on Intelligent USer Interfaces*, pages 153–158, 2005.

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

[BP06] Claudio Baccigalupo and Enric Plaza. Case-based sequential ordering of songs for playlist recommendation. In *Proceedings of the 8th European Conference on Case-Based Reasoning*, pages 286–300. Springer, 2006.

[CdVR08] Maarten Clements, Arjen P de Vries, and Marcel JT Reinders. Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In *Workshop on Exploiting Se-*

*mantic Annotations in Information Retrieval*, pages 18–24. ACM, 2008.

[CG98] Jaime Carbonell and Jade Goldstein. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 335–336. ACM, 1998.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[CHV15] Pablo Castells, Neil J Hurley, and Saul Vargas. Novelty and Diversity in Recommender Systems. In F Ricci and Others, editors, *Recommender Systems Handbook (2nd edition)*, pages 881–918. Springer, 2015.

[CKC+08] Charles L A Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 659–666. ACM, 2008.

[Cle08] Guillaume Cleuziou. An extended version of the k-means method for overlapping clustering. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[CLSZ18] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. RecSys Challenge 2018: Automatic Music Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 527–528. ACM, 2018.

[CWM+17] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. Learning to Recommend Accurate and Diverse Items. In *Proceedings of the 26th International Conference on World Wide Web*, pages 183–192. ACM, 2017.

[CZZ18] Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recomme-

ndation diversity. In *Advances in Neural Information Processing Systems*, pages 5623–5634, 2018.

[DK04] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[DRTD17] Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive Multi-attribute Diversity for Recommender Systems. *Information Sciences*, 382(C):234–253, 2017.

[EMB17] Farzad Eskandanian, Bamshad Mobasher, and Robin Burke. A Clustering Approach for Personalizing Diversity in Collaborative Recommender Systems. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 280–284. ACM, 2017.

[FBY+18] Andres Ferraro, Dmitry Bogdanov, Jisang Yoon, KwangSeob Kim, and Xavier Serra. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 2:1–2:5. ACM, 2018.

[FTTC+16] Ignacio Fernández-Tobias, Paolo Tomeo, Iván Cantador, Tommaso Di Noia, and Eugenio Di Sciascio. Accuracy and Diversity in Cross-domain Recommendations for Cold-start Users with Positive-only Feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 119–122. ACM, 2016.

[GDBJ10] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. User-perceived recommendation quality-factoring in the user interface. In *Proceedings of the ACM Conference on Recommender Systems Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces*, pages 22–25. ACM, 2010.

[HK15] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015.

[HKTR04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender

systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272. IEEE, 2008.

[Hoc97] Dorit S. Hochba. Approximation algorithms for np-hard problems. *SIGACT News*, 28(2):94–143, June 1997.

[Hof04] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[Hur13] Neil J Hurley. Personalised ranking with diversity. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 379–382. ACM, 2013.

[KB06] John Paul Kelly and Derek Bridge. Enhancing the diversity of conversational collaborative recommendations: a comparison. *Artificial Intelligence Review*, 25(1-2):79–95, 2006.

[KB11] Yehuda Koren and Robert Bell. Advances in Collaborative Filtering. In F Ricci and Others, editors, *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[KB16] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems*, 7(1):2:1–2:42, 2016.

[KKT+15] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A Konstan, and Paul Schrater. I like to explore sometimes: adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 19–26. ACM, 2015.

[KKZV18] Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. Investigating serendipity in recommender systems based on real user feedback. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1341–1350. ACM, 2018.

[KMI18] Surya Kallumadi, Bhaskar Mitra, and Tereza Iofciu. A line in the sand: Recommendation or ad-hoc retrieval? overview of recsys challenge 2018 submission by team bachpropagate. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 7:1–7:6. ACM, 2018.

[KMN+02] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 7:881–892, 2002.

[Kul18] Maciej Kula. Mixture-of-tastes Models for Representing Users with Diverse Interests. In *Proceedings of Workshop on Learning from User Interactions*, pages 1:1–1:5. ACM, 2018.

[LACB16] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 59–66. ACM, 2016.

[LBM11] Jin Ha Lee, Bobby Bare, and Gary Meek. How Similar Is Too Similar?: Exploring Users' Perceptions of Similarity in Playlist Evaluation. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 109–114. Citeseer, 2011.

[LHCA10] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2010.

[LKLJ18] Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. Effective nearest-neighbor music recommendations. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 3:1–3:6. ACM, 2018.

[LRD14] Shangsong Liang, Zhaochun Ren, and Maarten De Rijke. Personalized Search Result Diversification via Structured Learning. In *Proceedings of the 20th ACM SIGKDD International Conference*

*on Knowledge Discovery and Data Mining*, pages 751–760. ACM, 2014.

[LZXY10] Nathan N Liu, Min Zhao, Evan Xiang, and Qiang Yang. Online evolutionary collaborative filtering. In *Proceedings of the 4th ACM conference on Recommender systems*, pages 95–102. ACM, 2010.

[MRK06] Sean M McNee, John Riedl, and Joseph A Konstan. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *Proceedings of the Extended Abstracts on Human Factors in Computing Systems*, pages 1097–1101. ACM, 2006.

[NCGV18] Houssam Nassif, Kemal Oral Cansizlar, Mitchell Goodman, and SVN Vishwanathan. Diversifying music recommendations. *arXiv preprint arXiv:1810.01482*, 2018.

[NWF78] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[PUG16] Shameem A Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 15–22. ACM, 2016.

[PZT10] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 71–78. ACM, 2010.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[RRS15] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[SM01] Barry Smyth and Paul McClave. Similarity vs. diversity. In *Proceedings of the International Conference on Case-Based Reasoning*, pages 347–361. Springer, 2001.

[SMO10] Rodrygo L T Santos, Craig Macdonald, and Iadh Ounis. Exploiting Query Reformulations for Web Search Result Diversification. In *Proceedings of the 19th International Conference on World Wide Web*, pages 881–890. ACM, 2010.

[Ste18] Harald Steck. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 154–162. ACM, 2018.

[SYCY13] Ruilong Su, Li'Ang Yin, Kailong Chen, and Yong Yu. Set-oriented personalized ranking for diversified top-n recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 415–418. ACM, 2013.

[SZC+18] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

[SZW+12] Yue Shi, Xiaoxue Zhao, Jun Wang, Martha Larson, and Alan Hanjalic. Adaptive Diversification of Recommendation Results via Latent Factor Portfolio. In *Proceedings of the 35th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 175–184. ACM, 2012.

[TB17] Chun-Hua Tsai and Peter Brusilovsky. Leveraging Interfaces to Improve Recommendation Diversity. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 65–70. ACM, 2017.

[TB18] Chun-Hua Tsai and Peter Brusilovsky. Beyond the Ranked List: User-Driven Exploration and Diversification of Social Recommendation. In *23rd International Conference on Intelligent User Interfaces*, pages 239–250. ACM, 2018.

[TFTDNC16] Paolo Tomeo, Ignacio Fernández-Tobías, Tommaso Di Noia, and Iván Cantador. Exploiting linked open data in cold-start recommendations with positive-only feedback. In *Proceedings of the 4th*

*Spanish Conference on Information Retrieval*, pages 11:1–11:8. ACM, 2016.

[TLL17] Nava Tintarev, Christoph Lofi, and Cynthia Liem. Sequences of Diverse Song Recommendations: An exploratory study in a commercial system. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 391–392. ACM, 2017.

[TNH+16] Choon Hui Teo, Houssam Nassif, Daniel Hill, Sriram Srinivasan, Mitchell Goodman, Vijai Mohan, and SVN Vishwanathan. Adaptive, personalized diversity for visual discovery. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 35–38. ACM, 2016.

[Var15] Saúl Vargas Sandoval. *Novelty and Diversity Evaluation and Enhancement in Recommender Systems*. PhD thesis, Universidad Autónoma de Madrid, Spain, 2015.

[VC11] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender systems*, pages 109–116. ACM, 2011.

[VC12] David Vallet and Pablo Castells. Personalized Diversification of Search Results. In *Proceedings of the 35th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 841–850. ACM, 2012.

[VC13] Saúl Vargas and Pablo Castells. Exploiting the diversity of user preferences for recommendation. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 129–136. ACM, 2013.

[VCV11] Saúl Vargas, Pablo Castells, and David Vallet. Intent-oriented Diversity in Recommender Systems. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1211–1212. ACM, 2011.

[VCV12] Saúl Vargas, Pablo Castells, and David Vallet. Explicit Relevance Models in Intent-oriented Information Retrieval Diversification. In *Proceedings of the 35th ACM SIGIR International Conference on*

*Research and Development in Information Retrieval*, pages 75–84. ACM, 2012.

[VG15] Koen Verstrepen and Bart Goethals. Top-N Recommendation for Shared Accounts. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 59–66. ACM, 2015.

[VRC+18] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 9:1–9:6. ACM, 2018.

[WCZ18] Wen Wu, Li Chen, and Yu Zhao. Personalizing recommendation diversity based on user personality. *User Modeling and User-Adapted Interaction*, 28(3):237–276, 2018.

[WGK16] Martijn C. Willemsen, Mark P. Graus, and Bart P. Knijnenburg. Understanding the Role of Latent Feature Diversification on Choice Difficulty and Satisfaction. *User Modeling and User-Adapted Interaction*, 26(4):347–389, 2016.

[WH16] Jacek Wasilewski and Neil Hurley. Intent-Aware Diversification Using a Constrained PLSA. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 39–42. ACM, 2016.

[WH17] Jacek Wasilewski and Neil Hurley. Personalised Diversification Using Intent-Aware Portfolio. In *Proceedings of the 25th ACM Conference on User Modeling, Adaptation and Personalization*, pages 71–76. ACM, 2017.

[WH18] Jacek Wasilewski and Neil Hurley. Intent-aware item-based collaborative filtering for personalised diversification. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 81–89. ACM, 2018.

[WRB+18] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2165–2173. ACM, 2018.

[ZCL03] Cheng Xiang Zhai, William W Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 10–17. ACM, 2003.

[ZH08] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 123–130. ACM, 2008.

[ZH09] Mi Zhang and Neil Hurley. Novel item recommendation by user profile partitioning. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 508–515. IEEE Computer Society, 2009.

[ZMKL05] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, pages 22–32. ACM, 2005.

[ZSLC18] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. *arXiv preprint arXiv:1810.01520*, 2018.