**User Modeling and User-Adapted Interaction manuscript No.**
(will be inserted by the editor)

# Subprofile-Aware Diversification of Recommendations

**Mesut Kaya · Derek Bridge**

Received: date / Accepted: date

**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 the average 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 exhibts increases in both relevance and diversity.

In this paper, we present a new form of intent-aware diversification, which we call SPAD (Subprofile-Aware Diversification), and a variant called RSPAD (Relevance-based SPAD). In SPAD, the aspects are not item features; they are *subprofiles* of the user's profile. We present and compare a number of different ways to extract subprofiles from a user's profile. None of them is defined in

Mesut Kaya · Derek Bridge
Insight Centre for Data Analytics, School of Computer Science & Information Technology, University College Cork, Ireland
E-mail: {mesut.kaya|derek.bridge}@insight-centre.org

terms of item features. Therefore, SPAD is useful even in domains where item features are not available or are of low quailty.

On three pre-collected datasets from three different domains (movies, music artists and books), we compare SPAD and RSPAD to intent-aware methods in which aspects are item features. We find on these datasets that SPAD and RSPAD suffer even less from the relevance/diversity trade-off: across all three datasets, they increase both relevance and diversity for even more configurations than other approaches to diversification. Moreover, we find that SPAD and RSPAD are the most accurate systems across all three datasets.

**Keywords** Recommender systems · diversity · intent-aware diversification · subprofiles

## 1 Introduction

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 help the users of these platforms to discover new, interesting content. 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 (Eskandanian et al., 2017), and it may narrow, rather than broaden, a user's horizons (Cheng et al., 2017). In many domains, recommendations must be novel to the user or serendipitous, and a set of recommendations must be diverse (McNee et al., 2006). It is diversity that is the focus of this paper.[1]

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, and even increase satisfaction

[1] In this paper, we use the word "diversity" exclusively to refer to a property of a set of recommendations. Elsewhere, "diversity" (or sometimes "sales diversity" or "aggregate diversity") is a property of a recommender system as a whole, referring to the extent to which a system's recommendations cover the item catalog. For a survey of concepts and definitions, see (Kaminskas and Bridge, 2016).

with the chosen item, e.g. (Willemsen et al., 2016), especially when visual interfaces are designed to highlight the diversity of the recommended items, e.g. (Tsai and Brusilovsky, 2017, 2018).

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 (Adomavicius and Kwon, 2008). Past research has considered how to increase the level of diversity at the expense of negligible accuracy loss (Adomavicius and Kwon, 2009). However, as we will show in this paper, 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 (Carbonell and Goldstein, 1998). 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* (Santos et al., 2010; Vargas et al., 2012). The idea is that, to satisfy a user, a result set must cover her intention. 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. (2011) 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). 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 the degree that these are reflected in her profile, i.e. if there is more pop music in her profile then, other things being equal, there should be more pop music in the set of recommendations.

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 describes movies using just 18 genres (Harper and Konstan, 2015). In domains where tastes and interests are complex, subtle and highly subjective, such as movies and music, 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 intents of the users (Kula, 2017). 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 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. SPAD has two advantages over existing intent-aware diversification methods. First, it can be used in domains where item features are not available or are unreliable. Second, as we show in our experimental results, across multiple datasets and across almost all settings, SPAD improves both accuracy and diversity. It suffers 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.

In this paper, we make the following contributions:

- We define the SPAD and RSPAD intent-aware diversification methods, relating them to existing published work.
- We define eight methods for detecting subprofiles. We can group them into three types of methods: (i) two methods (NN-1 and NN-2) that use the

nearest-neighbours of liked items; (ii) three methods (IB+, DAMIB and DAMIB-COVER) that use the explanations of top-$n$ recommendations; and (iii) three methods ($\text{IB+}_{\text{cp}}$, $\text{DAMIB}_{\text{cp}}$ and $\text{DAMIB-COVER}_{\text{cp}}$) that consider profile coverage.
– 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 NN-1) is also most often the best in terms of recommendation accuracy and diversity.
– For each of the three datasets, 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.
– 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. On our three datasets, we show that SPAD and RSPAD always result in the highest precision. We also show that SPAD and RSPAD increase both precision and diversity (measured by $\alpha$-nDCG and other metrics) in almost all settings. By contrast, the other methods often increase both, but sometimes increase diversity at the expense of precision. SPAD and RSPAD are the two methods that most often defy the relevance/diversity trade-off.

We have presented two of the eight subprofile detection methods in previously published work (Kaya and Bridge, 2017, 2018a), along with a subset of their empirical results on just two of the datasets. Hence, by defining six other subprofile detection methods, comparing the methods with each other, and comparing the best method with other diversification techniques using more metrics and one more dataset, this paper is a considerably more comprehensive presentation of our research in this area. We have also referred to one of the subprofile detection methods in a publication that describes a different task (automatic playlist completion) on a dataset that is completely different from any of those used here (Kaya and Bridge, 2018b).

Section 2 reviews the state-of-the-art in recommendation diversification. Section 3 explains how SPAD and RSPAD work. Section 4 presents the details of the eight different subprofile detection methods. Section 5 describes the experimental methodology and datasets that we have used. Section 6 present the results of the experiments. The paper concludes with a discussion and ideas for future work in Section 7.

## 2 Related Work

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 doc-

**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 $RS$
1: $RL \leftarrow [\,]$
2: while $|RS| > 0$ do
3:     $i^* \leftarrow \arg\max_{i \in RS} f_{obj}(i, RL)$
4:     delete $i^*$ from $RS$
5:     append $i^*$ to the end of $RL$
6: return $RL$
```

uments is diverse, i.e. that they are different from one another (Carbonell and Goldstein, 1998; Clarke et al., 2008). 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 (Agrawal et al., 2009). In recommender systems, uncertainty is caused by small user profiles (e.g. for cold-start users) and by profiles that span different tastes. While a recommender should recommend items that it predicts are relevant to the user, its chances of recommending items that satisfy the user on a given occasion can be increased by recommending a set of diverse items (Castells et al., 2015; Kaminskas and Bridge, 2016).

There is some work in which diversity is explicitly a part of the objective function that the recommender system seeks to optimize when generating recommendations, e.g. (Hurley, 2013; Su et al., 2013; Cheng et al., 2017). There is also work in which diversity is formulated as a subgraph selection problem (Antikacioglu and Ravi, 2017). However, the dominant approach to diversification is greedy re-ranking.

## 2.1 Greedy re-ranking

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. $f_{obj}$ is usually defined as a linear combination of the item's relevance score and the contribution item $i$ makes to the diversity of $RL$, $\text{div}(i, RL)$, the trade-off between the two being controlled by a parameter $\lambda$ ($0 \leq \lambda \leq 1$):

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

In Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998), for example, $\text{div}(i, RL)$ is the maximum of the distances between $i$ and the items already selected:

$$\text{div}(i, RL) = \max_{j \in RL} \text{dist}(i, j) \tag{2}$$

The distance between items $i$ and $j$, $\text{dist}(i, j)$, can be calculated from meta-data (such as movie genres or book categories (Smyth and McClave, 2001; Ziegler et al., 2005)) or from item ratings data (Kelly and Bridge, 2006). Alternatively, $\text{div}(i, RL)$ can be computed as the average (or sum) of the all-pairs intra-list distances, $\text{div}(i, RL) = \text{ILD}(\{i\} \cup RL)$, where the ILD of any list of items $L$ is given by:

$$\text{ILD}(L) = \frac{2}{|L|(|L| - 1)} \sum_{i \in L} \sum_{j \in L, j \neq i} \text{dist}(i, j) \tag{3}$$

The final recommendation comprises the top-$N$ members of the re-ranked list, $RL$, where $N < |RL|$. Re-ranking using the ILD 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. More recent 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.

## 2.2 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 (Agrawal et al., 2009; Santos et al., 2010; Vargas et al., 2011, 2012).

Intent-aware methods for recommendation diversification (Vargas et al., 2011, 2012; Wasilewski and Hurley, 2016) take inspiration from the work done in IR. 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 (Koren and Bell, 2011).

In our review of these methods, let $I_u$ be items that are in the user's profile.

### *2.2.1 xQuAD*

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 Vargas's adaptation to recommender systems (Vargas Sandoval, 2015) of Santos et al.'s Query

Aspect Diversification framework (xQuAD) (Santos et al., 2010), diversification can be achieved by re-ranking a conventional recommender's recommendation set as per Algorithm 1 and Eq. 1 but with $\text{div}(i, RL) = \text{nov}_{\text{xQuAD}}(i, RL)$ defined as:

$$\text{nov}_{\text{xQuAD}}(i, RL) = \sum_{a \in A} [p(a|u)p(i|u, a) \prod_{j \in RL} (1 - p(j|u, a))] \tag{4}$$

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 above, let $I_u$ denote the items that are in the user's profile. Then $p(f|u)$ can be estimated as:

$$p(f|u) = \frac{|\{i \in I_u : f \in \mathcal{F}_i\}|}{\sum_{f' \in \mathcal{F}} |\{i \in I_u : f' \in \mathcal{F}_i\}|} \tag{5}$$

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

$$p(i|u, f) = \frac{\mathbb{1}(i, f)s(u, i)}{\sum_{j \in RS} \mathbb{1}(j, f)s(u, j)} \tag{6}$$

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

xQuAD is a generalization of the IA-Select method (Agrawal et al., 2009). The latter does not consider relevance, hence they are equivalent in the case when $\lambda = 1$ in Eq. 1.

### *2.2.2 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, $\text{div}(i, RL) = \text{nov}_{\text{RxQuAD}}(i, RL)$ is defined as:

$$\text{nov}_{\text{RxQuAD}}(i, RL) = \sum_{a \in A} [p(a|u)p(rel|i, u, a) \prod_{j \in RL} (1 - p(rel|j, u, a)p(stop|rel))] \tag{7}$$

$p(rel|i, u, a)$ 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 (Vargas et al., 2012). $p(stop|rel)$ 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 $\alpha$ (Vargas Sandoval, 2015).

### *2.2.3 SxQuAD and SRxQuAD*

Vargas and Castells (2013) define another two forms of intent-aware diversification, which they refer to as SxQuAD and SRxQuAD, this time based on combining sets of recommendations. 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. 4 or Eq. 7, which in turn is used for greedy re-ranking using Eq. 1, from which a top-$N$ can finally be recommended.

For SxQuAD, $p(i|u,a)$ in Eq. 4 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 $f$:

$$\mathrm{nov}_{\mathrm{SxQuAD}}(i, RL) = \sum_{f \in \mathcal{F}} [p(f|u)p(i|u_f) \prod_{j \in RL} (1 - p(j|u_f))] \tag{8}$$

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

$$p(i|u_f) = \frac{s(u_f, i)}{\sum_{j \in R_{u_f}} s(u_f, j)} \tag{9}$$

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. 7 is replaced by $p(rel|i,u_f)$.

There is an apparent similarity between our new approaches to diversification, SPAD and RSPAD, 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). 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 subprofile as an aspect and modeling the user's interests as a probability distribution over the aspects.

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

The advantage in intent-aware approaches, such as xQuAD and its variations, 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 (Wasilewski and Hurley, 2016). 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) (Wasilewski and Hurley, 2016). The learning is done by a constrained pLSA model (Hofmann, 2004). 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 (Wasilewski and Hurley, 2017).

Recently, Anelli et al. have incorporated temporal aspects into an intent-aware diversification framework based on xQuAD (Anelli et al., 2017). Noting that a user's intent can change during an interaction, they propose versions of Eq. 5 that include a temporal decay function on the one hand and that handle sessions on the other hand.

The advantage of intent-aware approaches to diversification 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.

## 2.4 Personalized diversification

There has been an amount of recent research on personalizing the level of diversification in IR, e.g. (Vallet and Castells, 2012; Liang et al., 2014), and in recommender systems, e.g. (Shi et al., 2012; Di Noia et al., 2017; Puthiya Parambath et al., 2016; Zhang and Hurley, 2008; Vargas and Castells, 2013; Wasilewski and Hurley, 2017).

In IR, Vallet and Castells (2012) 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. Liang et al. (2014) take a very different approach, treating diversification as a form of supervised learning using document terms and latent topics as features. 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 (Shi et al., 2012). 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 (Di Noia et al., 2017): 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 she has low or high entropy and whether she has a small or large profile. They then re-rank recommendation sets using modified versions of both Eq. 3 and Eq. 4. The modifications introduce quadrant-specific weights, thereby controlling the degree of diversification in a personalized way.

Puthiya Parambath et al. (2016) do not use greedy re-ranking. 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.

Eskandanian et al. (2017) use a clustering approach for personalizing diversity. 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 independently on the cluster of users. They measure the degree of diversity for users based on item categories (such as genres in movies).

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 (Zhang and Hurley, 2008). In other work, the same authors cluster the items in a user's profile, using rating similarities (Zhang and Hurley, 2009). 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.

## 3 Subprofile Aware Diversity

In this section, 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. In the work on intent-aware diversification that we described earlier (Section 2), aspects were often based on explicit item features, e.g. movie genres. In SPAD, by contrast, aspects are user *subprofiles*, i.e. subsets of the items that the user likes.

Let $I$ be the set of all items, $U$ be the set of all users, and $R$ be a $|U| \times |I|$ matrix, where $r_{ui} \in R$ is $u$'s rating of $i$ or $r_{ui} = \perp$ if $u$ has not rated $i$; and let $I_u$ be the set of items that user $u$ has rated, i.e. $I_u = \{i \in I : r_{ui} \neq \perp\}$. Subprofiles 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.), i.e. $I_u^+ = I_u = \{i \in I : r_{ui} \neq \perp\}$. In the case of a recommender system that uses numeric explicit ratings $r_{ui}$ (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, we use $I_u^+ = \{i \in I_u : r_{ui} \geq 4\}$ for 1–5 stars ratings.

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 interests and tastes 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 Section 4. 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. 1 with $\text{div}(i, RL) = \text{nov}_{\text{xQuAD}}(i, RL)$ (Eq. 4).

## 3.1 SPAD

What differs between SPAD and other forms of intent-aware diversification is the computation of the probabilities used in Eq. 4. 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 $S \in \mathcal{S}_u^*$.

Analogously to Eq. 5, $p(S|u)$ can be estimated as:

$$p(S|u) = \frac{|S|}{\sum_{S' \in \mathcal{S}_u^*} |S'|} \tag{10}$$

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

$$p(i|u,S) = \frac{\mathbb{1}(i,S)s(u,i)}{\sum_{j \in RS} \mathbb{1}(j,S)s(u,j)} \tag{11}$$

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:

$$\mathbb{1}(i,S) = \begin{cases} 1 & \text{if } i \in \bigcup_{j \in S} \text{KNN}(j) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $\text{KNN}(j)$ 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. 12, where we use similarity more directly but, in preliminary experiments, it performed less well. We may explore it further in future.)

**Table 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 $u$'s liked items | IB+$_{\text{cp}}$ |
| | DAMIB$_{\text{cp}}$ |
| | DAMIB-COVER$_{\text{cp}}$ |

### 3.2 RSPAD, SSPAD and SRSPAD

Analogously to the relationship between xQuAD and RxQuAD (Vargas et al., 2012), we can define RSPAD, which is a variant of SPAD, by replacing aspects $a$ by subprofiles $S$ in Eq. 7.

Similarly, we can also define two further approaches, SSPAD and SRSPAD, which are analogous to SxQuAD and SRxQuAD (Section 2.2.3). 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 (Vargas and Castells, 2013).

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 paper.

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

## 4 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 1. The following sections present the eight methods in detail. Of these eight, DAMIB and DAMIB-COVER come from the work of Verstrepen and Goethals (2015). We have previously shown how DAMIB-COVER and IB+ can be used with SPAD in (Kaya and Bridge, 2017) and (Kaya and Bridge, 2018a), respectively. NN-1 is briefly referred to in (Kaya and Bridge, 2018b) for the task of automatic playlist completion.

### 4.1 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.

### *4.1.1* 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:

$$S_u^i = \{j \in I_u^+ : i \in \text{KNN}(j) \land i \neq j\} \cup \{i\} \tag{13}$$

KNN$(j)$ contains the top-$k$ items $j' \in I$ whose similarity to $j$, sim$(j, j')$, is highest. For similarity, 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 $\mathcal{S}_u$:

$$\mathcal{S}_u^* = \{S \in \mathcal{S}_u : \neg\exists S' \in \mathcal{S}_u \land S \subset S'\} \tag{14}$$

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 paper.

### *4.1.2* 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:

$$S_u^i = \{j \in I_u^+ : j \in \text{KNN}(i) \land i \neq j\} \cup \{i\} \tag{15}$$

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. 14.

Both NN-1 and NN-2 introduce a hyper-parameter, namely $k$, the number of neighbours to use. This is in addition to, and different from, the hyper-parameter $k$ used in the indicator function in Eq. 12. To distinguish them, we refer to the latter using $k_{ind}$ and the former by $k_{nn}$.

## 4.2 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. They 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. 14.

The recommender system that IB+, DAMIB and DAMIB-COVER use is an item-based nearest-neighbours recommender system (Deshpande and Karypis, 2004). 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 hyper-parameters, namely $n$ and $k$. $n$ is the number of recommendations and, since all three methods use item-based nearest-neighbours recommender systems (see below), $k$ is the number of neighbours used by these systems. 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, 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.

### *4.2.1* IB+

IB+ is an item-based nearest-neighbours recommender system for implicit ratings (i.e. for positive-only ratings) (Deshpande and Karypis, 2004). 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 users liked-item-set that have the candidate items as one of their $k$-nearest-neighbours:

$$S_u^i = \{j \in I_u^+ : i \in \text{KNN}(j)\} \tag{16}$$

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 $S_u^i$:

$$s_{\text{IB+}}(u,i) = \sum_{j \in S_u^i} \text{sim}(i,j) \tag{17}$$

The candidate subprofiles ($\mathcal{S}_u$) are the explanations ($S_u^i$) for the $n$ candidate items whose scores are highest. These are pruned using Eq. 14 to give the final subprofiles ($\mathcal{S}_u^*$).

### *4.2.2* DAMIB

The DAMIB and DAMIB-COVER recommender systems were originally developed by Verstrepen & Goethals for recommending to shared accounts (Verstrepen and Goethals, 2015). 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 the 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 system whose explanations will be the subprofiles that we use in SPAD. Even though we are re-purposing these two recommender systems 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 $\{i \in I : r_{ui} = \bot\}$ to each member of the powerset $S \in 2^{I_u^+}$. It does this in a similar way to the IB+ recommender system: 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 $j$:

$$s_{DAMIB}(S, i) = \sum_{j \in S, i \in KNN(j)} \text{sim}(i, j) \tag{18}$$

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

$$s_{DAMIB}(u, i) = \max_{S \in 2^{I_u^+}} \frac{1}{|S|^{0.75}} s_{DAMIB}(S, i) \tag{19}$$

As can be seen in Eq. 19, DAMIB multiplies the scores by $\frac{1}{|S|^{0.75}}$, where the value 0.75 is chosen based on experimental results in (Verstrepen and Goethals, 2015). 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:

$$S_u^i = \arg\max_{S \in 2^{I_u^+}} \frac{1}{|S|^p} s(S,i) \tag{20}$$

Eq. 19 and Eq. 20 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, $s_{DAMIB}(u,i)$. We, instead, use their explanations as candidate subprofiles, which we prune using Eq. 14.

#### *4.2.3* 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. 14 to give the final subprofiles ($\mathcal{S}_u^*$).

### 4.3 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

**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 $r_{ui} = \bot$
 2: Produce ranked list $t$ of all $i$ for which $r_{ui} = \bot$ 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 $|\mathcal{S}_u| < n$ do
 7:     $i \leftarrow t[r]$
 8:     if $|S_u^i \setminus Covered| \geq 1$ then
 9:         insert $S_u^i$ into $\mathcal{S}_u$
10:         $Covered \leftarrow Covered \cup S_u^i$
11:         remove $i$ from $t$
12:         if $Covered = I_u^+$ then
13:             $Covered \leftarrow \{\}$
14:             $r \leftarrow 1$
15:     else
16:         $r \leftarrow r + 1$
17:         if $r > |t|$ then
18:             $Covered \leftarrow \{\}$
19:             $r \leftarrow 1$
20: return $\mathcal{S}_u$
```

**Algorithm 3** COVER$(u, cp, alg, cover)$

```
Input: $u \in U$, $cp \in (0,1]$, $alg \in \{\text{IB+}, \text{DAMIB}\}$, $cover \in \{true, false\}$
Output: candidate subprofiles from $I_u^+$
 1: Use $alg$ to compute $s_{alg}(u,i)$ and $S_u^i$ for all $i$ for which $r_{ui} = \bot$
 2: Produce ranked list $t$ of all $i$ for which $r_{ui} = \bot$ 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:     $i \leftarrow t[r]$
 8:     if $cover \wedge |S_u^i \setminus Covered| < 1$ then
 9:         continue
10:     insert $S_u^i$ into $\mathcal{S}_u$
11:     $Covered \leftarrow Covered \cup S_u^i$
12:     if $|Covered| \geq cp \times |I_u^+|$ then
13:         break
14:     $r \leftarrow r + 1$
15:     if $r > |t|$ then
16:         break
17: return $\mathcal{S}_u$
```

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. 17) or DAMIB (Eq. 19). Finally, *cover* is 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

**Table 2** Subprofile detection using profile coverage

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

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.

We obtain our three new detection methods, IB+$_{\text{cp}}$, DAMIB$_{\text{cp}}$ and DAMIB-COVER$_{\text{cp}}$, by the way we call Algorithm 3, as shown in Table 2. IB+$_{\text{cp}}$, DAMIB$_{\text{cp}}$ and DAMIB-COVER$_{\text{cp}}$ replace hyper-parameter $n$ by hyper-parameter *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

We have presented eight subprofile detection methods. We compare them empirically in Section 6.2. Here, we compare them in a more qualitative way.

– *Algorithmic.* Referring to Table 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 hyper-parameters. All have $k_{ind}$ for use by Eq. 12. But NN-1 and NN-2 have just one more hyper-parameter, $k_{nn}$, the number of neighbours to use in Eq. 13 and Eq. 15 respectively. The other six methods, all of which use an item-based recommender algorithm, share hyper-parameter $k_{IB}$, which is used within the recommender algorithm (Eq. 16). Then, IB+, DAMIB and DAMIB-COVER additionally have hyper-parameter $n$, the number of recommendations to generate (and hence the number of candidate subprofiles), whereas IB+$_{\text{cp}}$, DAMIB$_{\text{cp}}$ and DAMIB-COVER$_{\text{cp}}$ have hyper-parameter *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 users 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).

**Table 3** Datasets

| | |
|---|---|
| MovieLens | 6040 users; 3706 items; ∼1M ratings |
| | avg. 165.6 ($\sigma$ = 192.74) movies per user |
| | 18 genres in total; avg. 1.65 per movie |
| LastFM | 992 users; 7280 items; ∼500k ratings |
| | avg. 515.94 ($\sigma$ = 475.14) artists per user |
| | 71833 tags in total; avg. 8 per artist |
| LibraryThing | 7279 users; 37232 items; ∼750K ratings |
| | avg. 102.95 ($\sigma$ = 132.68) books per user |
| | 4800 tags in total; avg. 9.08 per book |

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

## 5 Experiments

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

### 5.1 Datasets

We use three datasets: the MovieLens 1M dataset,[2] the LastFM dataset,[3] and LibraryThing dataset (Clements et al., 2008), which record user preferences for movies, music artists and books, respectively. We modify the MovieLens 1M and LastFM datasets in the same way as in (Kaminskas and Bridge, 2016). Specifically, the listening event frequencies in the LastFM dataset are converted into ratings on a 1–5 scale, artists who are listened to by fewer than 20 users are discarded, and the dataset is augmented with additional meta-data (user-generated tags). In the LibraryThing dataset, ratings are on a 1–5 scale, and we took an approach similar to the one in (Kaminskas and Bridge, 2016) for obtaining meta-data: 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. We summarize the characteristics of the resulting datasets in Table 3.

### 5.2 Recommender systems

We compare SPAD and RSPAD with the other diversification techniques available in the RankSys library[4]: MMR (Carbonell and Goldstein, 1998), xQuAD (Vargas et al., 2011), RxQuAD (Vargas et al., 2012) and c-pLSA (Wasilewski and Hurley, 2016). We also compare our methods with SxQuAD

[2] `http://grouplens.org/datasets/movielens/`
[3] `http://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html`
[4] `https://github.com/RankSys`

and SRxQuAD (Vargas and Castells, 2013). 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 following baseline recommenders (again using their RankSys implementations): probabilistic latent semantic analysis (pLSA) (Hofmann, 2004), a fast alternative least-squares matrix factorization recommender (MF) (Pilászy et al., 2010), and a factorization machine that uses Bayesian pairwise loss for ranking (FMBPR) (Bayer, 2015).

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 LastFM and LibraryThing datasets because the implementation is based on the maximum possible item features (71833 user-generated tags in LastFM and 4800 in LibraryThing), whereas the other re-ranking approaches that use item features only depend on the number of distinct features that describe the items in $I_u$.

## 5.3 Methodology

In our experiments, 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. Results are averaged over five runs with different random splits.

There are many hyper-parameters. We find their values using the validation sets. We select the hyper-parameter values for each baseline recommender that optimize precision on the validation sets (Vargas et al., 2012). Then, for each user, we generate a recommendation set $RS$, where $|RS| = 100$, using the baseline recommender with its best hyper-parameter values. We re-rank $RS$ to produce ranked list $RL$ using each of the re-ranking methods with each of their combinations of hyper-parameter values. Then, from each $RL$, we select the top-$N$ recommendations, $N = 10$. Finally, for each re-ranking method, we select hyper-parameter values that give the best average $\alpha$-nDCG across the validation sets. We show all the hyper-parameter values in the Appendix to this paper.

Now we train the baselines using their selected hyper-parameter 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 hyper-parameter values. Then, from each $RL$, we select the top-$N$ recommendations, $N = 10$, and measure the evaluation metrics that we describe in Section 5.4 on the test set.

## 5.4 Evaluation measures

For relevance, we measure Precision @$N$ for $N = 10$:

$$\text{Precision}\,@N = \frac{\sum_{i=1}^{N} \text{rel}(u, i)}{N} \tag{21}$$

where rel$(u, i)$ mean item $i$ is relevant to user $u$. We treat test set items with a rating of 4 or 5 as being relevant for the MovieLens and LastFM datasets. LibraryThing uses a 1–5 rating scale but allows half-marks, such as 4.5; for consistency, we treat a LibraryThing rating of 4 or more as being relevant.

There is no single ideal metric for diversity. Accordingly, we use five metrics: (i) $\alpha$-nDCG (Clarke et al., 2008), which is an aspect-aware version of nDCG; (ii) intent-aware expected reciprocal rank, ERRIA (Agrawal et al., 2009); (iii) subtopic recall, Srecall (Zhai et al., 2003); (iv) Intra-List Diversity, ILD, as computed by Eq. 3 (Ziegler et al., 2005); and (v) Expected Intra-List Diversity, EILD (Vargas and Castells, 2011), which is a rank- and relevance-aware version of ILD. We give their definitions below. For each, we use their RankSys implementations.

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

$$\alpha\text{-nDCG}(L) = \frac{1}{\alpha\text{-IDCG}} \sum_{i \in L} \left[ \frac{1}{\log_2(\text{r}(i, L) + 1)} \sum_{f \in \mathcal{F}} \text{rel}(i|u, f) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - \alpha\, \text{rel}(j|u, f)) \right] \tag{22}$$

where $\alpha$-IDCG is the highest possible value of $\alpha$-nDCG in the case where the recommendation set is made of ideally diversified relevant items, $L$ is the ranked list of recommended items (of size $N$) that is being evaluated, r$(i, L)$ is the position of $i$ in $L$, and 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 (Vargas Sandoval, 2015) given earlier, we use $p(\mathit{stop}|\mathit{rel}) = 0.5$ in RxQuAD and RSPAD too.)

We also use ERRIA (Agrawal et al., 2009), which is an intent-aware measure:

$$\text{ERRIA}(L) = \sum_{f \in \mathcal{F}} p(f|u) \sum_{i \in L} \frac{1}{\text{r}(i, L)} p(\text{rel}\,|i, u, f) \prod_{\substack{j \in L, \\ \text{r}(j,L) < \text{r}(i,L)}} (1 - p(\text{rel}\,|j, u, f)) \tag{23}$$

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 ERRIA cannot be used directly (Wasilewski and Hurley, 2016). What we do is to use the probabilities computed by xQuAD, even when evaluating SPAD and RSPAD. This gives xQuAD an advantage, which must be kept in mind when looking at the results.

Subtopic Recall, Srecall, is a metric that measures how well the recommendation set covers the feature space (Zhai et al., 2003):

$$\text{Srecall}(L) = \frac{|\cup_{i\in L}\mathcal{F}_i|}{|\mathcal{F}|} \tag{24}$$

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 (Ziegler et al., 2005). We presented its definition already as Eq. 3.

Expected Intra-List Diversity (Vargas and Castells, 2011) is a rank- and relevance-aware version of Equation 3:

$$\text{EILD}(L) = \sum_{i,j\in L, i\neq j} C_i \,\text{disc}(k_i)\,\text{disc}(k_j|k_i) p(\text{rel}\,|i,u) p(\text{rel}\,|j,u)\,\text{dist}(i,j) \tag{25}$$

where $\text{disc}(k_i) = \frac{1}{log(k_i+2)}$ is the rank discount for item $i$ at position $k$, and $\text{disc}(k_j|k_i) = \text{disc}(\max(1, k_j - k_i))$ 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 features 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 (Vargas and Castells, 2011).

We have already mentioned that xQuAD has an advantage in the case of ERRIA. But, in fact, all five measures of diversity are computed with respect to item features $\mathcal{F}$. All may therefore favour recommenders that re-rank using those features, such as MMR, xQuAD, RxQuAD and c-pLSA. Our new methods, SPAD and RSPAD, make no use of the features at all and so they are at a disadvantage in these experiments.

## 6 Results

We divide this section into three: first we analyze the subprofiles that we found in the user profiles in the different datasets; then, we compare the performance

**Table 4** Subprofile Stats

| | |
|---|---|
| MovieLens | 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 |
| LastFM | 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 |
| LibraryThing | 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 |

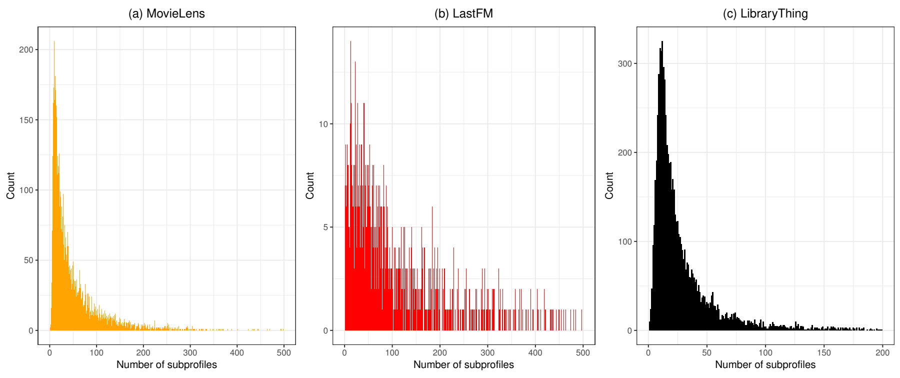

**Fig. 1** Number of subprofiles

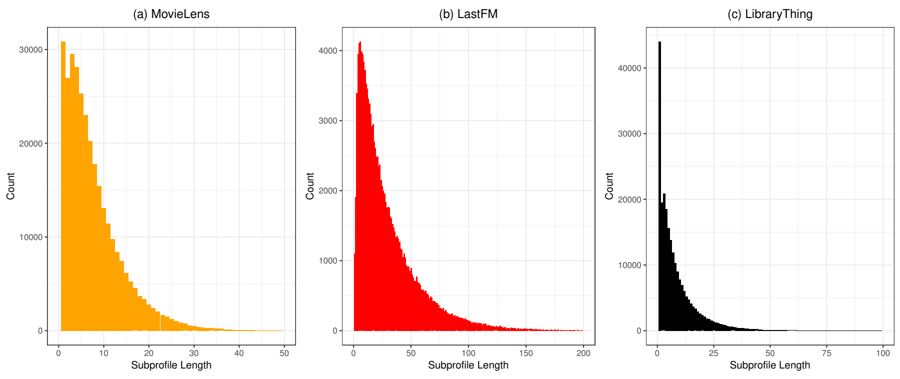

**Fig. 2** Length of subprofiles

of the different subprofile detection methods; finally, we compare SPAD and RSPAD to existing intent-aware diversification algorithms.

## 6.1 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 NN-1 as subprofile detection algorithm since, from the discussion we gave in Section 4.4 and the results we will give in Section 6.2, it is the one we select to use in Section 6.3. NN-1 has hyper-parameters. We set their values using the method already described. Specifically, we use the values in the Appendix where MF is the baseline recommender.

Table 4 shows the average number of subprofiles per user and the average length of the subprofiles. In more detail, Figure 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 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 LastFM dataset both the number of subprofiles and the lengths of the subprofiles are greater than for the MovieLens and LibraryThing 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. Hence, user profiles are longer in the LastFM dataset — see Table 3. Longer user profiles tend to contain longer liked-item-sets, which will tend to give rise to more subprofiles and longer subprofiles.

Table 4 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:

$$\text{avg-p-sim}(u) = \frac{\sum_{S \in \mathcal{S}_u^*} \sum_{S' \in \mathcal{S}_u^*, S \neq S'} \text{p-sim}(S, S')}{|\mathcal{S}_u^*|(|\mathcal{S}_u^*| - 1)} \tag{26}$$

where $\mathcal{S}_u^*$ is the final set of subprofiles for $u$ and 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 is the mean avg-p-sim($u$) for all users $u$.

As Table 4 shows, LastFM subprofiles are more similar to each other than MovieLens and LibraryThing 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 MovieLens dataset — Table 5. 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.

**Table 5** Example subprofiles. These are the final subprofiles of user 5870 in the MovieLens 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 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 |

The twelve subprofiles shown in Table 5 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. 14: they are subsets of other candidate subprofiles.

## 6.2 Results for different subprofile detection methods

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

**Table 6** Precision and $\alpha$-nDCG for different subprofile detection approaches for MovieLens, LastFM and LibraryThing datasets using MF as the baseline. 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. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$) except those shown in italics.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision @10 | $\alpha$-nDCG | Precision @10 | $\alpha$-nDCG |
| MovieLens | | | | | |
| 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+$_{\text{cp}}$ | 0.4 | 0.2953 | 0.3305 | 1.27 | 3.36 |
| DAMIB$_{\text{cp}}$ | 0.5 | *0.2934* | 0.3313 | 0.62 | 3.61 |
| DAMIB-COVER$_{\text{cp}}$ | 0.5 | 0.2981 | 0.337 | 2.21 | 5.39 |
| LastFM | | | | | |
| 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+$_{\text{cp}}$ | 0.2 | *0.4682* | *0.4242* | 0.61 | -0.03 |
| DAMIB$_{\text{cp}}$ | 0.2 | 0.47 | *0.4255* | 1.0 | 0.27 |
| DAMIB-COVER$_{\text{cp}}$ | 0.2 | 0.471 | 0.4278 | 1.21 | 0.81 |
| LibraryThing | | | | | |
| 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+$_{\text{cp}}$ | 0.4 | 0.1857 | 0.256 | 7.18 | 6.14 |
| DAMIB$_{\text{cp}}$ | 0.4 | 0.1892 | 0.2601 | 9.16 | 7.83 |
| DAMIB-COVER$_{\text{cp}}$ | 0.5 | 0.1909 | **0.2625** | 10.16 | 8.82 |

Table 6 shows the results for the MovieLens, LastFM and LibraryThing datasets using MF as the baseline recommender system. We do not show the results for the cases where pLSA and FMBPR are the baseline recommender systems due to space limitations. We chose to show the MF results because all of the re-ranking algorithms achieve both their best precision and $\alpha$-nDCG values using MF as the baseline recommender.

Table 6 shows that, of the eight methods, NN-1 has the highest precision for all but the LibraryThing 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 MovieLens dataset, NN-2 performs the best; for LastFM, it is NN-1; and for LibraryThing, it is DAMIB-COVER$_{\text{cp}}$. On balance, NN-1 is the best-performing method for the MF baseline. Although not shown, it is also the best-performing method when pLSA and FMBPR are the baselines. The

**Table 7** Precision and $\alpha$-nDCG for different re-ranking algorithms on the MovieLens dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The value of $\lambda$ that optimizes $\alpha$-nDCG for this baseline and each re-ranking strategy is given. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$).

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision @10 | $\alpha$-nDCG | Precision @10 | $\alpha$-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 |

**Table 8** Diversity metrics except $\alpha$-nDCG for different re-ranking algorithms on the MovieLens dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The values of $\lambda$ that we use are the ones in Table 7. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$) except those shown in italics.

| | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|
| | ERRIA | Srecall | EILD | ILD | ERRIA | Srecall | EILD | ILD |
| MF | 0.2102 | 0.4783 | 0.2123 | 0.7176 | | | | |
| MMR | 0.2115 | 0.5116 | 0.2217 | 0.7566 | 0.61 | 6.97 | 4.45 | 5.44 |
| xQuAD | 0.2232 | **0.6271** | 0.2033 | 0.7592 | 6.17 | 31.13 | -4.24 | 5.80 |
| RxQuAD | 0.2369 | 0.5834 | 0.1907 | 0.7405 | 12.67 | 21.98 | -10.2 | 3.19 |
| SxQuAD | 0.2297 | 0.5846 | 0.2016 | 0.7353 | 9.25 | 22.23 | -5.06 | 2.46 |
| SRxQuAD | **0.24** | 0.6199 | 0.2094 | **0.7768** | 14.19 | 29.61 | -1.38 | 8.25 |
| c-pLSA | 0.2174 | *0.478* | 0.2171 | 0.7083 | 3.44 | -0.06 | 2.27 | -1.29 |
| SPAD | 0.2197 | 0.4957 | **0.2219** | 0.7244 | 4.52 | 3.63 | 4.5 | 0.95 |
| RSPAD | 0.2202 | 0.5071 | 0.2211 | 0.7309 | 4.74 | 6.03 | 4.13 | 1.86 |

qualitative arguments we gave in Section 4.4 also favoured NN-1 and NN-2: they are simpler, run faster and require setting fewer hyper-parameters. Hence, in the next section, we present results for SPAD and RSPAD using NN-1.

## 6.3 Results for intent-aware diversification algorithms

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 results for different values of $\lambda$.

### *6.3.1 Results for different algorithms*

The precision and $\alpha$-nDCG results for the experiments on the MovieLens dataset are shown in Table 7; and the results for the other diversity metrics (i.e. ERRIA, Srecall, EILD and ILD) are given in Table 8. MF is the

**Table 9** Precision and $\alpha$-nDCG for different re-ranking algorithms on the LastFM dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The value of $\lambda$ that optimizes $\alpha$-nDCG for this baseline and each re-ranking strategy is given. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$).

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision @10 | $\alpha$-nDCG | Precision @10 | $\alpha$-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 |

best performing baseline in terms of precision and $\alpha$-nDCG. Moreover, all of the re-ranking approaches achieve their best values of precision and $\alpha$-nDCG by re-ranking $RS$ recommendation sets initially generated by the MF baseline recommender. Therefore, we only show re-ranking results for MF. In each block of Tables 7 and 8 (and also Tables 9, 10, 11 and 12), results for the MF baseline are presented first, and then results for each of the re-ranking algorithms. For each algorithm, we report the results using the value of $\lambda$ that gives highest $\alpha$-nDCG on the validation set (Wasilewski and Hurley, 2016).

Consider precision and $\alpha$-nDCG first. SPAD achieves the highest precision, while c-pLSA and RSPAD achieve the second and the third best precision scores. For $\alpha$-nDCG, SxQuAD is the best re-ranking method, and 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 features. Indeed, $\alpha$-nDCG is very similar to what is used for re-ranking in xQuAD and its variations. Even so, SPAD and RSPAD achieve higher diversity than c-pLSA and MMR. Furthermore, when we look at precision and $\alpha$-nDCG together, we see that xQuAD and its variations achieve their diversity at the expense of the largest decreases in precision.

Next, consider the other diversity metrics shown in Table 8. SPAD, RSPAD and MMR improve all these diversity metrics. But MMR does this at the cost of lower precision whereas SPAD and RSPAD increase precision in comparison with the baseline. 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 compared with the baseline with respect to all but the EILD metric.

The results for the LastFM dataset are in Tables 9 and 10. 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 9. Here, RSPAD and SPAD achieve the highest and second highest precision scores. Again, despite making no use of explicit features, SPAD and RSPAD increase diversity measured

**Table 10** Diversity metrics except $\alpha$-nDCG for different re-ranking algorithms on the LastFM dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The values of $\lambda$ that we use are the ones in Table 9. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p <$ 0.05) except those shown in italics.

| | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|
| | ERRIA | Srecall | EILD | ILD | ERRIA | Srecall | EILD | ILD |
| MF | 0.2012 | 6.39E-4 | 0.3671 | 0.7638 | | | | |
| MMR | *0.2013* | **7.06E-4** | **0.378** | **0.8094** | 0.06 | 10.37 | 2.99 | 5.97 |
| xQuAD | 0.2067 | 6.61E-4 | 0.3779 | 0.778 | 2.74 | 3.4 | 2.94 | 1.87 |
| RxQuAD | 0.21 | 6.34E-4 | *0.3661* | 0.7652 | 4.38 | -0.89 | -0.25 | 0.19 |
| SxQuAD | 0.2067 | 6.26E-4 | *0.3666* | 0.7552 | 2.75 | 2.05 | -0.12 | -1.12 |
| SRxQuAD | **0.211** | 6.37E-4 | 0.3689 | 0.7682 | 4.91 | -0.38 | 0.49 | 0.58 |
| SPAD | *0.2036* | 6.36E-4 | 0.3734 | 0.762 | 1.22 | -0.53 | 1.872 | -0.23 |
| RSPAD | 0.2041 | 6.35E-4 | 0.3743 | *0.7624* | 1.48 | -0.71 | 1.97 | -0.18 |

**Table 11** Precision and $\alpha$-nDCG for different re-ranking algorithms on the LibraryThing dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The value of $\lambda$ that optimizes $\alpha$-nDCG for this baseline and each 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.

| | | Metrics | | % change over baseline | |
|---|---|---|---|---|---|
| | $\lambda$ | Precision @10 | $\alpha$-nDCG | Precision @10 | $\alpha$-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 |

by $\alpha$-nDCG. xQuAD achieves the highest $\alpha$-nDCG. Interestingly, all the re-ranking methods, except MMR, increase precision (albeit only slightly in the case of RxQuAD) 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 10. The diversity with respect to ERRIA is increased by all of the re-ranking approaches compared with the baseline. With respect to Srecall, only MMR and xQuAD out-performed the baseline. For EILD, all except RxQuAD and SxQuAD improve the diversity compared with the baseline. For ILD, MMR achieves the best and xQuAD the second best improvement compared with the baseline; RxQuAD and SRxQuAD show a slight increase; but SPAD, RSPAD and SxQuAD show a decrease.

The results for the LibraryThing dataset are in Tables 11 and 12. c-pLSA is missing from these results for the same reason we explained above for the LastFM dataset. First, consider the results shown in Table 11. They are very similar to the ones for the LastFM dataset. RSPAD and SPAD achieve the

**Table 12** Diversity metrics except $\alpha$-nDCG for different re-ranking algorithms on the LibraryThing dataset using MF as the baseline. The best result for each metric is highlighted in bold for each block. The values of $\lambda$ that we use are the ones in Table 11. All of the results are statistically significant with respect to their baseline (Wilcoxon signed rank with $p < 0.05$) except those shown in italics.

| | Metrics | | | | % change over baseline | | | |
|---|---|---|---|---|---|---|---|---|
| | ERRIA | Srecall | EILD | ILD | ERRIA | Srecall | EILD | ILD |
| MF | 0.1143 | 0.0097 | 0.1137 | 0.7577 | | | | |
| MMR | *0.1143* | 0.0099 | 0.115 | 0.7687 | - | 2.44 | 1.11 | 1.46 |
| xQuAD | **0.1213** | **0.0104** | **0.1314** | **0.7884** | 6.19 | 7.17 | 15.53 | 4.05 |
| RxQuAD | 0.1197 | 0.0097 | 0.1209 | 0.7721 | 4.82 | 0.51 | 6.29 | 1.9 |
| SxQuAD | 0.1195 | 0.0092 | 0.1223 | 0.7441 | 4.58 | -4.9 | 7.52 | -1.79 |
| SRxQuAD | 0.1212 | 0.0098 | 0.1257 | 0.7802 | 6.1 | 1.3 | 10.58 | 2.97 |
| SPAD | 0.1183 | 0.0099 | 0.1302 | 0.7726 | 3.57 | 2.18 | 14.48 | 1.97 |
| RSPAD | 0.1178 | 0.0097 | 0.128 | 0.7655 | 3.07 | 0.22 | 12.56 | 1.03 |

highest precision and they both increase $\alpha$-nDCG along with precision. xQuAD achieves the highest $\alpha$-nDCG. Again, all of the re-ranking approaches, except MMR, increase precision along with $\alpha$-nDCG. SPAD, RSPAD and xQuAD give the best balance between increased precision and increased $\alpha$-nDCG.

Finally, the results for the other diversity metrics are presented in Table 12. With a few exceptions, all the re-ranking approaches increase all the diversity metrics. Specifically, MMR does not improve diversity in terms of ERRIA, and SxQuAD does not improve diversity in terms of ILD and Srecall.

### *6.3.2 Results for different values of $\lambda$*

Here we look at the effect of parameter $\lambda$, which controls the balance between relevance and diversity in Equation 1. 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 MovieLens dataset (Figure 3 for precision and Figure 4 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 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 LastFM can be found in Figures 5 and 6. Again, for precision, 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

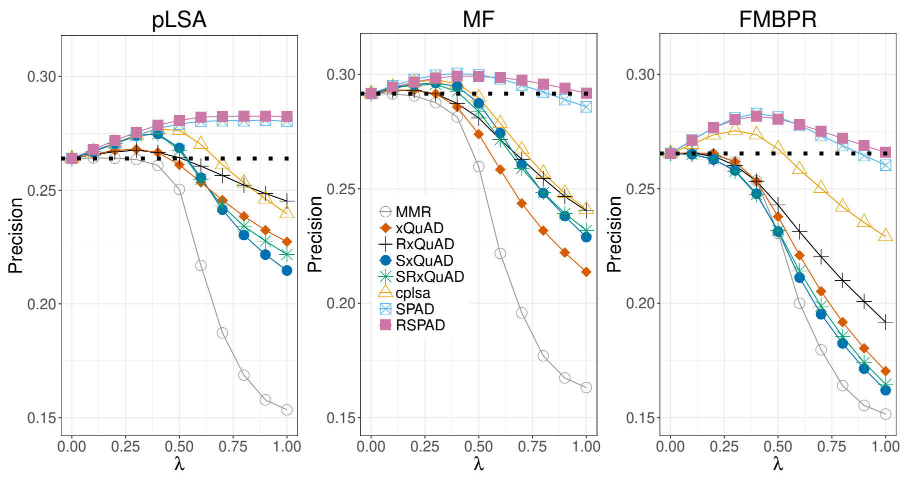

**Fig. 3** MovieLens dataset. Precision@10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baselines is shown by the dotted line.

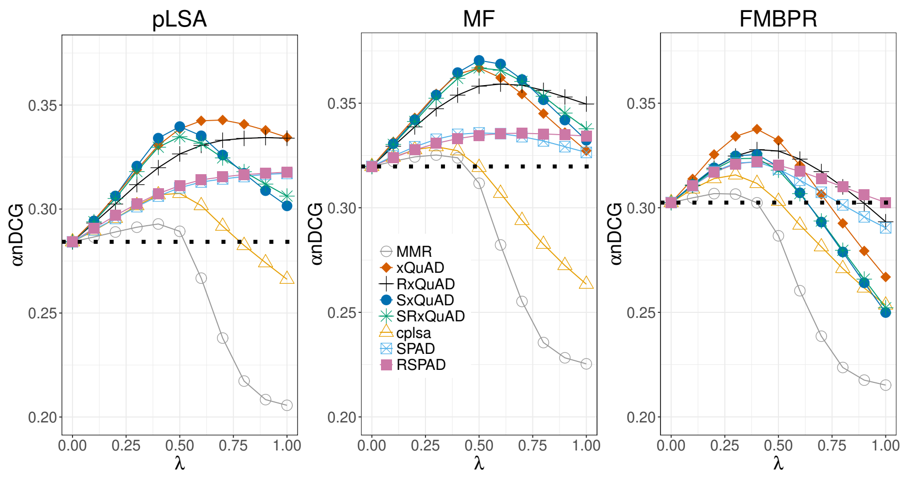

**Fig. 4** MovieLens dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-nDCG for each baseline is shown by the dotted line.

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 xQuAD and MMR achieve higher $\alpha$-nDCG values than SPAD and RSPAD for some small values of $\lambda$, where MF is the baseline. For

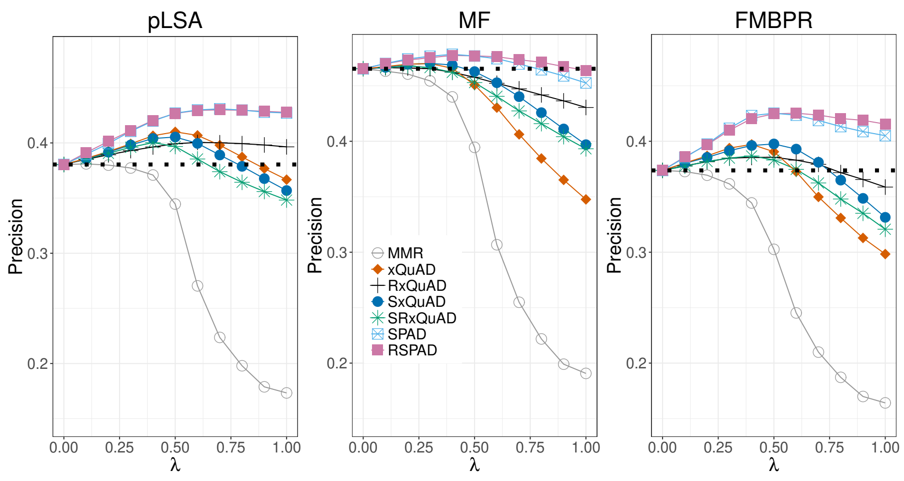

**Fig. 5** LastFM dataset. Precision @10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

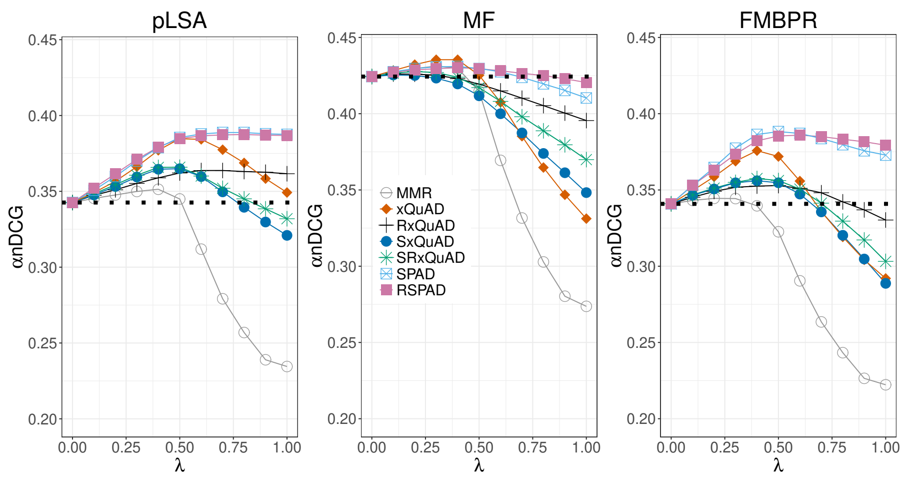

**Fig. 6** LastFM dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-nDCG for each baseline is shown by the dotted line.

the LastFM dataset, SPAD and RSPAD gives the best balance between increased precision and $\alpha$-nDCG for almost all values of $\lambda$.

LibraryThing results are in Figures 7 and 8. Precision results for SPAD and RSPAD are similar to those for LastFM. One small difference is, where MF is the baseline: for larger values of $\lambda$, xQuAD and its variations slightly outperform SPAD and 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 $\lambda$.

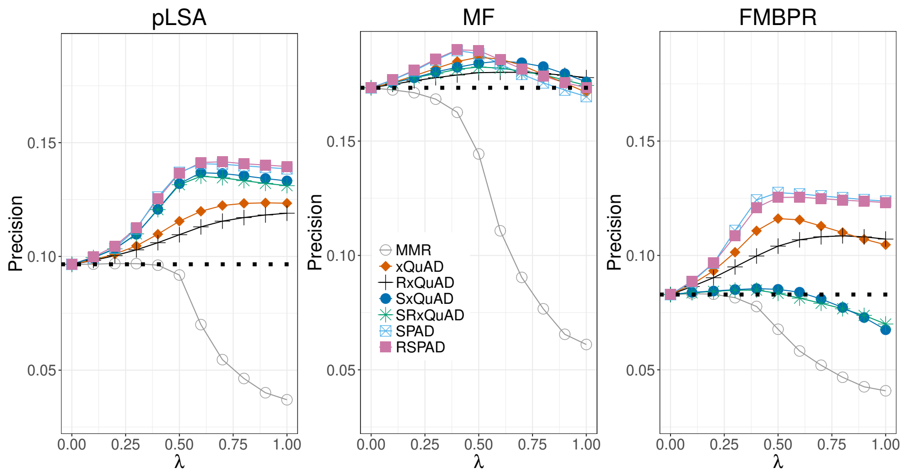

**Fig. 7** LibraryThing dataset. Precision@10 for varying $\lambda$ using pLSA, MF and FMBPR baselines. Precision for each baseline is shown by the dotted line.

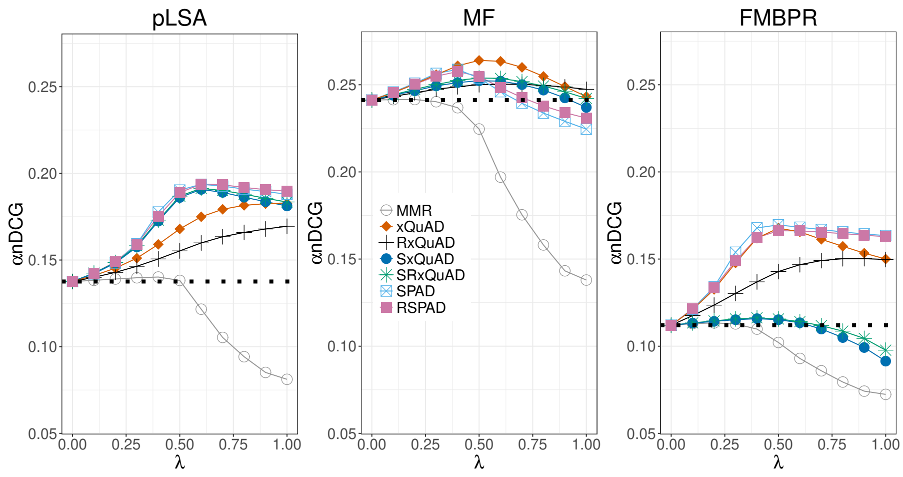

**Fig. 8** LibraryThing dataset. $\alpha$-nDCG for varying $\lambda$ using pLSA, MF and FMBPR baselines. $\alpha$-nDCG for each baseline is shown by the dotted line.

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 paper, 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.

## 7 Conclusions and Future Work

We have presented a new form of intent-aware diversification called Subprofile-Aware Diversification (SPAD) and its variant, Relevance-based SPAD (RSPAD). In both, the aspects to be covered by the re-ranked recommendations are subprofiles of the user's profile, each representing a distinct user taste or interest, instead of item features.

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.

Using subprofiles detected by NN-1, we have compared SPAD and RSPAD against several existing intent-aware diversification methods, ones that use item features (such as genres) 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.

In the future, we would like to apply SPAD to new domains. One challenge arises in domains where the temporal ordering of the items in a user's profile is important, such as e-commerce shopping and tourism. We have already applied SPAD to one such domain, i.e. the completion of music playlists (Kaya and Bridge, 2018b), but, when doing so, we ignored the temporal ordering. We would also like to explore the interpretability of 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 (Bridge and Dunleavy, 2014; Bilgic and Mooney, 2005). We also hope to use subprofiles directly in the cost function that is optimized by a recommender system, instead of using them for re-ranking. Finally, we will adapt diversity evaluation metrics so that they use subprofiles instead of item features, allowing a comparison in which SPAD and RSPAD are not so much at a disadvantage.

**Acknowledgements** This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 which is co-funded under the European Regional Development Fund.

## A Hyper-parameters Values

First, we will show the hyper-parameter values for the baseline recommender systems.

For pLSA, MF and FMBPR, we choose the number of latent factors ($d$) from $V = \{10, 30, 50, \ldots, 290, 310\}$. FMBPR's learning rate ($lr$) and regularization parameters ($regW$ and $regM$) are chosen from $\{0.01, 0.001\}$, and MF's confidence level ($\alpha$) is chosen from $\{1, 2, \ldots, 10\}$. The values that get selected are as follows:

- pLSA: $d = 50$ for MovieLens; $d = 30$ for LastFM ; $d = 270$ for LibraryThing.
- MF: $d = 30, \alpha = 1.0$ for MovieLens; $d = 30, \alpha = 1.0$ for LastFM; $d = 330, \alpha = 1.0$ for LibraryThing.
- FMBPR: $d = 190$, $lr = 0.01$, $regM = 0.01$, $regW = 0.001$ for MovieLens; $d = 10$, $lr = 0.01$, $regW = 0.01$, $regM = 0.001$ for LastFM; $d = 270$, $lr = 0.01$, $regM = 0.01$, $regW = 0.01$ for LibraryThing.

Second, we show the hyper-parameter values for the re-ranking and the subprofile detection methods.

All of the re-ranking approaches have hyper-parameter $\lambda$, which controls the balance between relevance and diversity (Eq. 1), whose value we select from $[0.1, 0.2, \ldots, 1.0]$.

For the subprofile detection methods, we select the values of $k_{ind}$, $k_{nn}$ and $k_{IB}$ from $V$, and we select the value of $cp$ from the set $[0.5, 0.6, \ldots, 1.0]$.

Table 13 shows the values that get selected.

**Table 13** Hyper-parameters values for different subprofile detection approaches for all three datasets and all three baselines. These are optimized using validation sets.

| | pLSA | | | | | | MF | | | | | | FMBPR | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ | $\lambda$ | $n$ | $k_{nn}$ | $k_{IB}$ | $k_{ind}$ | $cp$ |
| MovieLens | | | | | | | | | | | | | | | | | | |
| 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+$_{cp}$ | 0.7 | - | - | 10 | 10 | 1.0 | 0.4 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{cp}$ | 1.0 | - | - | 10 | 30 | 0.9 | 0.5 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 0.9 |
| DAMIB-COVER$_{cp}$ | 1.0 | - | - | 10 | 30 | 0.9 | 0.5 | - | - | 10 | 30 | 1.0 | 0.4 | - | - | 10 | 10 | 0.9 |
| LastFM | | | | | | | | | | | | | | | | | | |
| 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+$_{cp}$ | 0.5 | - | - | 10 | 30 | 1.0 | 0.2 | - | - | 50 | 10 | 0.5 | 0.5 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{cp}$ | 0.6 | - | - | 50 | 10 | 1.0 | 0.2 | - | - | 70 | 10 | 1.0 | 0.6 | - | - | 50 | 10 | 1.0 |
| DAMIB-COVER$_{cp}$ | 0.6 | - | - | 50 | 10 | 1.0 | 0.2 | - | - | 70 | 10 | 1.0 | 0.6 | - | - | 50 | 10 | 1.0 |
| LibraryThing | | | | | | | | | | | | | | | | | | |
| 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+$_{cp}$ | 0.5 | - | - | 10 | 10 | 1.0 | 0.4 | - | - | 10 | 10 | 1.0 | 0.5 | - | - | 10 | 10 | 1.0 |
| DAMIB$_{cp}$ | 0.6 | - | - | 30 | 10 | 1.0 | 0.4 | - | - | 30 | 10 | 0.8 | 0.5 | - | - | 10 | 10 | 0.9 |
| DAMIB-COVER$_{cp}$ | 0.6 | - | - | 30 | 10 | 0.5 | 0.5 | - | - | 10 | 10 | 1.0 | 0.5 | - | - | 10 | 10 | 1.0 |

## References

Adomavicius G, Kwon Y (2009) Toward more diverse recommendations: Item re-ranking methods for recommender systems. In: Procs. of the 19th Workshop on Information Technologies and Systems, pp 79–84

Adomavicius G, Kwon YO (2008) Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach. In: Procs. of the 2008 Workshop on Information Technologies and Systems, pp 151–156

Agrawal R, Gollapudi S, Halverson A, Ieong S (2009) Diversifying search results. In: Procs. of the 2nd ACM International Conference on Web Search and Data Mining, pp 5–14

Anelli VW, Bellini V, Di Noia T, La Bruna W, Tomeo P, Di Sciascio E (2017) An Analysis on Time- and Session-aware Diversification in Recommender Systems. In: Procs. of the 25th Conference on User Modeling, Adaptation and Personalization, ACM, pp 270–274

Antikacioglu A, Ravi R (2017) Post Processing Recommender Systems for Diversity. In: Procs. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 707–716

Bayer I (2015) Fastfm: a library for factorization machines. arXiv preprint arXiv:150500641

Bilgic M, Mooney RJ (2005) Explaining recommendations: Satisfaction vs. promotion. In: Beyond Personalization Workshop, IUI, vol 5, p 153

Bridge D, Dunleavy K (2014) 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, p 22

Carbonell J, Goldstein J (1998) The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In: Procs. of the 21st ACM SIGIR International Conference on Research and Development in Information Retrieval, ACM, pp 335–336

Castells P, Hurley NJ, Vargas S (2015) Novelty and Diversity in Recommender Systems. In: Ricci F, Others (eds) Recommender Systems Handbook (2nd edition), Springer, pp 881–918

Cheng P, Wang S, Ma J, Sun J, Xiong H (2017) Learning to Recommend Accurate and Diverse Items. In: Procs. of the 26th International Conference on World Wide Web, pp 183–192

Clarke CLA, Kolla M, Cormack GV, Vechtomova O, Ashkan A, Büttcher S, MacKinnon I (2008) Novelty and diversity in information retrieval evaluation. In: Procs. of the 31st ACM SIGIR International Conference on Research and Development in Information Retrieval, pp 659–666

Clements M, de Vries AP, Reinders MJ (2008) Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In: Workshop on Exploiting Semantic Annotations in Information Retrieval

Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS) 22(1):143–177

Di Noia T, Rosati J, Tomeo P, Di Sciascio E (2017) Adaptive Multi-attribute Diversity for Recommender Systems. Information Sciences 382(C):234–253

Eskandanian F, Mobasher B, Burke R (2017) A Clustering Approach for Personalizing Diversity in Collaborative Recommender Systems. In: Procs. of the 25th Conference on User Modeling, Adaptation and Personalization, ACM, pp 280–284

Harper FM, Konstan JA (2015) The MovieLens Datasets: History and Context. ACM Trans Interact Intell Syst 5(4):19:1–19:19

Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Transactions on Information Systems 22(1):89–115

Hurley NJ (2013) Personalised ranking with diversity. In: Procs. of the 7th ACM Conference on Recommender Systems, pp 379–382

Kaminskas M, Bridge D (2016) Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Trans Interact Intell Syst 7(1):2:1–2:42

Kaya M, Bridge D (2017) Intent-aware diversification using item-based subprofiles. In: Tikk D, Pu P (eds) Procs. of the Poster Track of the 11th ACM Conference on Recommender Systems, CEUR Workshop Proceedings, vol-1905

Kaya M, Bridge D (2018a) Accurate and diverse recommendations using item-based subprofiles. In: Procs. of the 31th International Florida Artificial Intelligence Research Society Conference, AAAI, pp 462–467

Kaya M, Bridge D (2018b) Automatic playlist continuation using subprofile-aware diversification. In: Procs. of the Workshop on the ACM Recommender Systems Challenge (Workshop Programme of the Twelfth ACM Conference on Recommender Systems), pp 1:1–1:6

Kelly JP, Bridge D (2006) Enhancing the diversity of conversational collaborative recommendations: a comparison. Artificial Intelligence Review 25(1-2):79–95

Koren Y, Bell R (2011) Advances in Collaborative Filtering. In: Ricci F, Others (eds) Recommender Systems Handbook, Springer, pp 145–186

Kula M (2017) Mixture-of-tastes Models for Representing Users with Diverse Interests. CoRR abs/1711.08379

Liang S, Ren Z, De Rijke M (2014) Personalized Search Result Diversification via Structured Learning. In: Procs. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 751–760

McNee SM, Riedl J, Konstan JA (2006) Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In: Procs. of the CHI'06 Extended Abstracts on Human Factors in Computing Systems, pp 1097–1101

Pilászy I, Zibriczky D, Tikk D (2010) Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. In: Procs. of the Fourth ACM Conference on Recommender Systems, ACM, pp 71–78

Puthiya Parambath SA, Usunier N, Grandvalet Y (2016) A coverage-based approach to recommendation diversity on similarity graph. In: Procs. of the 10th ACM Conference on Recommender Systems, pp 15–22

Santos RLT, Macdonald C, Ounis I (2010) Exploiting Query Reformulations for Web Search Result Diversification. In: Procs. of the 19th International Conference on World Wide Web, pp 881–890

Shi Y, Zhao X, Wang J, Larson M, Hanjalic A (2012) Adaptive Diversification of Recommendation Results via Latent Factor Portfolio. In: Procs. of the 35th ACM SIGIR International Conference on Research and Development in Information Retrieval, pp 175–184

Smyth B, McClave P (2001) Similarity vs. diversity. In: Procs. of the International Conference on Case-Based Reasoning, Springer, pp 347–361

Su R, Yin L, Chen K, Yu Y (2013) Set-oriented personalized ranking for diversified top-n recommendation. In: Procs. of the 7th ACM Conference on Recommender Systems, pp 415–418

Tsai CH, Brusilovsky P (2017) Leveraging Interfaces to Improve Recommendation Diversity. In: Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, ACM, pp 65–70

Tsai CH, Brusilovsky P (2018) Beyond the Ranked List: User-Driven Exploration and Diversification of Social Recommendation. In: 23rd International Conference on Intelligent User Interfaces, ACM, pp 239–250

Vallet D, Castells P (2012) Personalized Diversification of Search Results. In: Procs. of the 35th ACM SIGIR International Conference on Research and Development in Information Retrieval, pp 841–850

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the 5th ACM Conference on Recommender systems, pp 109–116

Vargas S, Castells P (2013) Exploiting the diversity of user preferences for recommendation. In: Procs. of the 10th Conference on Open Research Areas in Information Retrieval, pp 129–136

Vargas S, Castells P, Vallet D (2011) Intent-oriented Diversity in Recommender Systems. In: Procs. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp 1211–1212

Vargas S, Castells P, Vallet D (2012) Explicit Relevance Models in Intent-oriented Information Retrieval Diversification. In: Procs. of the 35th ACM SIGIR International Conference on Research and Development in Information Retrieval, ACM, pp 75–84

Vargas Sandoval S (2015) Novelty and Diversity Evaluation and Enhancement in Recommender Systems. PhD thesis, Universidad Autónoma de Madrid, Spain

Verstrepen K, Goethals B (2015) Top-N Recommendation for Shared Accounts. In: Procs. of the 9th ACM Conference on Recommender Systems, pp 59–66

Wasilewski J, Hurley N (2016) Intent-Aware Diversification Using a Constrained PLSA. In: Procs. of the 10th ACM Conference on Recommender Systems, pp 39–42

Wasilewski J, Hurley N (2017) Personalised Diversification Using Intent-Aware Portfolio. In: Adjunct Publication of the 25th ACM Conference on User Modeling, Adaptation and Personalization, pp 71–76

Willemsen MC, Graus MP, Knijnenburg BP (2016) Understanding the Role of Latent Feature Diversification on Choice Difficulty and Satisfaction. User Modeling and User-Adapted Interaction 26(4):347–389

Zhai CX, Cohen WW, Lafferty J (2003) Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: Procs. of the 26th ACM SIGIR International Conference on Research and Development in Information Retrieval, pp 10–17

Zhang M, Hurley N (2008) Avoiding monotony: improving the diversity of recommendation lists. In: Procs. of the 2nd ACM Conference on Recommender Systems, pp 123–130

Zhang M, Hurley N (2009) 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-Volume 01, IEEE Computer Society, pp 508–515

Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Procs. of the 14th International Conference on World Wide Web, pp 22–32

**Mesut Kaya** is a a Ph.D. candidate in the Insight Centre for Data Analytics based in the School of Computer Science and Information Technology at University College Cork, Ireland. His research focuses on ways to diversify set of recommendations.

**Derek Bridge** is a senior lecturer also in the Insight Centre for Data Analytics based in the School of Computer Science and Information Technology at University College Cork, Ireland. At national level within the Insight Centre, he leads the Recommender Systems Group.