Noname manuscript No.
(will be inserted by the editor)

How does Serendipity affect Diversity in Recommender Systems? A Serendipity-Oriented Greedy Algorithm

Denis Kotkov[1] · Jari Veijalainen[1] · Shuaiqiang Wang[2]

This is the final draft of the following article: Kotkov, D., Veijalainen, J. & Wang, S. How does serendipity affect diversity in recommender systems? A serendipity-oriented greedy algorithm. *Computing* **102**, 393411 (2020). https://doi.org/10.1007/s00607-018-0687-5

Abstract Most recommender systems suggest items that are popular among all users and similar to items a user usually consumes. As a result, the user receives recommendations that she/he is already familiar with or would find anyway, leading to low satisfaction. To overcome this problem, a recommender system should suggest novel, relevant and unexpected i.e., serendipitous items. In this paper, we propose a serendipity-oriented, reranking algorithm called a serendipity-oriented greedy (SOG) algorithm, which improves serendipity of recommendations through feature diversification and helps overcome the overspecialization problem. To evaluate our algorithm, we employed the only publicly available dataset containing user feedback regarding serendipity. We compared our SOG algorithm with topic diversification, popularity baseline, singular value decomposition, serendipitous personalized ranking and Zheng's algorithms relying on the above dataset. SOG outperforms other algorithms in terms of serendipity and diversity. It also outperforms serendipity-oriented algorithms in terms of accuracy, but underperforms accuracy-oriented algorithms in terms of accuracy. We found that the increase of diversity can hurt

Denis Kotkov
deigkotk@student.jyu.fi

Jari Veijalainen
jari.veijalainen@jyu.fi

Shuaiqiang Wang
wangshuaiqiang1@jd.com

[1] University of Jyvaskyla, Faculty of Information Technology, P.O.Box 35, FI-40014 University of Jyvaskyla, Jyvaskyla, Finland
[2] Data science lab, JD.com, Beijing, China (The research was conducted while the author was working for the University of Jyvaskyla, Finland.)

accuracy and harm or improve serendipity depending on the size of diversity increase.

Keywords Recommender Systems, Learning to Rank, Serendipity, Novelty, Unexpectedness, Algorithms, Evaluation, Serendipity-2018

1 Introduction

Recommender systems are software tools that suggest items of use to users [27, 17]. An item is "a piece of information that refers to a tangible or digital object, such as a good, a service or a process that a recommender system suggests to the user in an interaction through the Web, email or text message" [17]. For example, an item could refer to a movie, a song or a new friend.

To increase the number of items that will receive high ratings most recommender systems tend to suggest items that are (a) popular, as these items are consumed by many individuals and are often of high quality in many domains [5] and (b) similar to those the user has assigned high ratings, as these items correspond to users' preferences [29,17,19]. As a result, users might become bored with the suggestions provided, as (1) users are likely to be familiar with popular items, while the main reason these users would use a recommender system is to find novel and relevant items [5] and (b) users often lose interest in using the system when they are offered only items similar to items from their profiles (the so-called overspecialization problem) [29,17,19,18]. Here the term *user profile* refers to the unique ID and the set of items rated by the target user [17], though it might include information, such as real name, user name and age in other papers.

To suggest novel and interesting items and overcome the overspecialization problem, recommender systems should suggest serendipitous items. Some researchers consider novel and unexpected items serendipitous [32], while others suggest that serendipitous items are relevant and unexpected [22]. Although there is no agreement on the definition of serendipity [19], in this paper, the term *serendipitous items* refers to items relevant, novel and unexpected to a user [17,19,18]:

- An item is *relevant* to a user if the user has expressed or will express preference for the item. The user might express his/her preference by liking or consuming the item depending on the application scenario of a particular recommender system [17,19]. In different scenarios, ways to express preference might vary. For example, we might regard a movie as relevant to a user if the user gave it more than 3 stars out of 5 [33,21], whereas we might regard a song as relevant to a user if the user listened to it more than twice. The system is aware that a particular item is relevant to a user if the user rates the item, and unaware of its relevance otherwise.
- An item is *novel* to a user if the user had not heard of this item or had not thought of this item prior to the recommendation of this item [16]. Items novel to a user are usually unpopular, as users are often familiar

with popular items, where popularity can be measured by the number of ratings given to it in the system [17,19,18,5]. For example, a user is more likely to be familiar with the popular movie "The Shawshank Redemption" than with the unpopular movie "Coherence". Novel items also have to be relatively dissimilar to a user profile, as the user is likely to be familiar with items similar to the ones she/he has rated [17,19]. For example, a rock fan is more likely to be familiar with rock songs rather than with pop songs.

– An item is *unexpected* to a user if the user does not anticipate this item to be recommended to him/her or found by him/her or this item is just very dissimilar to what this user usually consumes [16]. The user does not expect items that are dissimilar to the ones usually recommended to him/her. Generally, recommender systems suggest items similar to items rated by the user [29,17,19]. Consequently, an item dissimilar to the rated ones is regarded as unexpected [17,19]. The measure of dissimilarity could be based on user ratings or item attributes depending on the application scenario of a recommender system [13]. For example, a comedy fan would mostly rate comedies and receive recommendations of comedies in a recommender system. A recommendation of documentary would be unexpected to the user, as the user does not expect a recommendation of this genre from this particular recommender system.

An item must be relevant, novel and unexpected at the same time to be considered serendipitous. For example, a recommender system suggests a very unpopular documentary to a comedy fan. Let us assume that the user has never heard of this movie (novel, as it is very unpopular) and does not expect this movie to be recommended to them (unexpected, as it does not match their tastes indicated in the system). Let us also assume that the user watches the movie and enjoys it (relevant). We will regard this movie serendipitous, as it was novel and unexpected to the user prior to the recommendation and it turned out to be relevant to her afterwards.

State-of-the-art serendipity-oriented recommendation algorithms are barely compared with one another and often employ different serendipity metrics and definitions of the concept, as there is no agreement on the definition of serendipity in recommender systems [32,21,19].

In this paper, we propose a serendipity-oriented recommendation algorithm based on our definition above. We compare our algorithm with state-of-the-art serendipity-oriented algorithms relying on the first and currently the only publicly available dataset containing user feedback regarding serendipity.

Our serendipity-oriented algorithm reranks recommendations provided by an accuracy-oriented algorithm and improves serendipity through feature diversification. The proposed algorithm is based on the existing reranking algorithm, topic diversification (TD) [34], and outperforms this algorithm and other algorithms in terms of serendipity and diversity. Our algorithm also outperforms the state-of-the-art serendipity-oriented algorithms in terms of accuracy.

Our algorithm has the following advantages:

- It considers each component of serendipity.
- It improves both serendipity and diversity.
- It can be applied to any accuracy-oriented algorithm.

The paper has the following contributions:

- We propose a serendipity-oriented recommendation algorithm.
- We evaluate existing serendipity-oriented recommendation algorithms.
- We investigate the effect of diversity on accuracy and serendipity.

The rest of the paper is organized as follows. Section 2 discusses earlier work in the field. Section 3 describes the proposed algorithm. Section 4 is dedicated to experimental setting, while Section 5 reports the results of the experiments. Finally, Section 7 draws conclusions.

2 Related works

In this section, we will discuss definitions of serendipity and diversity, and overview algorithms that improve these properties.

2.1 Definition of Serendipity

The term *serendipity* was coined by Horace Walpole by referencing a Persian fairytale, "The Three Princess of Serendip," in 1758. In the fairytale, the three princes of the country Serendip ventured out to explore the world and made many unexpected discoveries on their way[1]. In his letter, Horace Walpole mentioned that the princes were "always making discoveries, by accidents & sagacity, of things which they were not in quest of" [26].

The dictionary definition of serendipity is "the faculty of making fortunate discoveries by accident"[2]. However, there is no consensus on the definition of serendipity in recommender systems. Some researchers require items to be relevant and unexpected to be considered serendipitous [22,19], whereas other researchers suggest that serendipitous items are novel and unexpected [32, 19]. However, the most common definition of serendipity includes all three components: relevance, novelty and unexpectedness [18,17].

Novelty and unexpectedness also have multiple definitions, which results in eight variations of serendipity [16]. Novelty has two variations: strict novelty - the user has never heard about an item; and motivational novelty - the user had not thought of consuming an item, before this items was recommended to him/her. Unexpectedness has four variations: unexpectedness (relevant) - the user does not expect to enjoy the item; unexpectedness (find) - the user does not expect to find the item on his/her own; unexpectedness (implicit) - the item is very dissimilar to what the user usually consumes; and unexpectedness (recommend) - the user does not expect the item to be recommended

[1] "The Three Princes of Serendip" by Boyle Richard, 2000.

[2] http://www.thefreedictionary.com/serendipity

to him/her. Relevance has one variation and indicates how much the user enjoys consuming the item. Since serendipity consists of relevance, novelty and unexpectedness, relevance has one variation, novelty has two variations and unexpectedness has four variations, there are eight variations of serendipity (proposed in [16]): strict serendipity (relevant), strict serendipity (find), strict serendipity (implicit), strict serendipity (recommend), motivational serendipity (relevant), motivational serendipity (find), motivational serendipity (implicit) and motivational serendipity (recommend). We employ these variations in this study.

2.2 Improving Serendipity

There are three categories for serendipity-oriented algorithms [19]: (a) reranking algorithms (these algorithms change the order of items in recommendation lists using relevance scores provided by accuracy-oriented algorithms); (b) serendipity-oriented modifications (these algorithms are based on particular accuracy-oriented algorithms); and (c) novel algorithms (these algorithms are not based on any common accuracy-oriented algorithms, but rather utilize different techniques to improve serendipity).

Reranking algorithms improve serendipity by changing the order of the output of accuracy-oriented algorithms [19]. These algorithms often use relevance scores to filter out potentially irrelevant items first and then use other techniques to promote potentially serendipitous ones. For example, the algorithm proposed by Adamopoulos and Tuzhilin first filters out items likely to be irrelevant and obvious to a user and then orders items based on their overall utility for the user. The latter is based on how different an item is to users' expectations and on relevance scores for this item provided by an accuracy-oriented algorithm [1]. Another example is the algorithm proposed by Zhang et. al, Auralist [32]. The algorithm consists of the three other algorithms: Basic Auralist, which is responsible for relevance scores, Listener Diversity, which is responsible for diversity, and Declustering, which is responsible for unexpectedness. The algorithm orders items in the recommendation list according to the final score, which is represented by a linear combination of the scores provided by the three algorithms.

Serendipity-oriented modifications refer to common accuracy-oriented algorithms modified with a purpose of increasing serendipity [19]. The main difference between reranking algorithms and modifications is that modifications are always based on particular accuracy-oriented algorithms, whereas a particular reranking process can be applied to any accuracy-oriented algorithm, which provides relevance scores. For example, Nakatsuji et al. modified a common user-based collaborative filtering algorithm (k-nearest neighbor algorithm) [9] by replacing the user similarity measure with relatedness. It is calculated using random walks with restarts on a user similarity graph [23]. The graph consists of nodes corresponding to users and edges corresponding to similarities based on an item taxonomy. By utilizing the relatedness, for a

target user, the algorithm picks a neighborhood of users who are not necessarily similar, but who are in some way related to the target user [23]. Another example of modifications is the algorithm proposed by Zheng et al. The algorithm is based on PureSVD (a variation of the singular value decomposition algorithm) [7]. The main difference between PureSVD and its modification is that the objective function of the modification includes components responsible for unexpectedness, whereas the objective function of PureSVD lacks these components [33].

Novel serendipity-oriented algorithms neither fall into reranking nor into modifications categories, as they are not based on any common accuracy-oriented algorithms and do not use relevance scores provided by any accuracy-oriented algorithms [19]. For example, TANGENT recommends items using relevance scores and bridging scores, where both kinds of scores are inferred using a bipartite graph [24]. The graph contains nodes that represent users and items, and edges that represent ratings. The algorithm calculates relevance scores using random walks with restarts and bridging scores based on the calculated relevance scores [24]. Another example of an algorithm that belongs to the category of novel algorithms is random walk with restarts enhanced with knowledge infusion [10]. The algorithm orders items in recommendation lists according their relatedness to a user profile. The relatedness is calculated using random walks with restarts on an item similarity graph, where nodes correspond to items, and edges correspond to similarities between these items. To calculate the similarities, the authors used the spreading activation network based on Wikipedia and WordNet [10].

Most existing algorithms have been designed to achieve serendipity measured by artificial evaluation metrics due to the lack of publicly available datasets containing user feedback regarding serendipity. The results of artificial evaluation metrics might be misleading, as the assumptions that these metrics are based on might not correspond to the reality due to the lack of ground truth [15]. Furthermore, performance of most existing algorithms is not compared with that of others [18]. In this article, we propose a reranking algorithm and compare it with state-of-the-art algorithms in evaluation conducted on the first publicly available dataset containing serendipity ground truth.

2.3 Definition of Diversity

Diversity is a property of a recommendation list or a set of them composed by one or several recommender systems. It reflects how dissimilar items are to each other in the list [4,14]. To measure diversity inside a list, researchers often calculate an average pairwise dissimilarity of items in a recommendation list [4,14], where dissimilarity can be represented by any metric, which reflects how dissimilar items are to one another. A dissimilarity metric is often based on attributes of items. The higher the average pairwise dissimilarity, the higher the diversity of the list.

Diversity is considered as a desirable property of a recommender system, as it was proven to improve user satisfaction [34], and by diversifying the recommendation results, we are likely to suggest an item satisfying a current need of a target user [14]. A fan of the movie *The Matrix* is likely to prefer a recommendation list of movies similar to *The Matrix*, including this movie, rather than a recommendation list consisting of *The Matrix* sequels only.

Diversity is not always related to dissimilarity of items in a particular recommendation list. The term can also refer to diversity of recommendations provided by different recommender systems [3], diversity across recommendation lists suggested to all the users of a particular system [2], or diversity of recommendations to the same user in a particular system over time [20]. In these cases one needs a more complicated diversity measure than the pairwise average diversity of items inside one recommendation list.

2.4 Improving Diversity

Greedy reranking algorithms are very common in improving diversity of recommendation lists. They create two lists of items (a candidate list and a recommendation list), and iteratively move items from the candidate list to the recommendation list [14,4]. In each iteration, these algorithms calculate different scores, which depend on the algorithm. Based on these scores, the algorithms pick an item from the candidate list to be moved to the recommendation list [14,4]. For example, the TD algorithm, which our algorithm is based on, calculates in each iteration average similarities between each item in the candidate list and items in the recommendation list and uses the obtained scores to pick an item that is the most relevant but at the same time the most dissimilar to the items already added to the recommendation list [34].

Another group of the algorithms optimized for diversity take diversity into account in the process of generating recommendations. For example, Su et al. proposed an algorithm that integrates diversification in a traditional matrix factorization model [28]. Another example of an algorithm falling into this category is diversified collaborative filtering algorithm (DCF) that employs a combination of support vector machine and parametrized matrix factorization to generate accurate and diversified recommendation lists [6].

To the best of our knowledge, studies that focus on both serendipity and diversity are very limited. In this article, we propose an algorithm that improves both serendipity and diversity.

3 A Serendipity-Oriented Greedy Algorithm

To describe the proposed algorithm, we present the notation in Table 1. Let I be a fixed set of available items and U be a fixed set of users of a particular recommendation system at a particular point in time T. User u has rated or interacted with items $I_u, I_u \subseteq I$. The recommender system suggests $RS_u(n)$

Table 1: Notations

Symbol	Description
$I = \{i_1, i_2, ..., i_{\|\|I\|\|}\}$	the set of items
$I_u, I_u \subseteq I$	the set of items rated by user u (user profile)
$U = \{u_1, u_2, ..., u_{\|\|U\|\|}\}$	the set of users
$U_i, U_i \subseteq U$	the set of users who rated item i
$RS_u(n), RS_u(n) \subseteq I$	the set of top–n recommendations provided by an algorithm to user u
r_{ui}	the rating given by user u to item i
$\hat{r}_{ui}$	the prediction of the rating given by user u to item i

items to user u (at time T). The unique rating user u has given to item i before T is represented by r_{ui}, whereas the predicted rating generated by an algorithm is represented by $\hat{r}_{ui}$.

3.1 Description

We propose a SOG algorithm [18] that is based on a TD algorithm [34]. The objective of TD is to increase the diversity of a recommendation list. Both SOG and TD belong to the group of greedy reranking algorithms, meaning that these algorithms change the order of items provided by another algorithm [4]. According to the classification provided in [19], we propose a hybrid reranking algorithm following the post-filtering paradigm and considering unpopularity and dissimilarity.

Input : $RS_u(n)$: top–n recommendation set
Output: Res: picked item list
B: candidate set,
$\hat{r}_{ui}$: predicted rating of an item,
$Res[0] \leftarrow i$ with max $\hat{r}_{ui}$;
for $z \leftarrow 1$ *to* n **do**
  $B \leftarrow set(Res)$;// *set* converts a list to a set
  $B' \leftarrow RS_u(n) \backslash B$;
  forall $i \in B'$ **do**
    calculate $score_{uiB}$
  end
  $Res[z] \leftarrow i$ with max $score_{ui}$;
end

Algorithm 1: Description of SOG

Algorithm 1 describes the proposed approach. An accuracy-oriented algorithm predicts item ratings $\hat{r}_{u,i}$ and generates top–n suggestions $RS_u(n)$ for user u. SOG iteratively picks items from the set corresponding to $RS_u(n)$ to fill diversified list Res. In each iteration the algorithm generates a candidate set B' that contains top–n recommendations $RS_u(n)$ except items already picked to the list Res (converted to the set B). A candidate item with the highest score is added to the diversified list Res. The result Res contains the same items as $RS_u(n)$, but in a (possibly) different order.

The score is a linear combination of parameters important for serendipity:

$$score_{uiB} = \sum_{j=1}^{h} \alpha_j \cdot p_{uiBj} \quad . \tag{1}$$

where α_j is the weight of parameter p_{uiBj}, while h is the number of parameters. In our algorithm, we took into account four parameters: relevance, diversity, dissimilarity of an item to the user profile and unpopularity, which resulted in the following equation:

$$score_{uiB} = \alpha_{rel} \cdot \hat{r}_{ui} + \alpha_{div} \cdot div_{iB} + \alpha_{prof} \cdot prof_{ui} + \alpha_{unpop} \cdot unpop_i \quad . \tag{2}$$

where div_{iB} indicates the average dissimilarity of item i and the candidate set B, $prof_{ui}$ represents the average dissimilarity of item i to items consumed by user u in the past (I_u) and $unpop_i$ indicates unpopularity of item i. The diversity parameter is calculated as follows [33]:

$$div_{iB} = \frac{1}{||B||} \sum_{j \in B} 1 - sim_{i,j} \quad , \tag{3}$$

where similarity $sim_{i,j}$ can be any kind of similarity measure varying in the range $[0, 1]$. The dissimilarity to the user profile is calculated as follows:

$$prof_{ui} = \frac{1}{||I_u||} \sum_{j \in I_u} 1 - sim_{i,j} \quad . \tag{4}$$

In case the user has not rated any items yet (the cold start problem [11]), $prof_{ui}$ can be set to any constant (in our experiments we picked 0), as the contribution of this part of equation 2 will be the same for each candidate item. It is possible to calculate the dissimilarity of an item to a user profile based on information about user, such as location, age or gender, but this information is often unavailable and this extension is beyond the scope of this article. The unpopularity parameter is based on the number of ratings, assigned to a particular item:

$$unpop_i = \frac{||U_i||}{||U||} \quad . \tag{5}$$

Each of the four parameters is normalized to the range $[0, 1]$. The three parameters $\hat{r}_{ui}$, $prof_{ui}$ and $unpop_i$ can be calculated prior to the first iteration

of the algorithm, while div_{Bi} can only be calculated on each iteration of the algorithm, as it depends on the candidate set B.
Although SOG is based on TD [34], our algorithm has two key differences with respect to TD:

- SOG considers item scores instead of positions of items in lists, which leads to more accurate scores ($score_{ui}$).
- SOG takes into account parameters important for serendipity.

Our algorithm has four main advantages:

- The algorithm considers each component of serendipity.
- As our algorithm is based on the diversification algorithm, SOG improves both serendipity and diversity.
- As SOG is a reranking algorithm, it can be applied to any accuracy-oriented algorithm, which might be useful for a live recommender system (reranking could also be conducted on the client's side in a client-server application scenario).
- Our algorithm employs four weights that allow to control serendipity. The weights could be different for each user and be adjusted as the user becomes familiar with the system.

3.2 Computational complexity

The algorithm contains three loops (algorithm 1): the loop from 1 to n, the loop from 1 to $||B'||$, and the loop from 1 to $||B||$ to calculate div_{Bi} (eq. 3). The overall number of actions can be calculated as follows:

$$\begin{aligned}(n-1)\cdot 1 + (n-2)\cdot 2 + (n-3)\cdot 3 + ... + (n-n)\cdot n = \\ = n\cdot(1+2+3+...+n) - (1^2+2^2+3^2+...+n^2) = \\ = n\cdot n\cdot\frac{1+n}{2} - \frac{n(n+1)(2n+1)}{6} = \frac{n^3-2n^2+n}{6} \quad ;\end{aligned} \tag{6}$$

$$\mathcal{O}\left(\frac{n^3-2n^2+n}{6}\right) = \mathcal{O}(n^3) \quad . \tag{7}$$

The computational complexity of the algorithm is $\mathcal{O}(n^3)$ (excluding pre-calculation), where n is the number of items in the input set $RS_u(n)$. The complexity of our algorithm is relatively high. However, we expect n to be relatively small (in our experiments $n = 10$), as by increasing n, one increases the chance of suggesting irrelevant (and therefore non-serendipitous) items to the user.

In cases when n is required to be very high, which makes the computational time unacceptable, one might want to ignore $div_{i,B}$ in equation 2, which will decrease the computational complexity to $\mathcal{O}(n^2)$. This will also decrease the diversification effect of the algorithm, but keep the improvement of serendipity.

4 Experiments

In this section, we present the dataset we used in our experiments, baseline algorithms and evaluation metrics.

4.1 Dataset

To compare performance of our algorithm with the baselines, we evaluated these algorithms on the Serendipity-2018 dataset, as to the best of our knowledge, this is the only publicly available dataset, which contains user feedback regarding serendipity [16]. As the amount of this feedback is limited, we generated additional user feedback based on this dataset. We then split this dataset into three different datasets to evaluate our baselines. In this section, we first describe the dataset and then provide details on its preprocessing.

4.1.1 Description

Serendipity-2018 contains ratings given by users to movies on the movie recommender system MovieLens[3], where users rate movies they watched in the past on the scale from 0.5 to 5 stars and receive recommendations of movies to watch based on their ratings. The authors of the dataset conducted a survey in MovieLens, where they asked users how serendipitous these users find particular movies. In the survey, the authors selected movies that were assigned low number of ratings in the system (unpopular movies) and given high ratings by the users (relevant movies), as these movies were likely to be serendipitous to the users.

The authors proposed eight variations of serendipity and asked users to indicate how serendipitous each movie was to them according to each of the eight variations (see sec. 2.1). The dataset thus contains eight binary variables indicating whether a movie is serendipitous or not according to a particular variation.

The dataset contains two types of user feedback: relevance ratings and serendipity ratings. Relevance ratings are 5-star ratings that indicate how much users enjoyed watching the movies. Serendipity ratings are binary ratings that indicate whether users considered movies as serendipitous or as non-serendipitous. Serendipity-2018 contains 10 million relevance ratings given by 104,661 users to 49,151 different movies and 2,150 serendipity ratings given by 481 users (up to five serendipity ratings per user) to 1,678 different movies.

4.1.2 Preprocessing

In this experiment, we targeted the union of six variations of serendipity, as the two remaining variations are likely to reduce user satisfaction [16]. A movie

[3] https://movielens.org

was considered as serendipitous to a user, if this movie was serendipitous to the user according to at least one of the six remaining serendipity variations: strict serendipity (find), strict serendipity (implicit), strict serendipity (recommend), motivational serendipity (find), motivational serendipity (implicit) and motivational serendipity (recommend). For detailed discussion the reader is urged to consult [16] and section 2.1.

We generated a number of serendipity ratings due to the lack of these ratings in the dataset. For each of these users, we randomly selected five movies rated by the user with a relevance rating and labeled these movies non-serendipitous for this user. We regarded these movies non-serendipitous, as they were unlikely to be serendipitous to the users. According to the dataset, the chance of a movie to be serendipitous to the user is up to 13%[4], provided that the authors of the dataset selected movies that were likely to be serendipitous in their survey. These movies were relevant to the users, as users gave them high relevance ratings and likely to be novel, as these movies had relatively low number of ratings in MovieLens. To randomly select relevance ratings and label them non-serendipitous, we did not control for popularity or relevance. The chance of making a mistake labeling a movie non-serendipitous to the user is thus much lower than 13%. The final dataset contained 4,555 serendipity ratings (2,405 were generated) given by 481 users to 1,931 different movies and 10 million relevance ratings given by 104,661 users (including the 481 users) given to 49,151 different movies.

To tune and evaluate the baselines, we split the final dataset into three datasets: the training dataset, the tuning dataset and the test dataset. The training dataset contains almost 10 million relevance ratings, while the tuning dataset contains 3,043 relevance and serendipity ratings (67% of serendipity ratings) of the same user-movie pairs. The test dataset contains 1,512 relevance and serendipity (33% of serendipity ratings) ratings of the same user-movie pairs. To tune the parameters of the baselines, we trained them on the relevance ratings of the training dataset and tuned the parameters based on the performance of these baselines on the serendipity ratings of the tuning dataset. We then trained the baselines with the inferred parameters on relevance ratings of the training and tuning datasets combined and evaluated them on the relevance (to measure relevance) and serendipity (to measure serendipity) ratings of the test dataset.

4.1.3 Similarity measure

To calculate how similar movies are to each other, the algorithms evaluated in this research require a similarity measure. We picked tag based similarity measure, since this is an important factor for detecting serendipitous movies in Serendipity-2018 [16]. Tag based similarity measure is based on tag genome data [31], which in turn is based on tags that users assign to movies in MovieLens. The tags are the keywords that indicate different features of movies.

[4] 277 out of 2,150 user-movie pairs are serendipitous

For example, the movie "The Shawshank Redemption" is attached tags "narrated", "prison" and "escape". Tag genome contains most popular tags and scores indicating how much each of these tags applies to a particular movie. These scores vary in the range $[0, 1]$ and are calculated based on the number of users who assigned these tags, user ratings and other user-contributed data. To calculate similarity between movies, we employed weighted cosine similarity [31]:

$$sim(i,j) = \frac{\sum_{k=1}^{m} i_k \cdot j_k \cdot w_k}{\sqrt{\sum_{k=1}^{m} i_k^2 \cdot w_k}\sqrt{\sum_{k=1}^{m} j_k^2 \cdot w_k}} \quad , \tag{8}$$

where m is the number of tags, i_k indicates the score of tag k applied to movie i, while w_k indicates the weight of tag k. The weight is calculated as follows:

$$w_k = \frac{\log(||U_{t_k}||)}{\log||I_{t_k}||} \quad , \tag{9}$$

where U_{t_k} corresponds to the set of users, who assigned tag t_k, while I_{t_k} corresponds to the set of movies, for which the tag score is greater than 0.5 ($i_k > 0.5$).

4.2 Baselines

We implemented the following baseline algorithms:

- **POP** ranks items according to the number of ratings each item received in descending order.
- **SVD** is a singular value decomposition algorithm that ranks items according to generated scores [33]. The objective function of the algorithm is the following:

$$min \quad \sum_{u \in U} \sum_{i \in I_u} (r_{ui} - p_u \cdot q_i^T)^2 + \beta(||p_u||^2 + ||q_i||^2) \quad , \tag{10}$$

  where p_u and q_i are user-factor vector and item-factor vector, respectively, while $\beta(||p_u||^2 + ||q_j||^2)$ represents the regularization term. Based on tuning, we picked the following parameters: feature number=200, learning rate=10^{-5} and regularization term=0.1.
- **SPR** (serendipitous personalized ranking) is an algorithm based on SVD that maximizes the serendipitous area under the ROC (receiver operating characteristic) curve [21]:

$$max \quad \sum_{u \in U} f(u) \quad , \tag{11}$$

$$f(u) = \sum_{i \in I_u^+} \sum_{j \in I_u \setminus I_u^+} \cdot z_u \cdot \sigma(0, \hat{r}_{ui} - \hat{r}_{uj})(||U_j||)^{\alpha} \quad , \tag{12}$$

  where I_u^+ is a set of items a user likes. We considered that a user likes items that she/he rates higher than threshold θ (in our experiments $\theta = 3$).

Normalization term z_u is calculated as follows: $z_u = \frac{1}{||I_u^+|| ||I_u \setminus I_u^+||}$. Based on tuning, we picked the following parameters: Bayesian loss function, $\alpha = 0.4$, feature number=200, learning rate=10^{-5} and regularization term=0.1.

– **Zheng's** is an algorithm based on SVD that considers observed and unobserved ratings and weights the error with unexpectedness [33]:

$$min \quad \sum_{u \in U} \sum_{i \in I_u} (r_{ui} - p_u \cdot q_i^T)^2 \cdot w_{ui} + \\ + \beta(||p_u||^2 + ||q_i||^2) \quad , \tag{13}$$

$$w_{ui} = \left(1 - \frac{||U_i||}{max_{j \in I}(||U_j||)}\right) + \frac{\sum_{j \in I_u} \cdot diff(i,j)}{||I_u||} \quad , \tag{14}$$

where $max_{j \in I}(||U_j||)$ is the maximum number of ratings given to an item. A collaborative dissimilarity between items i and j is represented by $diff(i,j)$. The dissimilarity is calculated as $diff(i,j) = 1 - \rho_{i,j}$, where similarity $\rho_{i,j}$ corresponds to the Pearson correlation coefficient:

$$\rho_{i,j} = \frac{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in S_{i,j}} (r_{uj} - \overline{r}_u)^2}} \quad , \tag{15}$$

where $S_{i,j}$ is the set of users rated both items i and j, while $\overline{r}_u$ corresponds to an average rating for user u. In our implementation, we excluded unobserved ratings due to the size of our dataset. Based on tuning, we picked the parameters: feature number=200, learning rate=10^{-5} and regularization term=0.1.

– **TD** is a topic diversification algorithm, where similarity corresponds to eq. 8 and the ratings are predicted by SVD [34]. Based on tuning, we set $\Theta_F = 0.9$.

– **SOG** is the proposed serendipity-oriented greedy algorithm, where the ratings are predicted by SVD. Based on tuning, we set $\alpha_{rel} = 0.9$, $\alpha_{div} = 0.1$, $\alpha_{prof} = 0.7$ and $\alpha_{unpop} = 0.7$.

4.3 Evaluation metrics

The main objective of our algorithm is to improve serendipity of a recommender system. A change of serendipity might affect other properties of a recommender system. To demonstrate the dependence of different properties and features of the baselines, we employed evaluation metrics to measure three properties of recommender systems: accuracy, as it is a common property [19]; serendipity, as SPR, Zheng's and SOG are designed to improve this property [21,33]; and diversity, as this is one of the objectives of TD [34]:

– To measure a ranking ability of an algorithm, we use normalized discounted cumulative gain (NDCG), which, in turn, is based on discounted cumulative

Table 2: Serendipity

Algorithm	Serendipity@1	Serendipity@3	Serendipity@5
POP	0.021	0.021	0.051
Random	0.170	0.149	0.140
TD	0.277	0.227	0.226
SVD	0.277	0.241	0.213
Zheng's	**0.319**	0.248	0.204
SPR	**0.319**	0.291	**0.268**
SOG	0.277	**0.305**	0.230

gain (DCG) [12]:

$$DCG_u@n = rel_u(1) + \sum_{i=2}^{n} \frac{rel_u(i)}{log_2(pos(i))} \quad , \tag{16}$$

where $rel_u(i)$ indicates relevance of item i with rank $pos(i)$ for user u, while n indicates the number of top recommendations selected. $pos(i)$ is the distance of the item from the beginning of the list (1, 2, 3, ..., n). The NDCG metric is calculated as follows:

$$NDCG_u@n = \frac{DCG_u@n}{IDCG_u@n} \quad , \tag{17}$$

where $IDCG_u@n$ is $DCG_u@n$ value calculated for a recommendation list with an ideal order according to relevance.

– To measure serendipity, we adopted the accuracy metric precision, since user feedback regarding serendipity is the binary variable:

$$Serendipity_u@n = \frac{ser_u@n}{n} \quad , \tag{18}$$

where $ser_u@n$ corresponds to the number of serendipitous items in the first n results. To tune our algorithms, we used Serendipity@3.

– To measure diversity, we employed an intra-list dissimilarity metric [33]:

$$Div_u@n = \frac{1}{n \cdot (n-1)} \sum_{i \in RS_u(n)} \sum_{j \neq i \in RS_u(n)} 1 - sim_{i,j} \quad , \tag{19}$$

where similarity $sim_{i,j}$ corresponds to tag based similarity measure (eq. 8).

The experiments were conducted using the Lenskit framework[5] [8].

5 Results

Tables 2, 3 and 4 demonstrate performance of baselines in terms of accuracy, serendipity and diversity. The following observations can be noticed (we provide them along with explanations):

[5] https://lenskit.org/

Table 3: Accuracy

Algorithm	NDCG@1	NDCG@3	NDCG@5
POP	0.832	0.842	0.864
Random	0.823	0.850	0.878
Zheng's	0.855	0.884	0.902
SPR	0.850	0.886	0.905
SOG	**0.881**	0.887	0.899
TD	**0.881**	0.898	0.920
SVD	**0.881**	**0.903**	**0.921**

Table 4: Diversity

Algorithm	Div@5
SVD	0.347
Zheng's	0.347
POP	0.353
SPR	0.356
TD	0.359
Random	0.367
SOG	**0.371**

1. Serendipity (table 2)
   1.1. SOG outperforms other algorithms at top-3 results
   1.2. SOG underperforms SPR at top-1, as our algorithm always keeps the most relevant item as the first one in the list
   1.3. SOG underperforms SPR at top-5, as our algorithm was tuned for top-3 results
   1.4. POP demonstrates the lowest performance among the presented baselines, as the most popular items are the most well-known and the least surprising to the users [19,33,32,18]
   1.5. The serendipity-oriented algorithms Zhengs and SPR outperform the accuracy-oriented algorithm SVD, as the serendipity-oriented algorithms were designed to achieve high serendipity
   1.6. Personalized algorithms (Zheng's, SPR, TD, SVD and SOG) outperform non-personalized ones (POP and Random)
2. Accuracy (table 3)
   2.1. SOG underperforms the accuracy-oriented algorithms SVD and TD, and outperforms the serendipity-oriented algorithms SPR and Zheng's due to the objective of our algorithm
   2.2. Personalized algorithms (Zheng's, SPR, TD, SVD and SOG) outperform non-personalized ones (POP and Random)
   2.3. The accuracy-oriented algorithm SVD outperforms serendipity-oriented ones (SOG, SPR and Zheng's), as SVD was optimized for serendipity, while the other algorithms were optimized for serendipity
3. Diversity (table 4)
   3.1. SOG outperforms other algorithms in terms of diversity

Observations 1.1, 2.1 and 3.1 suggest that our algorithm has the highest serendipity and diversity among the presented algorithms, the highest accuracy among serendipity-oriented algorithms. As expected, the popularity baseline has the lowest serendipity (observation 1.4), personalized algorithms have higher performance in terms of accuracy and serendipity than non-personalized ones (observations 1.6 and 2.2), algorithms optimized for serendipity outperform those optimized for accuracy in terms of serendipity (observation 1.5) and algorithms optimized for accuracy outperform those optimized for serendipity in terms of accuracy (observation 2.3).

According to observations 1.1 and 3.1, serendipity and diversity are properties that can be increased simultaneously. Meanwhile observations 1.1 and 2.1 indicate that the increase of serendipity can cause the decrease of accuracy.

5.1 Investigating the effect of diversity

To investigate the effect of diversity on serendipity and accuracy, we run TD multiple times varying the damping factor from 0 till 0.95. We picked TD for the sake of simplicity.

Figure 1 demonstrates the performance of TD in terms of diversity, accuracy and serendipity. The figure suggests that with the increase of damping factor, diversity increases, accuracy decreases, serendipity increases, achieves its peak and then decreases. These observations indicate that serendipity can be increased along with diversity for the relatively low price of accuracy. For example, by increasing the damping factor from 0 to 0.35, one can achieve the increase of diversity from 0.346 to 0.366 (5.4%), the increase of serendipity from 0.212 to 0.217 (1.9%) and the decrease of accuracy from 0.921 to 0.918 (0.3%).

6 Discussion

In our experiments, we only considered the movie domain, as the only publicly available serendipity dataset contains information on movies. In other domains, the findings might be different. In fact, in some domains and situation, serendipity either might not be suitable or might need to be redefined. For example, generating a play list based on the number of songs or keywords might not require any serendipity [30]. The investigation of the effect serendipity in other domains required user studies and datasets from these domains.

In our experiments, we assumed that the number of candidate items n is relatively small (around 20), as with the increase of n, our algorithm is likely to pick items irrelevant to the user, which is likely to repulse him/her. This assumption is reasonable when serendipitous recommendations are mixed with non-serendipitous ones. However, in the situation, when a recommender system needs to suggest serendipitous items to the user regardless of the number of irrelevant ones ("surprise me" option), n might need to be high, which would

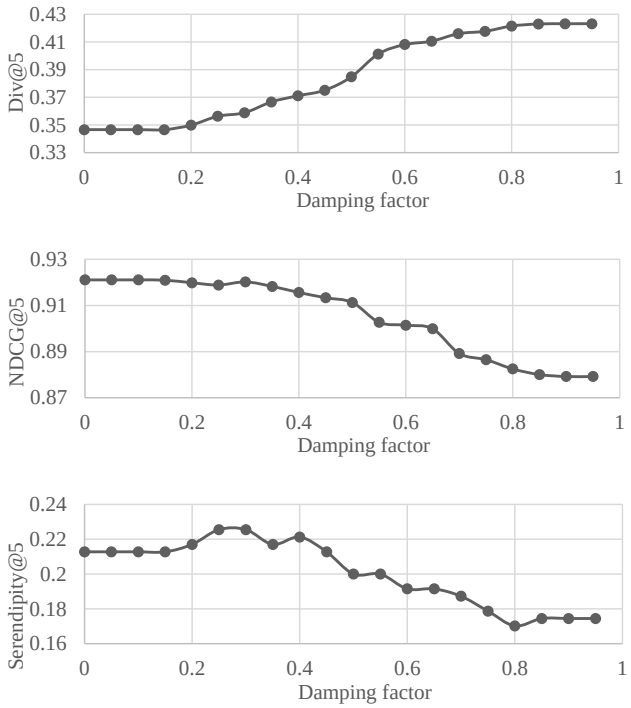

Fig. 1: Performance of TD with Θ_F (damping factor) varying from 0 to 0.95

significantly increase the time to generate recommendations. A solution in this situation might be to choose another baseline algorithm, such as SPR.

7 Conclusion and Future Work

We proposed serendipity-oriented greedy (SOG) algorithm, provided evaluation results of our algorithm and state-of-the-art algorithms on the only publicly available dataset that contains user feedback regarding serendipity. We also investigated the effect of diversity on accuracy and serendipity.

According to our results, our algorithm outperforms other algorithms in terms of serendipity and diversity, serendipity-oriented algorithms in terms of accuracy, but underperforms accuracy-oriented algorithms in terms of accuracy.

We found that accuracy, serendipity and diversity are not independent properties of recommender systems. The increase of diversity can hurt accuracy and hurt or improve serendipity depending on size of the increase.

In our future work, we are planning to further investigate serendipity by designing serendipity-oriented algorithms and evaluating them with real users. Having a bigger dataset on serendipity might provide insights on serendipity.

Deep learning seems to be a promising direction for designing serendipity-oriented algorithms [25]. User studies might help to further investigate the effect of serendipity on users and the performance of algorithms in terms of user satisfaction.

8 ACKNOWLEDGEMENT

The research at the University of Jyväskylä was performed in the MineSocMed project, partially supported by the Academy of Finland, grant #268078 and the KAUTE Foundation.

References

1. Adamopoulos, P., Tuzhilin, A.: On unexpectedness in recommender systems: Or how to better expect the unexpected. ACM Transactions on Intelligent Systems and Technology **5**(4), 1–32 (2014)
2. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Transactions on Knowledge and Data Engineering **24**(5), 896–911 (2012)
3. BellogíN, A., Cantador, I., Castells, P.: A comparative study of heterogeneous item recommendations in social systems. Information Sciences **221**, 142–169 (2013)
4. Castells, P., Hurley, N.J., Vargas, S.: Novelty and diversity in recommender systems. In: Recommender Systems Handbook, pp. 881–918. Springer (2015)
5. Celma Herrada, Ò.: Music recommendation and discovery in the long tail. Ph.D. thesis, Universitat Pompeu Fabra (2009)
6. Cheng, P., Wang, S., Ma, J., Sun, J., Xiong, H.: Learning to recommend accurate and diverse items. In: Proceedings of the 26th International Conference on World Wide Web, pp. 183–192. International World Wide Web Conferences Steering Committee (2017)
7. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 39–46. ACM, New York, NY, USA (2010). DOI 10.1145/1864708.1864721. URL http://doi.acm.org/10.1145/1864708.1864721
8. Ekstrand, M.D., Ludwig, M., Konstan, J.A., Riedl, J.T.: Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In: Proceedings of the 5th ACM Conference on Recommender Systems, pp. 133–140. ACM, New York, NY, USA (2011)
9. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact. **4**(2), 81–173 (2011). DOI 10.1561/1100000009. URL http://dx.doi.org/10.1561/1100000009
10. de Gemmis, M., Lops, P., Semeraro, G., Musto, C.: An investigation on the serendipity problem in recommender systems. Inf. Process. Manag. **51**(5), 695 – 717 (2015). DOI http://dx.doi.org/10.1016/j.ipm.2015.06.008. URL http://www.sciencedirect.com/science/article/pii/S0306457315000837
11. Gunawardana, A., Shani, G.: Evaluating recommender systems. In: Recommender systems handbook, pp. 265–308. Springer (2015)
12. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 41–48. ACM, New York, NY, USA (2000)
13. Kaminskas, M., Bridge, D.: Measuring surprise in recommender systems. In: Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design (Workshop Programme of the 8th ACM Conference on Recommender Systems) (2014)

14. Kaminskas, M., Bridge, D.: Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Transactions on Interactive Intelligent Systems (TiiS) **7**(1), 2 (2016)
15. Kotkov, D.: Serendipity in recommender systems. Jyväskylä studies in computing (281) (2018)
16. Kotkov, D., Konstan, J.A., Zhao, Q., Veijalainen, J.: Investigating serendipity in recommender systems based on real user feedback. In: Proceedings of SAC 2018: Symposium on Applied Computing. ACM (2018)
17. Kotkov, D., Veijalainen, J., Wang, S.: Challenges of serendipity in recommender systems. In: Proceedings of the 12th International conference on web information systems and technologies., vol. 2, pp. 251–256. SCITEPRESS (2016)
18. Kotkov, D., Veijalainen, J., Wang, S.: A serendipity-oriented greedy algorithm for recommendations. In: Proceedings of the 13th International Conference on Web Information Systems and Technologies, vol. 1, pp. 32–40. ScitePress (2017)
19. Kotkov, D., Wang, S., Veijalainen, J.: A survey of serendipity in recommender systems. Knowledge-Based Systems **111**, 180–192 (2016)
20. Lathia, N., Hailes, S., Capra, L., Amatriain, X.: Temporal diversity in recommender systems. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pp. 210–217. ACM (2010)
21. Lu, Q., Chen, T., Zhang, W., Yang, D., Yu, Y.: Serendipitous personalized ranking for top-n recommendation. In: Proceedings of the The IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 258–265. IEEE Computer Society, Washington, DC, USA (2012)
22. Maksai, A., Garcin, F., Faltings, B.: Predicting online performance of news recommender systems through richer evaluation metrics. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 179–186. ACM, New York, NY, USA (2015)
23. Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Fujimura, K., Ishida, T.: Classical music for rock fans?: Novel recommendations for expanding user interests. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 949–958. ACM, New York, NY, USA (2010). DOI 10.1145/1871437.1871558. URL http://doi.acm.org/10.1145/1871437.1871558
24. Onuma, K., Tong, H., Faloutsos, C.: Tangent: A novel, 'surprise me', recommendation algorithm. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 657–666. ACM, New York, NY, USA (2009). DOI 10.1145/1557019.1557093. URL http://doi.acm.org/10.1145/1557019.1557093
25. Pandey, G., Kotkov, D., Semenov, A.: Recommending serendipitous items using transfer learning. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1771–1774. ACM (2018)
26. Remer, T.G.: Serendipity and the three princes: From the Peregrinaggio of 1557, p. 20. Norman, U. Oklahoma P (1965)
27. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook, chap. Introduction to Recommender Systems Handbook, pp. 1–35. Springer US (2011)
28. Su, R., Yin, L., Chen, K., Yu, Y.: Set-oriented personalized ranking for diversified top-n recommendation. In: Proceedings of the 7th ACM conference on Recommender systems, pp. 415–418. ACM (2013)
29. Tacchini, E.: Serendipitous mentorship in music recommender systems. Ph.D. thesis (2012)
30. Vall, A., Dorfer, M., Schedl, M., Widmer, G.: A hybrid approach to music playlist continuation based on playlist-song membership pp. 1374–1382 (2018). DOI 10.1145/3167132.3167280. URL http://doi.acm.org/10.1145/3167132.3167280
31. Vig, J., Sen, S., Riedl, J.: The tag genome: Encoding community knowledge to support novel interaction. ACM Transactions on Interactive Intelligent Systems (TiiS) **2**(3), 13 (2012)
32. Zhang, Y.C., Séaghdha, D.O., Quercia, D., Jambor, T.: Auralist: Introducing serendipity into music recommendation. In: Proceedings of the 5th ACM International Conference on Web Search and Data Mining, pp. 13–22. ACM, New York, NY, USA (2012)

33. Zheng, Q., Chan, C.K., Ip, H.H.: An unexpectedness-augmented utility model for making serendipitous recommendation. In: Advances in Data Mining: Applications and Theoretical Aspects, vol. 9165, pp. 216–230. Springer International Publishing (2015)
34. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web, pp. 22–32. ACM, New York, NY, USA (2005)