JYVÄSKYLÄ STUDIES IN COMPUTING
281

Denis Kotkov

# Serendipity in Recommender Systems

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 281

Denis Kotkov

# Serendipity in Recommender Systems

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Alfa-salissa
kesäkuun 7. päivänä 2018 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Alfa hall, on June 7, 2018 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2018

# Serendipity in Recommender Systems

JYVÄSKYLÄ STUDIES IN COMPUTING 281

Denis Kotkov

# Serendipity in Recommender Systems

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2018

Editors
Marja-Leena Rantalainen
Faculty of Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-7438-1

URN:ISBN:978-951-39-7438-1
ISBN 978-951-39-7438-1 (PDF)

ISBN 978-951-39-7437-4 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2018

## ABSTRACT

Kotkov, Denis
Serendipity in Recommender Systems
Jyväskylä: University of Jyväskylä, 2018, 72 p. (+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 281)
ISBN 978-951-39-7437-4 (nid.)
ISBN 978-951-39-7438-1 (PDF)
Finnish summary
Diss.

The number of goods and services (such as accommodation or music streaming) offered by e-commerce websites does not allow users to examine all the available options in a reasonable amount of time. Recommender systems are auxiliary systems designed to help users find interesting goods or services (items) on a website when the number of available items is overwhelming. Traditionally, recommender systems have been optimized for accuracy, which indicates how often a user consumed the items recommended by system. To increase accuracy, recommender systems often suggest items that are popular and suitably similar to items these users have consumed in the past. As a result, users often lose interest in using these systems, as they either know about the recommended items already or can easily find these items themselves. One way to increase user satisfaction and user retention is to suggest *serendipitous* items. These items are items that users would not find themselves or even look for, but would enjoy consuming. Serendipity in recommender systems has not been thoroughly investigated. There is not even a consensus on the concept's definition. In this dissertation, serendipitous items are defined as relevant, novel and unexpected to a user.

In this dissertation, we (a) review different definitions of the concept and evaluate them in a user study, (b) assess the proportion of serendipitous items in a typical recommender system, (c) review ways to measure and improve serendipity, (d) investigate serendipity in cross-domain recommender systems (systems that take advantage of multiple domains, such as movies, songs and books) and (e) discuss challenges and future directions concerning this topic.

We applied a Design Science methodology as the framework for this study and developed four artifacts: (1) a collection of eight variations of serendipity definition, (2) a measure of the serendipity of suggested items, (3) an algorithm that generates serendipitous suggestions, (4) a dataset of user feedback regarding serendipitous movies in the recommender system MovieLens. These artifacts are evaluated using suitable methods and communicated through publications.

Keywords: recommender systems, serendipity, relevance, novelty, unexpectedness, personalization, evaluation, recommendation algorithms, evaluation metrics, offline experiments, user study, serendipity metrics

| | |
|---|---|
| **Author** | Denis Kotkov<br>Faculty of Information Technology<br>University of Jyväskylä<br>Finland |
| **Supervisors** | Professor Dr. Jari Veijalainen<br>Faculty of Information Technology<br>University of Jyväskylä<br>Finland |
| | Professor Dr. Pertti Saariluoma<br>Faculty of Information Technology<br>University of Jyväskylä<br>Finland |
| | Dr. Shuaiqiang Wang<br>Data Science Lab<br>JD.COM<br>China |
| **Reviewers** | Postdoctoral Researcher, Dr. Fedelucio Narducci<br>Department of Computer Science<br>University of Bari "Aldo Moro"<br>Italy |
| | Senior Lecturer, Dr. Derek Bridge<br>Department of Computer Science<br>University College Cork<br>Ireland |
| **Opponent** | Associate Professor, Dr. Konstantinos Stefanidis<br>Faculty of Natural Sciences<br>University of Tampere<br>Finland |

## ACKNOWLEDGEMENTS

Many people have helped me conduct the research presented in this dissertation. I would first like to thank my supervisors, Prof. Jari Veijalainen, Dr. Shuaiqiang Wang and Prof. Pertti Saariluoma for their support during this research. I am thankful to the external reviewers and the opponent for agreeing to review my dissertation and providing their comments.

I would like to thank my coauthors, Prof. Joseph A. Konstan, Qian Zhao, Dr. Alexander Semenov, Dr. Aleksandr Farseev and Prof. Tat-Seng Chua. I learned a lot from working with them.

I also appreciate the help from people who are not coauthors of this dissertation and who might not have been mentioned, but who provided their comments or guidance and contributed to this research. I would like to express my gratitude to Dr. Max Harper and the GroupLens team at the University of Minnesota, who participated in my pilot experiments and guided me during my research visit. I am thankful to the members of the Somea team and the MineSocMed project at the University of Jyväskylä for their comments and guidance, and I would like to thank Gaurav Pandey, Dr. Sergey Chernov, Alexandr Maslov, Dr. Oleksiy Mazhelis and Daria Wadsworth for their comments and suggestions.

I wish to thank my family and my friends for their support and interest in my work.

Finally, I would like to acknowledge the grants that financially supported my research: Academy of Finland grant #268078 (MineSocMed), the grant from the KAUTE Foundation and the grants provided by the University of Jyväskylä (including the mobility grant).

Jyväskylä, May 2018
Denis Kotkov

“Recommendations and personalization live in the sea of data we all create as we move through the world, including what we find, what we discover, and what we love. We’re convinced the future of recommendations will further build on intelligent computer algorithms leveraging collective human intelligence. The future will continue to be computers helping people help other people.”

– *Smith, B. & Linden, G. 2017. Two decades of recommender systems at amazon.com. IEEE Internet Computing 21 (3), 12–18.*

## LIST OF ACRONYMS

| | |
|---|---|
| **IBCF** | Item-Based Collaborative Filtering |
| **kFN** | k-Furthest Neighbor |
| **kNN** | k-Nearest Neighbor |
| **MAE** | Mean Absolute Error |
| **MAP** | Mean Average Precision |
| **MF** | Matrix Factorization |
| **NDCG** | Normalized Discounted Cumulative Gain |
| **RMSE** | Root Mean Squared Error |
| **RWR** | Random Walk With Restarts |
| **RWR-KI** | Random Walk With Restarts Enhanced With Knowledge Infusion |
| **SOG** | Serendipity-Oriented Greedy |
| **SVD** | Singular Value Decomposition |
| **TD** | Topic Diversification |
| **UBCF** | User-Based Collaborative Filtering |

## LIST OF FIGURES

## LIST OF TABLES

# CONTENTS

## LIST OF INCLUDED ARTICLES

| | |
|---|---|
| PI | Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems.*, 2016. |
| PII | Denis Kotkov, Joseph A. Konstan, Qian Zhao, Jari Veijalainen. Investigating Serendipity in Recommender Systems Based on Real User Feedback. *Proceedings of SAC 2018: Symposium on Applied Computing (Accepted)*, 2018. |
| PIII | Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang. Challenges of Serendipity in Recommender Systems. *Proceedings of the 12th International conference on web information systems and technologies.*, 2016. |
| PIV | Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang. A Serendipity-Oriented Greedy Algorithm for Recommendations. *Proceedings of the 13rd International conference on web information systems and technologies (WEBIST) (Best Paper Award)*, 2017. |
| PV | Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang. A Serendipity-Oriented Greedy Algorithm and a Complete Serendipity Metric for Recommendation Purposes. *Springer Computing (Under review).* |
| PVI | Aleksandr Farseev, Denis Kotkov, Alexander Semenov, Jari Veijalainen, Tat-Seng Chua. Cross-Social Network Collaborative Recommendation. *Proceedings of the 2015 ACM conference on Web science*, 2015. |
| PVII | Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen. Cross-domain recommendations with overlapping items. *Proceedings of the 12th International conference on web information systems and technologies*, 2016. |
| PVIII | Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen. Improving Serendipity and Accuracy in Cross-Domain Recommender Systems. *Web Information Systems and Technologies*, 2017. |

# 1 INTRODUCTION

Many e-commerce websites that appeared in the dot-com bubble in the late 1990s offered a wide variety of products to their customers. The number of these products varied depending on the website and could even exceed tens of millions. Although an increase of the number of products meant that each store was likely to offer a product suitable for a particular user, it became difficult for users to find suitable products due to the overwhelming number of products (Ricci et al., 2015).

To overcome this problem, the e-commerce websites started suggesting products from limited catalogs (Ekstrand et al., 2011b). Users could still see the whole catalog of products, but they were also suggested a list of recommendations, which was generated based on the history of purchases of these users and on what these users were looking for at the moment of recommendation generation.

To generate recommendations, e-commerce websites integrated systems that included graphical user interfaces, databases and algorithms. Originally, researchers used the term collaborative filtering to refer to these systems (Goldberg et al., 1992). However, later, researchers widely adopted the term recommender systems (Resnick and Varian, 1997).

Recommender systems received considerable attention due to their popularity among e-commerce websites (Ricci et al., 2015). However, these systems were originally designed to help users navigate in the information space rather than to help users buy goods or increase turnover of online stores (Rich, 1979; Goldberg et al., 1992). Recommender systems suggest a variety of content, such as songs, movies or articles, which can be consumed online and do not even have to be purchased separately, but can be accessed via subscription or for free.

To discuss recommender systems in general, we define recommender systems as software tools that suggest items of use to users (Ricci et al., 2015), where 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" (paper PIII). An item can refer to any object, such as a product on Amazon[1], a video on YouTube[2] or a song on Spotify[3].

Recommender systems suggest items to users and these users decide whether to consume objects, to which these items refer. For brevity, we use the term item to describe the reference and the object itself that a recommender system suggests. For example, when a user consumes an item, they can watch a video in YouTube or buy a product in Amazon.

Recommender systems' suggestions can be personalized or non-personalized (Schafer et al., 1999). In case of non-personalized recommendations, each user receives the same recommendations. For example, in Spotify, a user can receive a list of songs ordered according to their popularity in the service. In the case of personalized recommendations, different users receive different recommendations that are often based on items these users liked in the past. For example, Spotify can suggest weekly play lists based on the listening activity of the user. In this dissertation, we focus on personalized recommender systems.

As a rule, recommender systems generate personalized recommendations based on past user behavior, which is usually represented by user ratings (Ekstrand et al., 2011b). Users give ratings to items in different ways depending on the application scenario. For example, in Amazon, a user rates a product by buying it (Smith and Linden, 2017), while in YouTube, a user might rate a video by watching it or by clicking on like or dislike buttons (Davidson et al., 2010). In this dissertation, the term *user rating* (or simply *rating*) refers to the expression of user's opinion on an item in a recommender system.

User ratings are often noisy and might not correspond to reality. When users rate items retrospectively, they might not remember the experience of consuming these items well. Different users might use the same account in the system, which does not allow the system to differentiate between these users. Ratings can become outdated or users might make mistakes while rating items. In this dissertation, we do not consider these kinds of problems regarding user ratings.

To generate recommendations for a particular user (target user), recommender systems use information from the profile of the user. A user profile might include different information, such as an ID, age and location. However, usually recommender systems employ user ratings or information about items rated by the target user from their profile.

The process of generating recommendations for a target user can involve ratings of this user alone or can involve the ratings of other users. Recommendation algorithms that use ratings of only the target user usually generate recommendations based on attributes of items the target user liked. For example, a YouTube user likes several videos that are tagged with the keyword "cooking". A recommendation based on attributes would be another video tagged with the same keyword. Recommendation algorithms that integrate the ratings of other users vary significantly and can also use attributes of items. However, these sys-

[1] https://www.amazon.com/
[2] https://www.youtube.com/
[3] https://www.spotify.com/

tems usually employ rating patterns that they discover in user ratings. For example, a recommendation based on user ratings might be a video that is new to the target user and is liked by users who also liked favorite videos of the target user. To generate recommendations, recommender systems can also employ contextual information, such as time, place or mood of the user (Adomavicius and Tuzhilin, 2015). In this dissertation, we do not consider these kinds of systems that would also take into account the current context while generating recommendations.

## 1.1 Motivation and research questions

Traditionally, researchers have measured the success of recommender systems based on the ratings users assigned to recommended items. For example, an evaluation measure for an e-commerce website can depend on the proportion of recommended products that users bought, while for a movie recommender system this measure can depend on 5-star ratings that users assigned to recommended movies. Recommendation algorithms have mostly been competing within this success measure, which varies depending on the application scenario. In this dissertation, we refer to this success measure as *accuracy evaluation metric*, or just *accuracy*.

To achieve high accuracy, recommendation algorithms mostly target two categories of items: popular items (Smith and Linden, 2017; Lu et al., 2012) and items that are suitably similar to items in the profile of the target user (Iaquinta et al., 2010). In this dissertation, popularity is related to the absolute number of ratings an item receives in a recommender system. A high number of item ratings in the system often corresponds to the popularity of the item outside the system (for example, a number of sold copies of a book) (Celma Herrada, 2009). Item similarity is based on either user ratings or item attributes. Two items can be considered similar if they have common attributes (for example, two movies belonging to the same genre) or if they were given high ratings by the same users.

Popular items are often given high ratings, as they are of high quality (Celma Herrada, 2009; Kotkov et al., 2017). Items similar to items in the profile of the target user are likely to receive high ratings from this user, as these items correspond to this user's tastes (Tacchini, 2012; Kotkov et al., 2017).

Accuracy is one of the most important evaluation metrics, but it does not always correspond to user satisfaction (McNee et al., 2006). Enjoying an item is often different from enjoying the recommendation of this item. User ratings indicate how much a user enjoyed consuming a particular item, but they lack information whether this user liked the recommendation of this item. Users might consume and even enjoy consuming items that they are aware of or would consume regardless of recommendation, which results in accurate recommendations. However, these recommendations often do not satisfy users' needs when users want to find items that they would enjoy, but would not be aware of or even be able to find themselves (Herlocker et al., 2004; McNee et al., 2006). The following example illustrates a recommendation of a popular item. A user is looking for a movie to watch and uses a movie recommender system, which suggests movie titles. The system suggests the very popular movie *The Shawshank Redemption* to this user. The user gives a high rating to this movie, as they already watched and liked this movie. However, the user does not enjoy the recommendation of this movie, since the system did not help them find a movie to watch. Meanwhile, the recommendation of this movie positively contributed to accuracy of the movie recommender system.

Another example illustrates the recommendation of an item similar to items in the target user profile. An Amazon user receives a recommendation for a kettle, because this user recently looked for kettles in the website. The user purchases the kettle, as their kettle is broken and the accuracy of the recommender system increases. However, the user was already aware of kettles when they received the recommendation. In fact, the user would have bought this item anyway. If the recommender system had not suggested the kettle, the user would have found this item by searching. An even worse example is when a user is offered a kettle, because they have just bought one, as it is unlikely that the user would like to have two different kettles. A better method is to suggest a product that a user was not aware of or did not think about, but would enjoy using; this would lead to this user enjoying the website and buying more products.

Suggesting items that are popular and similar to items in the user profile results in high accuracy and is often appropriate. For example, when a new user joins a recommender system, they want to see that the recommender system follows their tastes and suggests high quality items. However, suggesting these items exclusively repulses users in the long term, since users either already know these items or would easily find these items themselves. In fact, recommending a user only items similar to those they rated disallows the user to broaden their preferences (the so-called overspecialization problem) (Tacchini, 2012; Iaquinta et al., 2010; Narducci et al., 2013) and disallows the recommender system to learn new preferences of this user.

One way to improve the user experience and retain users in the system is to suggest serendipitous items, as these are items the users would not find themselves or would not even look for, but would enjoy consuming. In recommender systems, items that are relevant, novel, and unexpected to a user are considered serendipitous in most cases. Serendipity of a recommender system is the property, which indicates how serendipitous items that this system suggests are (paper PI).

In this dissertation, we investigate serendipity not only in recommender systems in general, but also in a narrow area of cross-domain recommender systems. Cross-domain recommender systems take advantage of multiple domains, where a domain usually consists of items of a particular category, such as songs, books or movies (Fernández-Tobías et al., 2012; Cantador et al., 2015). These systems often improve recommendations by enriching the data from the *target domain* with additional datasets from *source domains*. The former refers to the domain from which suggested items are picked, and the latter refers to the domains that contain auxiliary information.

This dissertation is dedicated to serendipity in recommender systems. We made four contributions and answered eleven research questions. The research questions and motivation for them are as follows (the questions follow the logical order, while motivation is described in a chronological order):

***RQ1.*** *What is serendipity in recommender systems?* (Papers PI and PII)

We needed an operationalized definition of the term serendipity in the context of recommender systems to pick a suitable measure and design an algorithm which could improve this measure. It turned out that there was no consensus on the definition of the term. We thus picked the definition employed in most publications. Later, we revisited this research question by reconducting the literature review, since our experiments (such as conducting a survey and measuring performance of algorithms) required a more elaborate definition, which resulted in eight variations of the serendipity definition.

***RQ2.*** *What are the effects of items corresponding to different variations of novelty, unexpectedness and serendipity on users?* (Paper PII)

Our literature review suggested that serendipity includes three components: relevance, novelty and unexpectedness, where relevance indicates how much a user enjoyed consuming an item, novelty has two variations and indicates familiarity of the user to the item and unexpectedness has four variations and indicates how much the user is surprised by the item. The variations of serendipity components resulted in eight variations of serendipity.

Since our literature review suggested eight variations of serendipity, it became necessary to investigate the difference of variations of novelty, unexpectedness and serendipity in terms of user perception. In particular, we looked at two of the properties of serendipitous items: user preference broadening and user satisfaction.

***RQ3.*** *How rare are serendipitous items among items rated by the users in a typical collaborative filtering-based recommender system? To what extent does this kind of system help users find these items?* (Paper PII)

It was necessary to investigate whether it is feasible to suggest serendipitous items, since in the attempt of suggesting these items, a recommender system is likely to suggest many irrelevant ones (paper PI). Serendipity might not be worth optimizing for if it is almost impossible to recommend serendipitous items due to their rareness.

***RQ4.*** *How can we assess serendipity in recommender systems?* (Paper PI)

To design a recommendation algorithm that improves serendipity (serendipity-oriented), it was necessary to review existing serendipity measures in order to measure what we wanted to improve.

***RQ5.*** *What are effective features for detecting serendipitous items? What are the value ranges of these features for typical serendipitous items?* (Paper PII)

Serendipity metrics are based on assumptions, such as whether serendipitous items are unpopular or dissimilar to items in the user profile. These assumptions might not correspond to reality, since they are not based on experiments involving real users (Kotkov et al., 2018). We conducted a user study to detect important factors for detecting serendipitous items, which helps to design serendipity metrics and serendipity-oriented algorithms.

***RQ6.*** *What are the state-of-the-art recommendation algorithms that suggest serendipitous items?* (Paper PI)

To design a serendipity-oriented recommendation algorithm, we had to review existing ones.

***RQ7.*** *What are the challenges of serendipity in recommender systems?* (Paper PIII)

We reviewed common problems that occur when dealing with serendipity to be prepared for them in our research.

***RQ8.*** *What are the future directions of serendipity in recommender systems?* (Paper PI)

We indicated unsolved problems regarding serendipity in recommender systems to inspire future efforts on this topic.

***RQ9.*** *Can source domains improve accuracy in the target domain when only users overlap?* (Paper PVI)

***RQ10.*** *Can a source domain improve accuracy in the target domain when only items overlap?* (Papers PVII and PVIII)

The topic of cross-domain recommender systems is complicated because of the contradicting results of studies and the lack of publicly available datasets (paper PVII). We started investigating this topic by measuring the accuracy of existing algorithms, since accuracy is more easily investigated than serendipity and accuracy is related to serendipity (serendipitous items are those that users enjoy consuming).

***RQ11.*** *Can a source domain improve serendipity in the target domain when only items overlap?* (Paper PVIII)

Due to the complexity of cross-domain recommender systems, our next step was to measure the performance of common recommendation algorithms on cross-domain datasets in terms of serendipity.

## 1.2 Research process

This dissertation employs a design science research methodology (Peffers et al., 2007), which consists of six steps:

Step 1. Problem Identification and Motivation

Step 2. Objective of a Solution

Step 3. Design and Development

Step 4. Demonstration

Step 5. Evaluation

Step 6. Communication

The research methodology is structured in sequential order, but it allows for starting the process from steps 1, 2, 3, or 4, depending on the initiator of the research. The research methodology also allows iterations in the research process. It is possible to return from steps 5 and 6 to steps 2 and 3.

In this dissertation, we design four artifacts:

1. A definition of serendipity. By conducting a literature review, we discovered that there was no agreement on definition of serendipity. We therefore summarized the most common definitions and used them in our user study (papers PI and PII).

2. An evaluation metric. Since a commonly accepted definition of serendipity is missing, there is no agreement on the evaluation metric to measure serendipity in recommender systems (Silveira et al., 2017). We therefore proposed our metric based on the most common definitions of the concept (papers PV and PIV).

3. A serendipity-oriented algorithm. Due to the lack of a commonly agreed-upon serendipity metric, existing serendipity-oriented algorithms are optimized for different metrics (Silveira et al., 2017). We designed a serendipity-oriented recommendation algorithm optimized for our serendipity metric (Kotkov et al., 2017, 2018).

4. A serendipity-oriented dataset. Datasets containing user feedback regarding serendipity are publicly unavailable. We therefore collected and published the dataset containing user ratings indicating whether particular items are serendipitous to these users (paper PV).

Table 1 demonstrates the research methodology process regarding each artifact (Peffers et al., 2007). We did not use the dataset containing data on serendipity to design our algorithm and evaluation metric, as the dataset was collected for future experiments. The evaluation activity corresponds to design science evaluation methods (Von Alan et al., 2004). We used the following evaluation methods:

Informed argument (descriptive method): We used the literature review to build convincing arguments for our artifacts' utility.

Simulation (experimental method): We executed recommendation algorithms on pre-collected datasets.

Static analysis (analytical method): We examined characteristics of our dataset by using statistical methods.

TABLE 1 Design science research methodology process for this dissertation

| Artifact | Identify problem & motivate | Define objectives of a solution | Design & development | Demonstration | Evaluation | Communication |
|---|---|---|---|---|---|---|
| A definition of serendipity | No agreement on definition of serendipity | Develop the most suitable definition | A collection of eight definitions of serendipity | The definitions are used in a user study | Informed argument (descriptive), static analysis (analytical) | Papers PI, PIII and PII |
| An evaluation metric | Existing metrics do not corresponds to any of the eight definitions of serendipity | Measure serendipity of a recommender system according to the target definition of serendipity (among the eight definitions) | An evaluation metric | Qualitative analysis | Informed argument (descriptive), simulation (experimental) | Papers PIV and PV |
| A serendipity-oriented algorithm | Improvement in suggesting serendipitous items | Suggest items serendipitous according to the target definition of serendipity (among the eight definitions) | A serendipity -oriented recommendation algorithm | Quantitative and qualitative analysis | Simulation (experimental) | Papers PIV and PV |
| A serendipity-oriented dataset | The absence of datasets that include user feedback on serendipity | Collect a dataset that allows to investigate serendipity and evaluate serendipity -oriented algorithms | A dataset with user feedback on serendipity | Quantitative analysis | Static analysis (analytical) | Papers PII |

The minor contribution of this dissertation is software, which was developed to evaluate recommendation algorithms, conduct statistical analysis, prepare data used in the research, build charts and conduct surveys. The source code is publically available only for experiments conducted to evaluate our serendipity-oriented recommendation algorithm (papers PIV and PV). The source code can be found at GitHub[4]. We used the java open-source framework for recommender systems, LensKit (Ekstrand et al., 2011a).

## 1.3 Research methods

In this dissertation, we conducted a literature review, a user study and several offline experiments for our research.

### 1.3.1 Literature review

We used the literature review to address ***RQ1***, ***RQ4***, ***RQ6***, ***RQ7*** and ***RQ8*** (Papers PI and PIII). To collect articles that mention serendipity, we used the following sources: Google Scholar[5], Scopus[6] and Web of Science[7]. Our literature review

[4] https://github.com/Bionic1251/SerendipitousAlgorithm
[5] https://scholar.google.com/
[6] http://www.scopus.com/
[7] https://webofknowledge.com/

process consisted of the following steps:

1. We first conducted preliminary analysis by reading papers returned in response to the query "serendipity in recommender systems." We discovered that there is no consensus on the definition of serendipity or methods to measure it (Murakami et al., 2008; Kaminskas and Bridge, 2014; Zhang et al., 2012).

2. To investigate the existing definitions of serendipity in recommender systems and ways to assess this property, we selected the top 20 articles retrieved by the search engines in response to the search queries "definition of serendipity in recommender systems," "serendipity in recommender systems," "measure surprise in recommender systems" and "unexpectedness in recommender systems." To find more relevant articles, we employed a forward search (Levy and Ellis, 2006) by picking articles from references of the selected articles. Eventually, we found another 18 qualifying articles, six of which presented serendipity evaluation metrics.

3. To find serendipity-oriented algorithms (***RQ6***), we employed a backward search (Levy and Ellis, 2006) by picking articles that site articles which propose methods to measure serendipity.

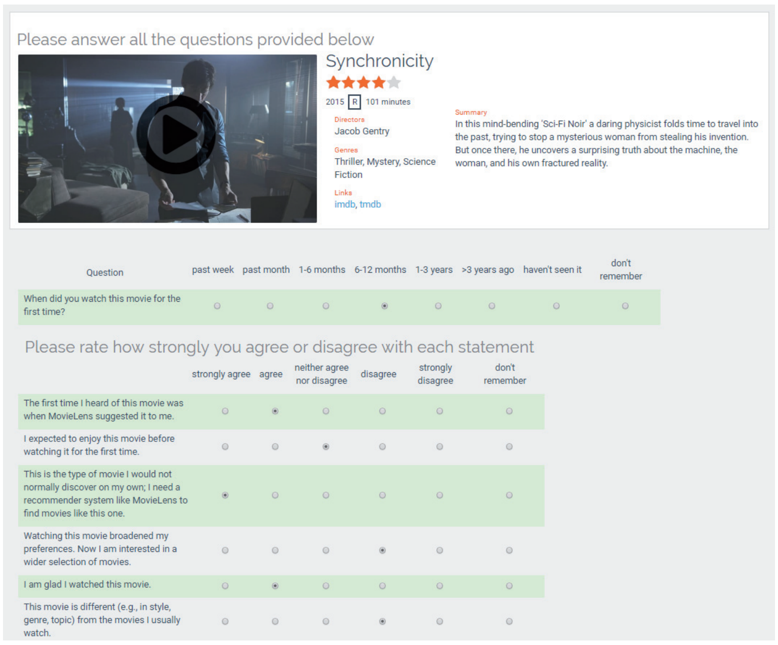
4. We only selected articles that focus on serendipity in recommender systems and filtered out the rest.

### 1.3.2 The user study

To address ***RQ1***, ***RQ2*** and ***RQ4***, we conducted a user study in the movie recommender system MovieLens[8] (paper PII). This system has been designed to provide users with movie recommendations based on movies these users previously watched. MovieLens users can rate movies retrospectively on a scale from 0.5 to 5 stars with the granularity of 0.5 star. Users might rate a movie in some time after they watched it, and the system does not allow users to indicate how long ago they watched the movie. MovieLens also allows users to perform other actions, such as adding a movie to the list of movies to watch, assigning keywords (tags) to movies, and adding new movies to the system.

We conducted a survey, in which we asked users questions regarding the serendipity of movies these users recently watched. On April 1, 2017, we started inviting users via emails to complete our survey. We selected users based on their registration date and recent activity. First, we selected users who joined the system at least one month before the experiment to make sure that users had had enough time to rate movies they watched a long time ago. Second, among the selected users, we picked those who assigned ratings of at least 3.5 stars to five or more movies during the three months before the experiment (from December 30, 2016 till March 30, 2017) and one month after the registration (for those who joined the system after November 30, 2016).

[8] https://movielens.org/



Please answer all the questions provided below

Synchronicity

★★★★☆

2015 R 101 minutes

Directors
Jacob Gentry

Genres
Thriller, Mystery, Science Fiction

Links
imdb, tmdb

Summary
In this mind-bending 'Sci-Fi Noir' a daring physicist folds time to travel into the past, trying to stop a mysterious woman from stealing his invention. But once there, he uncovers a surprising truth about the machine, the woman, and his own fractured reality.

| Question | past week | past month | 1-6 months | 6-12 months | 1-3 years | >3 years ago | haven't seen it | don't remember |
|---|---|---|---|---|---|---|---|---|
| When did you watch this movie for the first time? | ○ | ○ | ○ | ◉ | ○ | ○ | ○ | ○ |

Please rate how strongly you agree or disagree with each statement

| | strongly agree | agree | neither agree nor disagree | disagree | strongly disagree | don't remember |
|---|---|---|---|---|---|---|
| The first time I heard of this movie was when MovieLens suggested it to me. | ○ | ◉ | ○ | ○ | ○ | ○ |
| I expected to enjoy this movie before watching it for the first time. | ○ | ○ | ◉ | ○ | ○ | ○ |
| This is the type of movie I would not normally discover on my own; I need a recommender system like MovieLens to find movies like this one. | ◉ | ○ | ○ | ○ | ○ | ○ |
| Watching this movie broadened my preferences. Now I am interested in a wider selection of movies. | ○ | ○ | ○ | ◉ | ○ | ○ |
| I am glad I watched this movie. | ○ | ◉ | ○ | ○ | ○ | ○ |
| This movie is different (e.g., in style, genre, topic) from the movies I usually watch. | ○ | ○ | ○ | ◉ | ○ | ○ |

FIGURE 1 A MovieLens survey page

Figure 1 demonstrates the survey page in MovieLens, where we asked users to rate the statements. For each user, we picked five movies they rated during the three months before the experiment and one month after their registration. In cases where more than five movies satisfied our selection criterion, we picked the least popular ones (with the smallest number of ratings in the system), expecting that users encountered these movies in MovieLens, since users were likely to discover popular movies from other sources, such as social media, TV or friends.

We selected 2,305 users and sent an invitation to take our survey via email; 475 of the invited users gave their feedback on at least one movie. Overall, these users rated all the statements and answered all the questions about 2166 ratings (user-movie pairs). We excluded 20 of these ratings, since users indicated they had not watched these movies (the question regarding how long ago the user watched the movie included this option). Overall, we analyzed user feedback on 2,146 ratings given by 475 users.

### 1.3.3 Offline evaluations

We conducted offline evaluation to design our algorithm (Papers PIV and PV) and answer ***RQ7***, ***RQ8*** and ***RQ9*** (Papers PVII, PVIII and PVI). In offline evaluation, researchers conduct experiments using pre-collected or simulated datasets. In the experiments, researchers hide some ratings and let an algorithm predict them based on the rest of the data. The performance of algorithms is usually presented by evaluation metrics (Herlocker et al., 1999; Shani and Gunawardana, 2011; Ekstrand et al., 2011b).

We used offline evaluation to answer our research questions regarding cross-domain recommender systems, where we evaluated common recommendation algorithms on cross-domain datasets. We also used offline evaluation to iteratively improve and demonstrate our serendipity-oriented recommendation algorithm.

## 1.4 The dissertation structure

The structure of this dissertation is as follows. In the next chapter, we provide literature review regarding serendipity, recommender systems in general and cross-domain recommender systems in particular. Chapter 3 describes main results and contributions of our research. Chapter 4 contains short summaries of articles included in this dissertation. Finally, in chapter 5, we provide conclusions along with limitations, discussion and directions for further research in serendipity in recommender systems.

This dissertation is a compilation of eight articles; six articles have been published, one article is currently under review and one article has been accepted for publication. All the publication outlets (e.g. conference proceedings, journals and books) are peer reviewed. We received permission to include all eight articles to this dissertation by publishers and coauthors.

# 2 RELATED WORK

In this chapter, we provide literature review regarding recommender systems. We present the history of recommender systems, strategies to evaluate recommendation algorithms and metrics to assess their performance. We also discuss the history and definitions of serendipity in general and in relation to recommender systems. Finally, we provide an overview of cross-domain recommender systems.

## 2.1 Recommender systems

The term recommender system was proposed by Resnick and Varian in 1997 (Leino, 2014; Resnick and Varian, 1997). The authors noticed that the term *collaborative filtering* does not correspond to the variety of systems that suggest items. According to (Resnick and Varian, 1997), "recommender systems assist and augment this natural social process" of making "choices without sufficient personal experience of the alternatives."

Although similar in the ultimate goal of satisfying users' information needs, recommender systems are significantly different from search engines. Traditionally, a search engine receives a query as an input and provides a set of the most suitable items in response (Brin and Page, 2012). Personalized search engines utilize both queries and information about users. Two different users may receive different results when entering the same query (Smyth et al., 2011). In contrast, a recommender system does not receive any query. It receives information about the past behavior of users and returns a set of items users would enjoy (Ricci et al., 2015).

### 2.1.1 History of recommender systems

To the best of our knowledge, the first recommender system was called Grundy and designed in 1979 (Rich, 1979). Its goal was to suggest books to readers based on their personality traits. The system asked a target user questions regarding their personality in a scripted dialog and suggested a book to read based on answers of this user.

The first collaborative filtering recommender system was Tapestry, the e-mail filtering system (Goldberg et al., 1992). Tapestry enabled users to annotate messages that they found interesting or uninteresting. Others, in turn, could use the annotations to filter or browse the content. The authors used the term *collaborative filtering* to denote the process of individuals helping one another to filter content through interaction.

Recommender systems attracted considerable attention in the late 90s (Ekstrand et al., 2011b). It was discovered that these systems can improve sales of products by introducing users to potentially interesting items. In 2006, Netflix[1] presented Netflix Prize, a challenge to develop an algorithm that would outperform Netflix's internal algorithm Cinematch by 10% (Netflix, 2009). The competition attracted considerable attention due to the $1 million prize. The Netflix Prize highlighted the importance of recommender systems and motivated many researchers to contribute to the field.

Recommender systems are now a core function of most popular services, such as Amazon, Netflix or Quora[2] (Amatriain, 2016). Recommender systems have an effect on the turnover of these services. For example, more than 80% of movies that users watch in Netflix are chosen after being suggested to users by the recommender system (Amatriain, 2016). These popular services provide users with the recommended content first and offer browsing as an alternative (Amatriain, 2016).

### 2.1.2 Evaluation strategies

Recommendation algorithms are evaluated in two kinds of experiments: online and offline (Silveira et al., 2017). An online evaluation of a recommender system involves users interacting with the system. Researchers can assess the accuracy of this system by asking users whether they found a particular item relevant.

In offline experiments, researchers run algorithms on a pre-collected dataset. They split the dataset into two parts: training and test. Algorithms receive training data as an input and predict test data. The performance of algorithms is measured with evaluation metrics. Metrics for accuracy are the most common ones.

Online evaluations are regarded as more representative than offline evaluations. In an online evaluation, researchers receive data from real users, while in an offline evaluation, users' behavior is simulated. Meanwhile, conducting an online evaluation is more demanding and requires more resources than an offline evaluation, since the online evaluation is conducted either in an online recommender system or in a survey consisting of a few phases in which we ask users to rate items, generate recommendations, suggest them to the users and ask the

[1] https://www.netflix.com
[2] https://www.quora.com

TABLE 2 Notations

| Symbol | Description |
|---|---|
| $I = \{i_1, i_2, ..., i_{\|I\|}\}$ | the set of items |
| $I_u, I_u \subseteq I$ | the set of items in profile of user $u$ (rated by user $u$) |
| $REL_u, REL_u \subseteq I_u$ | the set of items relevant for user $u$ (rated highly by user $u$) |
| $F = \{f_1, f_2, ..., f_{\|F\|}\}$ | the set of features |
| $F_i, F_i \subseteq F$ | the set of features that describe item $i$ |
| $U = \{u_1, u_2, ..., u_{\|U\|}\}$ | 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 the recommender system to user $u$ |
| $r_{ui}$ | the rating given by user $u$ to item $i$ |
| $\hat{r}_{ui}$ | the prediction of the rating that user $u$ will give to item $i$ |

users to rate the recommended items.

### 2.1.3 Accuracy metrics

This section presents a review of the most common accuracy metrics that indicate the predictive power of algorithms in an offline experiment. Accuracy metrics capture how good recommender systems are at predicting relevant items (items that the user expressed preference for in the system). The following section presents prediction accuracy metrics, indicating how good an algorithm is at predicting user ratings as well as rank-based accuracy metrics, which indicate how precise an algorithm is at ordering recommended items according to their relevance for users.

To review evaluation metrics, we first present the notation in table 2. Let $I$ be a set of items and $U$ be a set of users in a recommender system at a particular moment of time $t$. Top-n recommendations suggested by the recommender system to user $u$ at time $t$ are denoted by $RS_u(n)$. The rating that the system predicts user $u$ will give to item $i$ at time $t$ is denoted by $\hat{r}_{u,i}$, while the rating that user $u$ will assign to item $i$ after time $t$ is denoted by $r_{u,i}$. All the items user $u$ rated before time $t$ are denoted by $I_u, I_u \subseteq I$. Item $i$ is described with features $F_i = \{f_{i,1}, f_{i,2}, ..., f_{i,\|F\|}\}$. For example, the movie *The Shawshank Redemption* can be described with features $F_{redemption} = \{prison, escape, narated\}$, while the song "Jingle Bells" by James Lord Pierpont can be described with features $F_{jingle} = \{christmas, celebration, holiday\}$.

#### 2.1.3.1 Prediction Accuracy

Many recommendation algorithms provide a prediction of a rating a particular user will give to an item in the future. One of the ways to evaluate these algorithms is to compare the ratings they predict to those given by the users. In offline experiments, an algorithm receives the training part of the dataset as input data and provides its predictions for the test part of the dataset. Let $\mathcal{R}_{test}$ be a set of ratings in the test part. The performance of algorithms in this case is usually measured by Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) (Celma Herrada, 2009; Ning et al., 2015):

$$MAE = \frac{1}{\|\mathcal{R}_{test}\|} \sum_{r_{ui} \in \mathcal{R}_{test}} |r_{ui} - \hat{r}_{ui}|, \tag{1}$$

$$RMSE = \sqrt{\frac{1}{\|\mathcal{R}_{test}\|} \sum_{r_{ui} \in \mathcal{R}_{test}} (r_{ui} - \hat{r}_{ui})^2}. \tag{2}$$

#### 2.1.3.2 Rank Accuracy

To assess the order of recommendations provided by the algorithm, researchers employ rank-based accuracy metrics. The basic assumption of these metrics is that the closer items are to the top of a recommended item list, the more relevant they are for a user.

Rank-based metrics are widely adopted in information retrieval to compare ranking models (Liu, 2009). Given a query, a ranking model returns an ordered list of items. Each list may contain items which are both relevant and irrelevant to the query. Rank-based metrics were designed to assess the order of items in retrieved lists.

**Precision@k, Recall@k and F1 measure.** The assumption behind the metrics is that a user only becomes familiar with the first $k$ recommendations. *Precision@k* indicates a share of relevant items with respect to the length of a recommended list (McSherry and Najork, 2008):

$$P@k = \frac{1}{\|U\|} \sum_{u \in U} \frac{\|RS_u(k) \cap REL_u\|}{k}. \tag{3}$$

*Recall@k* reflects a share of relevant items in the first $k$ suggestions:

$$R@k = \frac{1}{\|U\|} \sum_{u \in U} \frac{\|RS_u(k) \cap REL_u\|}{\|REL_u\|}. \tag{4}$$

F1 measure represents a combination of precision and recall:

$$F1@k = \frac{2}{\frac{1}{P@k} + \frac{1}{R@k}}. \tag{5}$$

**Mean Average Precision (MAP).** MAP reflects the order of items in the recommended list. The closer the relevant items to the top of the list, the higher the value of MAP.

$$MAP = \frac{1}{\|U\|} \sum_{u \in U} \frac{1}{\|R_u\|} \sum_{\|R_u\|}^{k=1} \frac{\|RS_u(k) \cap REL_u\|}{k} \cdot rel(i_{uk}), \tag{6}$$

where $i_{uk}$ corresponds to the item at position $k$ in the recommendation list generated to user $u$, while $rel(i_{uk})$ equals 1 if item $i_{uk}$ is relevant for user $u$, and 0 otherwise.

**Normalized Discounted Cumulative Gain (NDCG).** Precision, Recall, F1 measure and MAP are used to test binary relevance whether an item is either relevant to a user or not. In contrast, NDCG is applied for multiple relevance levels (Järvelin and Kekäläinen, 2002).

$$NDCG@K = \frac{1}{\|U\|} \sum_{u \in U} NDCG_u@K, \tag{7}$$

$$NDCG_u@K = \frac{DCG_u@K}{IDCG_u@K}, \tag{8}$$

where $DCG$ is discounted cumulative gain defined as follows:

$$DCG_u@K = r_{ui_1} + \sum_{k=2}^{\|R_u\|} \frac{r_{ui_k}}{\log_2(i_k)}, \tag{9}$$

where $r_{ui_k}$ is the rating user $u$ gave to item $i_k$, which is located at position $k$ in the recommendation list generated for user $u$. $NDCG_u$ is a normalized $DCG_u$. $IDCG_u$ (ideal $DCG_u$) is the $DCG_u$ value calculated for the list with perfect order, where relevant items are at the beginning of the list.

### 2.1.4 Diversity metrics

The term *diversity* has been used in different ways in recommender systems, such as (a) an average diversity of recommendation lists suggested to all users at a particular point in time (Castells et al., 2015; Kaminskas and Bridge, 2016); (b) diversity of recommendations suggested to a user over time (Lathia et al., 2010); or (c) diversity of all recommendations provided for users at a particular point in time (Adomavicius and Kwon, 2012). In this dissertation, we refer to diversity as a property of a recommender system which indicates an average pairwise dissimilarity of items in recommendation lists suggested by the system to users at a particular point in time (Castells et al., 2015; Kaminskas and Bridge, 2016).

Diversity is often viewed as a valuable property of a recommender system, since it has been shown to improve user satisfaction (Ziegler et al., 2005), and since a diversified recommendation list is likely to contain an item satisfying the user's current needs (Kaminskas and Bridge, 2016). For example, a user who bought the book "The Lord of the Rings" is likely to prefer suggestions of books in a suitable sense similar to this book, such as "The Hobbit" or "The Silmarillion" over a recommendation list consisting of only different editions of this book.

Diversity is an average pairwise dissimilarity of items in recommendation lists offered to users by the system, wherein dissimilarity indicates how dissimilar two items are (Smyth and McClave, 2001):

$$Div_u@n = \frac{1}{n \cdot (n-1)} \sum_{i \in RS_u(n)} \sum_{j \neq i \in RS_u(n)} dissim(i,j), \tag{10}$$

where $dissim(i,j)$ is any dissimilarity measure suitable in the current context, while $n$ is the length of the recommendation list (we assume here that all users are offered recommendation lists of the same length).

### 2.1.5 Accuracy-oriented algorithms

In this section, we describe accuracy-oriented algorithms, on which some serendipity-oriented algorithms are based.

#### 2.1.5.1 User-based collaborative filtering

The User-Based Collaborative Filtering (UBCF) algorithm is a rating prediction algorithm. The assumption behind UBCF is that if users used to agree on a choice in the past, then they would agree on that choice again in the future (Su and Khoshgoftaar, 2009; Ekstrand et al., 2011b). For each target user, the algorithm selects users (a neighborhood) who rated common items similarly based on a similarity measure. UBCF then picks items rated by users from the neighborhood, but not rated by the target user, and estimates ratings the target user might give to these items. Finally, the algorithm can pick top-n items with the highest predicted rating and suggest them to the user (Ekstrand et al., 2011b).

To calculate similarities, an algorithm can employ different similarity measures, such as Manhattan distance, Euclidean distance or cosine distance (Amatriain et al., 2011). One of the more common similarity measures is Pearson correlation (Said et al., 2013).

$$sim(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{ui} - \overline{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{vi} - \overline{r}_v)^2}}, \tag{11}$$

where $sim(u,v)$ is a similarity measure calculated for users $u$ and $v$, while $\overline{r}_v$ is an average rating among all ratings given by user $v$ in the system. The prediction for a rating is calculated as follows:

$$\hat{r}_{ui} = \overline{r}_u + \frac{\sum_{v \in U_u} sim(u,v)(r_{vi} - \overline{r}_v)}{\sum_{v \in U_u} |sim(u,v)|}, \tag{12}$$

where $U_u$ is the neighborhood of user $u$, $U_u \subseteq U$. In the next step, the recommender system can order items according to their predicted ratings and suggest a list to the user.

#### 2.1.5.2 Item-based collaborative filtering

For most online recommender systems, calculating UBCF is time consuming, since user base of these systems grows faster than item base (Ekstrand et al., 2011b). One of the solutions to this problem is Item-Based Collaborative Filtering (IBCF), which is similar to UBCF. IBCF calculates similarities between items instead of users, generates neighborhoods of most similar items for each item and

TABLE 3 An example of user-item rating matrix

| User | Movie 1 | Movie 2 | Movie 3 | Movie 4 |
|---|---|---|---|---|
| Mark | 1 | - | 5 | - |
| Alice | 2 | 1 | - | 1 |
| David | - | 5 | 5 | - |
| Bob | - | - | 3 | 5 |
| John | 5 | 1 | 2 | - |

predicts the rating a particular user will give to an item based on these similarities. Item similarities are more stable than user similarities. IBCF thus allows the recommender system to pre-compute item similarities and generate recommendations faster than UBCF for most online recommender systems.

Similarly to UBCF, IBCF may employ different similarities measures. Here we provide an example of cosine similarity. Let each item be represented as a vector $\mathbf{i} = (r_{u_1 i}, r_{u_2 i}, ..., r_{u_{\|U\|} i})$, where $r_{u_k i}$ is a rating user $u_k$ gave to item $i$.

$$sim(i,j) = \frac{\mathbf{i} \cdot \mathbf{j}}{\|\mathbf{i}\| \|\mathbf{j}\|}, \tag{13}$$

The rating is computed as follows:

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in I_i} sim(i,j)(r_{uj} - \bar{r}_j)}{\sum_{j \in I_i} |sim(i,j)|}, \tag{14}$$

where $I_i$ corresponds to the neighborhood of item $i$, $I_i \subseteq I$. Both IBCF and UBCF are often referred to as k-Nearest Neighbor (kNN) algorithms, since they calculate similarities between users or items and look for the closest neighbors.

#### 2.1.5.3 Matrix Factorization

Matrix Factorization (MF) or Singular Value Decomposition (SVD) algorithms are based on latent factors that motivate users to consume items (Cremonesi et al., 2010; Koren and Bell, 2011). For example, in a movie domain latent factors can be represented by genre, plot or cast of a movie. However, MF algorithms do not normally return interpretable latent factors. MF algorithms detect latent factors to predict user behavior.

MF considers a user-item rating matrix $\mathbf{R}$ of the size $\|U\| \times \|I\|$ where each cell corresponds to the rating a user gave to a specific item (Table 3). The task is to detect $k, k < \|U\| \wedge k < \|I\|$ latent factors. We thus assume that three matrices $\mathbf{\Sigma}_{(k \times k)}$, $\mathbf{U}_{(\|U\| \times k)}$ and $\mathbf{Q}_{(\|I\| \times k)}$ approximate $\mathbf{R}$, where $\mathbf{\Sigma}$ is a diagonal matrix, while $\mathbf{U}$ and $\mathbf{Q}$ are orthogonal matrices:

$$\hat{\mathbf{R}} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{Q}^T, \tag{15}$$

$$\mathbf{P} = \mathbf{U} \times \mathbf{\Sigma}, \tag{16}$$

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}, \tag{17}$$

where the row $p_u$ in $\mathbf{P}$ corresponds to latent factors of user $u$ and $q_i$ in $\mathbf{Q}$ corresponds to latent factors of item $i$. A predicted rating, then, is calculated as follows:

$$\hat{r}_{ui} = p_u \cdot q_i^T. \tag{18}$$

To calculate coefficients in matrices $\mathbf{P}$ and $\mathbf{Q}$, an algorithm employs gradient descent with the following optimization task, which may vary depending on the modification of the algorithm (Zheng et al., 2015):

$$\min \sum_{u \in U} \sum_{i \in I_u} (r_{ui} - p_u \cdot q_i^T)^2 + \lambda(|p_u|^2 + |q_i|^2) \tag{19}$$

where $\lambda$ is a regularization parameter to prevent the situation of the model describing training data very well, but failing to predict testing data (overfitting). As the optimization task finishes, the algorithm may recommend items based on matrix of predicted ratings $\hat{\mathbf{R}}$.

## 2.2 Serendipity: history and definition

The term *serendipity* was coined by Horace Walpole in his letter to Sir Horace Mann in 1754 by describing his unexpected discovery and referencing the fairy tale, "The Three Princes of Serendip." Originally, the story was written by Amir Khusrow in 1302. Cristoforo Armeno translated "The Three Princes of Serendip" from Persian to Italian and published it in 1557 (Cohn and Russell, 2015). The story describes the journey of three princes of the country Serendippo, who were sent by their father to explore the world (Boyle, 2000). Horace Walpole in his letter explained that the princes were "always making discoveries, by accidents and sagacity, of things which they were not in quest of" (Remer, 1965).

The term "serendipity" has been recognized as one of the most untranslatable words (Martin, 2008). According to the dictionary the term has the following definitions:

- "The faculty of making fortunate discoveries by accident"[3]
- "The faculty or phenomenon of finding valuable or agreeable things not sought for; also: an instance of this"[4]
- "An aptitude for making desirable discoveries by accident"[5]
- "The fact of finding interesting or valuable things by chance"[6]

[3] https://www.thefreedictionary.com/serendipity
[4] https://www.merriam-webster.com/dictionary/serendipity
[5] http://www.dictionary.com/browse/serendipity
[6] https://dictionary.cambridge.org/dictionary/learner-english/serendipity

- “The occurrence and development of events by chance in a happy or beneficial way”[7]

There is no consensus on definition of serendipity in recommender systems (paper PI). Researchers employ different definitions of the concept. Nguyen et al. (2017) indicated that serendipitous items are dissimilar to items in the user profile: “Serendipity indicates how different the recommendations are from what users usually consume.” Meanwhile, Karpus et al. (2017) suggested that serendipity includes unexpectedness and relevance: “Serendipity measures the number of unexpected and interesting items recommended.”

Most authors agree that serendipity includes three components: relevance, novelty and unexpectedness. For example, Silveira et al. (2017) stated: “The definition of serendipity consists of three components: utility, novelty and unexpectedness.” Here the term *utility* has the same meaning as relevance in the context of this dissertation. However, definitions of unexpectedness and novelty vary depending on a publication. For example, unexpectedness might mean that the user does not expect to find the item on their own (Herlocker et al., 2004; Ge et al., 2010; Hijikata et al., 2009) or that the user does not expect to enjoy the item (Adamopoulos and Tuzhilin, 2014).

Our definition of serendipity is based on literature review. According to our definition, serendipitous items are relevant, novel and unexpected to users, where relevance has one variation, novelty has two variations and unexpectedness has four variations. This results in eight variations of serendipity.

## 2.3 Cross-domain recommender systems

Recommender systems often suffer from data sparsity and the cold start problem (Cantador et al., 2015). Data sparsity is a very common problem which happens when a recommender system lacks user ratings to produce quality recommendations. Indeed, datasets of recommender systems are extremely sparse. For example, one of the most common datasets published by GroupLens, 10M MovieLens (Harper and Konstan, 2015), has sparsity of $1 - \frac{\#ratings}{\#items \cdot \#users} = 1 - \frac{10000054}{10681 \cdot 71567} = 0.986$. The cold start problem happens in three kinds of situations (Cantador et al., 2015): (a) the system has not yet received enough ratings from a new user to generate recommendations for this user, (b) the system has not yet received enough ratings from users regarding a new item to recommend this item to users, or (c) the system has not yet received enough ratings to generate quality recommendations to any users.

One way to alleviate these problems is to employ data from other domains, where the term domain refers to “a set of items that share certain characteristics that are exploited by a particular recommender system” (Fernández-Tobías et al., 2012). These characteristics include attributes of items and user ratings.

[7] https://en.oxforddictionaries.com/definition/serendipity

Domains can be represented by pictures, songs, movies, venues, Vimeo[8] videos or YouTube videos. Recommender systems that take advantage of multiple domains are called cross-domain recommender systems.

Cross-domain recommender systems usually use data from one or several domains (source domains) to improve recommendations in another domain (the target domain) (Cantador et al., 2015). Cross-domain recommender systems suggest items in the target domain to the users of this domain and use data from both the target domain and the source domain.

For example, let us assume there are target and source domains. Then, let $I_T$ and $U_T$ be a set of items and users of the target domain and $I_S$ and $U_S$ be a set of items and users of the source domain, respectively.

The target and source domains can overlap in four different ways (Cantador et al., 2015):

1. Item overlap. Items of the target domain overlap with items in the source domain: $I_T \cap I_S \neq \emptyset$.

2. User overlap. Users of the target domain overlap with users in the source domain: $U_T \cap U_S \neq \emptyset$. In this case, the domains share user ratings.

3. User and item overlap. Users and items of the target domain overlap with users and items in the source domain: $I_T \cap I_S \neq \emptyset \wedge U_T \cap U_S \neq \emptyset$.

4. No overlap. Neither users nor items of the target domain overlap with users or items in the source domain: $I_T \cap I_S = \emptyset \wedge U_T \cap U_S = \emptyset$. In this case, the domains can share user behavioral patterns or data correlations.

The target and source domains can overlap on different levels depending on the application scenario (Cantador et al., 2015):

- Attribute level. Items of different domains are of the same type, but have different attributes. For example, one domain is represented by documentaries, while another one is represented by comedies.

- Type level. Items of different domains are of similar type and have common attributes. For example, one domain is represented by songs, while another is represented by podcasts. Songs and podcasts share attributes, such as title and length, while songs also have genres and podcasts have topics.

- Item level. Items of different domains are of different types, but have common attributes. For example, songs and venues have different types, but share attributes, such as title or language (language spoken at a particular venue).

- System level. Items belong to different recommender systems. For example, products sold on Amazon and products sold on Ebay[9] belong to different systems.

[8] https://vimeo.com/
[9] https://www.ebay.com/

In this dissertation, we conduct two experiments involving cross-domain recommender systems (paper PVI). In the first experiment, we collect data from three online social networks (Foursquare, Twitter and Instagram) related to the same users, and improve recommendation accuracy by reducing data sparsity in Foursquare by using data from Twitter and Instagram. In this case, our target domain is represented by Foursquare, while our source domains are represented by Twitter and Instagram. These domains differ on an item level and have an intersection only in the sets of users.

In our second experiment, we collect data regarding music preferences from vk.com (target domain) and last.fm (source domain) and improve accuracy and serendipity in vk.com by enriching our dataset with data from last.fm (paper PVIII). In this case, only items overlap on a system level.

# 3 MAIN RESULTS AND CONTRIBUTIONS

In this chapter, we provide answers to our research questions and describe our contributions.

## 3.1 Serendipity in recommender systems

According to our literature review, there is no consensus on the definition of serendipity in recommender systems. Furthermore, there is no evidence that serendipitous items should be recommended to users, and it is unclear whether it is feasible to recommend them. We therefore picked eight variations of the most common definitions of serendipity, compared them in our user study using two user metrics, and assessed the feasibility and benefits of recommending these items.

### 3.1.1 Definitions

According to our literature review and user study, serendipity is a property of a recommender system which reflects how good a recommender system is at suggesting items that are serendipitous. Serendipitous items are relevant, novel and unexpected to the users. Serendipity thus consists of the three components: relevance, novelty and unexpectedness, where relevance has one variation, novelty has two variations and unexpectedness has four variations, which results in eight variations of definitions of serendipity.

The exact definition of relevance usually depends on a particular application scenario. For example, in the movie domain, we might regard a movie as relevant if the user watched the whole movie. In another application scenario, to consider a movie relevant to the user, we might need a user to assign a high rating to the same movie. We employ the wider definition of relevance: "an item is relevant to a user if the user expresses or will express their preference for the item in the future by liking or consuming the item depending on the application scenario" (paper PIV).

Novelty reflects familiarity of a user with an item:

1. *Strict novelty.* The user has never heard about the item (Kapoor et al., 2015).
2. *Motivational novelty.* The user has heard about the item, but has not consumed it (Kotkov et al., 2018; Silveira et al., 2017).

The definition of unexpectedness depends on the kind of user expectations. The user might not expect an item in different ways:

1. *Unexpectedness (relevant).* The user does not expect the item to be relevant to them (Adamopoulos and Tuzhilin, 2014).
2. *Unexpectedness (find).* The user would not have found the item on their own (Adamopoulos and Tuzhilin, 2014; Herlocker et al., 2004; Ge et al., 2010; Hijikata et al., 2009; Taramigkou et al., 2013).
3. *Unexpectedness (implicit).* The item is significantly dissimilar to items the user usually consumes (Kaminskas and Bridge, 2014; Zhang et al., 2012; Kotkov et al., 2016; Nguyen et al., 2017).
4. *Unexpectedness (recommend).* The user does not expect the item to be recommended to them (Kotkov et al., 2018).

Table 4 demonstrates components included in each definition. For example, motivationally serendipitous (find) items are items that (a) users enjoyed consuming (relevant), (b) users consumed, because these items were recommended to the users (motivational novelty) and (c) users would not have found these items on their own (unexpectedness (find)). Multiple variations of serendipity components result in eight definitions of serendipity: motivational serendipity (relevant), motivational serendipity (find), motivational serendipity (implicit), motivational serendipity (recommend), strict serendipity (relevant), strict serendipity (find), strict serendipity (implicit) and strict serendipity (recommend).

To evaluate our variations of serendipity, we conducted a user study (paper PII) and compared serendipity variations' results between each other in terms of user metrics, in which serendipitous items are believed to achieve high results: preference broadening and user satisfaction. We excluded the user metric of overcoming the overspecialization problem through the design of our user study. The overspecialization problem happens in content-based recommender systems, while we conducted an experiment in a recommender system which mostly uses collaborative filtering algorithms.

***RQ1.*** *What is serendipity in recommender systems?*

The serendipity of a recommender system is that property, which indicates how good the recommender system is at suggesting items that are serendipitous. Serendipitous items are relevant, novel and unexpected to the users, where relevance has one variation, novelty has two variations and unexpectedness has four variations resulting in eight variations of serendipity definitions. These variations are explained above.

TABLE 4 Definitions of serendipity. The sign "+" indicates inclusion of a component to the definition of serendipity.

| Kind of serendipity | Relevance | Strict novelty | Motivational novelty | Unexpectedness (relevant) | Unexpectedness (find) | Unexpectedness (implicit) | Unexpectedness (recommend) |
|---|---|---|---|---|---|---|---|
| Strict serendipity (relevant) | + | + | | + | | | |
| Strict serendipity (find) | + | + | | | + | | |
| Strict serendipity (implicit) | + | + | | | | + | |
| Strict serendipity (recommend) | + | + | | | | | + |
| Motivational serendipity (relevant) | + | | + | + | | | |
| Motivational serendipity (find) | + | | + | | + | | |
| Motivational serendipity (implicit) | + | | + | | | + | |
| Motivational serendipity (recommend) | + | | + | | | | + |

#### 3.1.1.1 Effects of variations of novelty and unexpectedness

We conducted the user study in MovieLens to measure the effects of variations of unexpectedness, novelty and serendipity on users (section 1.3.2). Table 5 demonstrates statements we asked users to rate regarding each movie in the survey. Each statement was rated on the scale with the following six values (5-point Likert-scale and one "don't remember" option): "strongly agree," "agree," "neither agree nor disagree," "disagree," "strongly disagree" and "don't remember." We considered that a movie corresponded to a serendipity component (such as strict novelty or unexpectedness (relevant)) or a metric (such as user satisfaction or preference broadening), when the user rated a corresponding statement with "agree," or "strongly agree," except for unexpectedness (relevant). Due to the formulation of the statement, for this variation of unexpectedness we considered that a movie corresponded to this variation if the ratings were "neither agree nor disagree," "disagree" or "strongly disagree." All movies we asked users about in our survey were relevant to those users, since we picked only movies these users gave at least 3.5 stars out of 5.

To investigate the effects of variations of novelty, unexpectedness and serendipity, we conducted pairwise comparisons of item groups belonging to different variations. We ran cumulative link mixed-effect regression models (for more details please see paper PII) and conducted statistical tests on our findings. The following groups of relevant items were compared:

TABLE 5 Statements that we asked users to rate regarding each movie.

| Serendipity component | Statement |
|---|---|
| Strict novelty | The first time I heard of this movie was when MovieLens suggested it to me. |
| Motivational novelty | MovieLens influenced my decision to watch this movie. |
| Unexpectedness (relevant) | I expected to enjoy this movie before watching it for the first time. |
| Unexpectedness (find) | This is the type of movie I would not normally discover on my own; I need a recommender system like MovieLens to find movies like this one. |
| Unexpectedness (implicit) | This movie is different (e.g., in style, genre, topic) from the movies I usually watch. |
| Unexpectedness (recommend) | I was (or, would have been) surprised that MovieLens picked this movie to recommend to me. |
| Preference broadening | Watching this movie broadened my preferences. Now I am interested in a wider selection of movies. |
| User satisfaction | I am glad I watched this movie. |

- Each novelty variation against its corresponding non-novelty variation
- Each unexpectedness variation against its corresponding non-unexpectedness variation
- Each serendipity variation against its corresponding non-serendipity variation
- Each novelty variation against the other novelty variation
- Each unexpectedness variation against each other unexpectedness variation
- Each serendipity variation against each other serendipity variation

We found that relevance, in combination with any variation of unexpectedness or novelty, broadens user preferences more than relevance without a corresponding variation. Among items that users eventually gave high ratings to, items that users did not expect to like or to be recommended to them before consumption were less enjoyable. Users enjoyed items more when they expected to like the items and to have them recommended (or had no expectations for the items) before consuming them.

Variations of unexpectedness are different in terms of preference broadening and user satisfaction. Unexpectedness (find) and unexpectedness (implicit) outperform unexpectedness (relevant) in terms of both preference broadening and user satisfaction. Meanwhile, unexpectedness (find) outperforms unexpectedness (implicit) in terms of user satisfaction. Unexpectedness (find) includes items that users thought they would not find themselves. Unexpectedness(implicit) indicates items that users considered dissimilar to what they usually consume, while unexpectedness (relevant) indicates items that users did not expect to like prior to consuming.

#### 3.1.1.2 Effects of serendipity variations

We found that serendipitous items outperform relevant non-serendipitous items in terms of preference broadening, but we could not draw any conclusions regarding the pairwise comparisons of serendipity variations in terms of preference broadening. In particular, items that are serendipitous according to every variation of serendipity except for motivational serendipity (relevant) broaden user preferences more than corresponding relevant non-serendipitous ones.

Our results suggested that different kinds of serendipity differ in terms of user satisfaction, but the differences in results regarding comparisons of serendipitous items and corresponding non-serendipitous ones were statistically insignificant. In particular, we found that in terms of user satisfaction, motivational serendipity (find) outperforms strict serendipity (implicit) which, in turn, outperforms motivational serendipity (relevant). Meanwhile, strict serendipity (recommend) outperforms motivational serendipity (recommend) in terms of user satisfaction.

***RQ2.*** *What are the effects of items corresponding to different variations of novelty, unexpectedness and serendipity on users?*

Among relevant items, different variations of novelty, unexpectedness and serendipity generally have positive effects on preference broadening. Meanwhile, some of these variations have different effects on preference broadening and user satisfaction in comparison to each other.

### 3.1.2 Rareness of serendipitous items

In our user study, we selected movies that users watched on MovieLens, which allowed us to assess the number of serendipitous items in a typical collaborative filtering-based recommender system (section 1.3.2) (paper PII). Our sample was biased, as we selected unpopular relevant movies and users who had been using the system for at least one month. The selected movies were more likely to be serendipitous than other movies in the system according to our literature review. Our sample only represented 0.008% of ratings ($\frac{2146}{25650696}$), 0.8% of users ($\frac{475}{58855}$) and 0.6% of movies ($\frac{1689}{278477}$). We thus provided a rough upper boundary estimation of the number of serendipitous items in a typical collaborative filtering-based recommender system.

Among 2,146 ratings that users gave their feedback on, 302 (14%) were serendipitous according to at least one variation. The entire database of MovieLens contained 25,650,696 ratings, and 15,854,339 (or 61%) of them were higher than 3.5 stars (we considered these ratings relevant), which suggested that up to 8.5% ($0.14 * 0.61 \approx 0.085$) were serendipitous. We inferred the number of movie recommendations that users took in our survey based on user ratings regarding novelty (strict and motivation novelty in table 5). Our samples included 437 movies (user-movie pairs), and 302 (69%) of them were serendipitous according to at least one definition. The information regarding which recommended MovieLens movies users watched and which of them users enjoyed watching was missing. We thus provided a rough upper boundary estimation that up to 69% of recommendations provided by MovieLens that users watched were serendipitous according to at least one variation. For the rarest kind of serendipity, strict serendipity (recommend), up to 1.8% were serendipitous among all the movies in MovieLens and 14.4% were serendipitous among recommended and taken ones. For the most frequent kind of serendipity, strict serendipity (find), these numbers were 5.1% and 41.4%, respectively.

***RQ3.*** *How rare are serendipitous items among items rated by the users in a typical collaborative filtering-based recommender system? To what extent does this kind of system help users find these items?*

We assessed the ratio of serendipitous items in a typical collaborative filtering-based recommender system based on our user study (paper PII). According to our estimation, among all items users rate in a typical collaborative filtering-based recommender system, up to 8.5% are serendipitous according to at least one variation, while among recommendations provided by the system and taken by users, this proportion is as high as 69%. For the rarest kind of serendipity, strict serendipity (recommend), these ratios are 1.8% and 14.4%, while for the most frequent kind of serendipity, strict serendipity (find), they are 5.1% and 41.4%, respectively.

### 3.1.3 Assessing serendipity

In the following section, we first present ways to assess serendipity according to the literature, and then present those results of our user study related to measuring serendipity in an offline setting and designing a serendipity-oriented recommendation algorithm.

#### 3.1.3.1 Online evaluation

An online evaluation of serendipity requires asking a user questions regarding serendipity. This can be done in two ways: (1) researchers can ask a user whether the latter finds a particular item serendipitous or not, or (2) researchers can ask a user several questions, where each question corresponds to one or several components of serendipity. For example, in our user study we inquired for serendipity by asking several questions, one question per component variation. Asking a single question about serendipity might make results uncertain, as the concept of serendipity is complex and the users might interpret the complex descriptions in different ways. Meanwhile, asking users to answer several questions is more demanding for them than just one question.

#### 3.1.3.2 Offline evaluation and evaluation metrics

In the offline evaluation, researchers simulate user behavior in the system by running recommendation algorithms on datasets that were generated by real users while they were using the system in the past. Researchers split a dataset into two parts: training and test. An algorithm receives the training part as input data, then returns its predictions for the test data. Researchers simulate user behavior by comparing predictions of the algorithm with the test part of the dataset. The performance of the algorithm is measured with evaluation metrics.

Evaluation metrics to assess serendipity offline are divided into two categories (paper PI): component metrics and full metrics. Component metrics measure serendipity components separately, while full metrics measure serendipity as a whole.

For example, Vargas and Castells (2011) proposed a component metric which measures novelty and neglects other components of serendipity:

$$nov(i,u) = 1 - \frac{\|U_i\|}{\|U\|}, \tag{20}$$

where $U_i$ corresponds to the set of users who rated item $i, U_i \subseteq U$. Another serendipity evaluation metric was originally proposed by Murakami et al. (2008) and later modified by other researchers. The following is one of the modifications proposed by Ge et al. (2010):

$$ser_{ge} = \frac{\|R_u \cap REL_u \setminus PM\|}{\|R_u\|} \tag{21}$$

where $PM$ is the set of items recommended by a primitive model (recommender system). The primitive recommender system is chosen arbitrarily and required to provide suggestions with low serendipity.

Both component metrics and full metrics provide estimations of serendipity, but might cause mistakes in the assessment. A combination of component metrics might not measure serendipity of a system, as high values of separate metrics can be caused by different items, where each item has a high value in terms of one metric, but low values in terms of other metrics. Full metrics can also be confusing, since most of them depend on a primitive recommender system, while there is no agreement on this system (Kaminskas and Bridge, 2014; Kotkov et al., 2016; De Pessemier et al., 2014). Different primitive systems result in different values of the metric. Furthermore, full metrics disregard multiple judgments of serendipity. In terms of these metrics, an item is either serendipitous or not, while in a real life scenario multiple levels of judgement may appear similar to relevance (Järvelin and Kekäläinen, 2002).

***RQ4.*** *How can we assess serendipity in recommender systems?*

Serendipity in a recommender system can be assessed online or offline. In an online setting, researchers ask users whether a particular item is serendipitous with one or several questions. In an offline setting, researchers simulate user behavior using pre-collected datasets and measure the performance of algorithms with serendipity evaluation metrics. These metrics either capture serendipity as a whole (full metrics) or capture different components of the concept (component metrics).

#### 3.1.3.3 Features important for detecting serendipitous items

To design a serendipity-oriented algorithm and a serendipity metric, we first need to detect characteristics (or features) useful in detecting serendipitous items. Items can have many features, such as popularity or similarity to items in the profile of a particular user. However, such features might have different importance for detecting serendipitous items.

To find features important for detecting serendipitous items, we ran a cumulative link mixed-effect regression model (for more details please see paper PII) and statistical tests. We discovered four features that are important for detection of serendipitous movies:

- Predicted rating. MovieLens displays predicted ratings along with information about movies. The target user can choose an algorithm, which predicts their ratings. The majority of users prefer Item-Based Collaborative Filtering (IBCF) and Matrix Factorization (MF) algorithms (section 2.1.5).

- Popularity. The feature is calculated as follows:

  $$logpop_i = \ln(\|U_i\|), \tag{22}$$

  where $\|U_i\|$ is the number of ratings received by movie $i$ during the year 2016 in MovieLens (the study was conducted in 2017). We picked the number of ratings during that year instead of the overall number of ratings, as these ratings better correspond to novelty of a movie for a user. Many older movies have received more ratings, as they had a long time to gather them. However, these old movies were likely to be unfamiliar to active users in the system. More well-known movies, such as *The Shawshank Redemption*, *Toy Story* and *The Matrix* are still among the most popular movies according to the popularity metric using ratings from the year 2016.

- Average tag-based similarity to the user profile. To calculate the average tag-based distance we employed the tagging model *tag genome* (Vig et al., 2012), which is based on tags users assign to movies themselves. We calculated the distance as follows:

  $$sim\_prof(u,i) = \frac{1}{\|I_u\|} \sum_{j \in I_u, j \neq i} sim(i,j), \tag{23}$$

  where $sim(i,j)$ is the similarity measure of weighted cosine similarity in (Vig et al., 2012).

- Average collaborative similarity to user profile. We chose this feature because this is a common similarity measure in the literature on serendipity (Kaminskas and Bridge, 2014; Zheng et al., 2015). We calculated this feature according to Equation 23, where $sim(i,j)$ is the cosine similarity (equation 13) between movie rating vectors $i$ and $j$.

Our results only indicate that these four features are important for detecting serendipitous movies. However, for most features, our results do not suggest that serendipitous movies tend to have particular value ranges. For example, we cannot draw a conclusion that serendipitous movies are usually unpopular, but we can conclude that popularity is an important feature for detecting serendipitous movies. Our results allowed us to draw a conclusion regarding value ranges only for the predicted rating feature. According to our results, serendipitous movies tend to have a higher rating when predicted by a collaborative filtering algorithm, such as IBCF or MF.

Our findings mostly correspond to our literature review (paper PI), which suggests that popularity and similarity of items to items users consumed in the past are features that make items serendipitous. In particular, our literature review suggests that serendipitous items are generally less popular and similar to a user profile than non-serendipitous ones, but we could not draw these conclusions based on our results.

***RQ5.*** *What are effective features for detecting serendipitous items? What are the value ranges of these features for typical serendipitous items?*

According to our study, features that are important for detecting serendipitous items are predicted rating, popularity, content and collaborative similarity to items in a user profile. Serendipitous items have higher ratings predicted by collaborative filtering algorithms than corresponding non-serendipitous ones.

### 3.1.4 Serendipity-oriented recommendation algorithms

Serendipity-oriented algorithms can be categorized based on data they employ or based on their architecture. Categorization based on the data involves three categories: collaborative-based filtering, content-based filtering and a hybrid category. There are two categorizations based on the architecture of algorithms: design categorization and paradigm categorization.

Design categorization includes three categories:

1. Reranking algorithms (Reranking). To improve serendipity, an algorithm can leverage ratings predicted by an accuracy-oriented algorithm and rerank the output of a recommender system.

2. Serendipitous-oriented modification (Modification). This category represents modifications of accuracy-oriented algorithms. The main difference between modification and reranking is that reranking algorithms can use any accuracy-oriented algorithms that assign ranking scores to items, while modifications can only be applied to a particular algorithm. For example, an MF algorithm with a different objective function than the original algorithm (Zheng et al., 2015) can be regarded as a modification.

3. Novel algorithms (New). Serendipity-oriented algorithms, which do not correspond to reranking or modification categories, belong to this category.

Paradigm categorization is based on the step responsible for the improvement of serendipity:

- Pre-filtering. A recommendation algorithm preprocesses the input data for an accuracy-oriented algorithm to improve serendipity.
- Post-filtering. A recommendation algorithm reranks results of accuracy-oriented algorithms.
- Modeling. Actions for serendipity improvement can be taken in the phase of generating recommendations.

#### 3.1.4.1 Utility model

An example of a reranking algorithm that uses the post-filtering paradigm is the algorithm based on utility (Adamopoulos and Tuzhilin, 2014). The algorithm orders items in a recommendation list based on their scores. For each item, the score is based on the combination of two metrics: unexpectedness utility and quality utility. Quality utility is based on ratings provided by an accuracy-oriented recommendation algorithm.

To calculate unexpectedness utility, the algorithm generates a set of expected items for the target user. The set includes items in the target user profile and items in a suitable way similar to these items. The unexpectedness utility is based on similarity of an item to items in the set of expected items.

#### 3.1.4.2 K-furthest neighbor

An example of a modification algorithm that uses the pre-filtering paradigm is the k-Furthest Neighbor (kFN) algorithm (Said et al., 2013). kFN is based on UBCF (section 2.1.5.1). To overcome the popularity bias of recommendations generated by UBCF, kFN suggests items disliked by users dissimilar to the target user. The algorithm uses the dissimilarity measure, which is calculated based on a similarity measure, such as Pearson correlation (equation 11) or cosine similarity (equation 13), where the ratings of one of the two users are inverted. For example, if a user rated items $(i1, i2, i3)$ with ratings $(1, 1, 4)$, the inverted ratings for this user would be $(5, 5, 2)$.

#### 3.1.4.3 Random walk with restarts enhanced with knowledge infusion

An example of an algorithm that belongs to the category of novel serendipity-oriented algorithms and uses the modeling paradigm is Random Walk With Restarts Enhanced With Knowledge Infusion (RWR-KI) (de Gemmis et al., 2015). The algorithm orders items in the recommendation list according to their relatedness to items in the target user profile. The relatedness is inferred by Random Walk With Restarts (RWR) operating on a similarity graph. In the similarity graph, nodes correspond to items, while weighted edges correspond to similarities between connected nodes. The weights of the edges correspond to similarity values. RWR receives the similarity graph and the set of starting nodes, $I_u$ as input values and returns the relatedness of each item to the items from the target user profile. RWR-KI orders items according to their relatedness scores, removes items already rated by the user and suggests the recommendation list to the user.

***RQ6.** What are the state-of-the-art recommendation algorithms that suggest serendipitous items?*

In our literature review (paper PI), we overviewed serendipity-oriented algorithms and provided two classifications: based on architecture and based on paradigms.

### 3.1.5 Challenges and future directions

Designing a serendipity-oriented recommendation algorithm requires to choosing suitable objectives. It is therefore necessary to investigate how to assess serendipity in recommender systems, which requires a definition of the concept. Serendipity is difficult to investigate because (a) it contains emotional dimensions and is therefore very unstable and (b) the lack of observations makes it difficult to make assumptions regarding serendipity that would be reasonable in most cases.

***RQ7.** What are the challenges of serendipity in recommender systems?*

According to our conference literature review (paper PIII), suggesting serendipitous items involves four main challenges: disagreement on the definition of serendipity, difficulty in measurement, presence of an emotional dimension and lack of serendipitous encounters.

***RQ8.** What are the future directions of serendipity in recommender systems?*

In our literature review (paper PI), we indicated four future directions of serendipity-oriented algorithms. The first direction indicates that serendipity-oriented algorithms and evaluation metrics should take into account both item popularity and similarity to items in a user profile. The second direction involves context-aware recommender systems (Adomavicius and Tuzhilin, 2015). Because context-aware recommender systems can pick items based on the context rather than user tastes, they might suggest more serendipitous items. The third direction includes cross-domain recommender systems (Cantador et al., 2015). Having information from additional domains, a recommender system can infer which items are familiar to a user, suggest items relevant to the user and select items that fit the context. The fourth direction is dedicated to group recommendations (Efthymiou et al., 2017; Masthoff, 2015). It is difficult to suggest items serendipitous to each user in a group of individuals, because the tastes of each user must be considered.

## 3.2 Serendipity in cross-domain recommender systems

To investigate serendipity and accuracy in cross-domain recommender systems, we conducted three sets of offline experiments. We conducted the first set of experiments to investigate whether source domains can improve accuracy in the target domains, when the domains differ on an item level and only sets of users overlap (paper PVI). We collected data from three online social networks: Twitter, Foursquare and Instagram. We used these data to improve recommendation accuracy in Foursquare. We extracted four feature spaces from each domain: venue categories (Foursquare), Linguistic Inquiry and Word Count (Twitter) (Francis and Booth, 1993), Latent Dirichlet Allocation (Twitter) (Blei et al., 2003) and image concepts (Instagram). We used these feature spaces to calculate user similarities in User-Based Collaborative Filtering (UBCF), which generated recommendations of Foursquare venue categories to users. We then combined outputs from UBCF employing each feature space and achieved an increase in accuracy of recommending venue categories to users.

In the second set of experiments (paper PVII), we investigated whether a source domain can improve accuracy in the target domain when only items overlap on a system level. We collected data from the online social network vk.com and music recommendation service last.fm regarding audio recordings to which users of these websites listen. We employed Item-Based Collaborative Filtering (IBCF) and content based filtering to generate recommendations of audio recordings to users in the dataset from vk.com. We then added different portions of data from last.fm to the data from vk.com and observed an increase of recommendation accuracy in the dataset collected from vk.com. Our results also suggested that with the increase of data from the source domain, the accuracy in the target domain increases.

Finally, in the third set of experiments (paper PVIII), we investigated whether a source domain can improve both accuracy and serendipity in the target domain when only items overlap on a system level. We conducted the same experiments as in the second set of experiments. The main differences were that we also measured serendipity of the generated recommendations and used slightly different amounts of data from the source domain. Our results demonstrated an increase of accuracy and serendipity in the target domain, when the data from both domains were combined.

***RQ9.*** *Can source domains improve accuracy in the target domain when only users overlap?*

According to our results (paper PVI), source domains can improve accuracy in a target domain when only users overlap. In our experiment, we used data of the same users collected from the three popular online social networks: Foursquare, Instagram and Twitter. We improved recommendation performance in Foursquare by combining user data from Instagram, Twitter and Foursquare.

***RQ10.*** *Can a source domain improve accuracy in the target domain when only items* *overlap?*

According to our results (papers PVII and PVIII), a source domain can improve accuracy in the target domain when only items overlap on a system level. Our results suggest that the integration of the source domain results in a decrease of accuracy for content-based filtering and an increase of accuracy for collaborative filtering. Our results indicate that the more items overlap in source and target domains with respect to the whole dataset the higher the improvement of accuracy for collaborative filtering.

***RQ11.*** *Can a source domain improve serendipity in the target domain when only items overlap?*

According to our results (paper PVIII), the source domain can improve serendipity in the target domain when only items overlap on a system level for both collaborative filtering and content-based filtering algorithms.

## 3.3 Contributions

In this section, we describe the four contributions of this dissertation: a definition of serendipity, serendipity evaluation metric, serendipity-oriented algorithm and serendipity-oriented dataset.

### 3.3.1 A definition of serendipity in recommender systems

Based on the literature review (paper PI), we operationalized eight variations of serendipity in recommender systems (***RQ1***). Each variation includes three components: relevance, novelty and unexpectedness, where novelty and unexpectedness have multiple variations.

### 3.3.2 A serendipity evaluation metric

To measure serendipity in an offline experiment, we designed a serendipity metric prior to our user study. The metric is based on assumptions regarding serendipity from our literature review (paper PI). Our serendipity metric is based on a traditional full metric that was originally proposed by Murakami et al. (2008). Although the traditional serendipity metric successfully captures the relevance and popularity of recommendations, it disregards similarity of recommendations to items in the user profile. Our metric takes this feature into account by using the set of potentially unexpected items $UNEXP_u$. This set contains items that have features unfamiliar to the user (the user has not yet rated any items with these features). We add an item to $UNEXP_u$, if this item has at least one feature new to user $u$. Our serendipity metric is calculated as follows:

$$Ser_u@n = \frac{\|(RS_u(n) \backslash PM) \cap REL_u \cap UNEXP_u\|}{n}, \tag{24}$$

where $PM$ is the set of items recommended by a primitive recommender system (more details in section 3.1.3.2). We tested our serendipity metric on two MovieLens datasets: 100K ML and HetRec. The inclusion criteria for the set can vary depending on the application scenario. In our experiments, we populated the set of unexpected items, $UNEXP_u$, with movies that have at least one genre new to the target user.

### 3.3.3 A serendipity-oriented greedy algorithm

In papers PIV and PV, we proposed the Serendipity-Oriented Greedy (SOG) algorithm, which is based on the Topic Diversification (TD) algorithm. According to our classification (section 3.1.4), the SOG algorithm is a reranking algorithm that uses the post-filtering paradigm.

```
Algorithm 1: Description of the SOG algorithm using pseudocode
Input  : $RS_u(n)$: top–n recommendations for user $u$
Output: $Res$: recommendation list
$\hat{r}_{ui}$: predicted rating of item $i$ for user $u$,
$Res = <>$;
$C = RS_u(n)$;
$i' = i$ with max $\hat{r}_{ui}, i \in C$;
$Res[0] = i'$;
$C = C \setminus \{i'\}$;
while $\|Res\| < n$ do
    $B = set(Res)$;// $set$ converts a list to a set
    $i\ = i$ with max $g(u, i, B), i \in C$;
    Add $i'$ to the top of $Res$;
    $C \setminus \{i'\}$;
end
```

The main idea of both reranking algorithms, SOG and TD, is described by Algorithm 1. The reranking algorithm receives a set of $n$ recommendations provided by an accuracy-oriented algorithm $RS_u(n)$ (table 2) along with ratings predicted for these recommendations $r_{ui}$. The reranking algorithm initializes the set of candidate items, $C$, with the set of top-n recommendations, $RS_u(n)$. The algorithm then iteratively removes items from the candidate set, $C$, and adds them to the top of the recommendation list $Res$. The algorithm chooses all the items from the candidate set $C$, except for the first item, based on the value of scoring function $g(u, i, B)$. The algorithm chooses the first item based on the rating predicted by the accuracy-oriented algorithm.

The main difference between the SOG and TD algorithms is the scoring function $g(u, i, B)$. In the case of TD, the function represents an adjustable trade-off between diversification of the list $Res$ and predicted relevance of items, while the function employed by the SOG algorithm is an adjustable trade-off between diversification, relevance and similarity of items to items in the target user profile.

For the SOG algorithm, the function is calculated as follows:

$$g(u,i,B) = (1-\Theta_F)\cdot \hat{r}_{ui} + \Theta_F \cdot c(u,i,B), \tag{25}$$

$$c(u,i,B) = d_{iB} + \Theta_S \cdot (\max_{f\in(F_i\setminus F_u)}(\hat{r}_{uf}) + unexp_{ui}), \tag{26}$$

where $\Theta_S$ and $\Theta_F$ are a serendipity weight and damping factor responsible for serendipity and diversity of the recommendation list *Res*, respectively. Rating $\hat{r}_{uf}$ is a predicted rating of feature $f$ for user $u$ (table 2), which represents how likely user $u$ is to enjoy consuming an item having feature $f$. If the target user is familiar with all the features of our dataset $F_i\setminus F_u = \emptyset$, then $\max_{f\in(F_i\setminus F_u)}(\hat{r}_{uf}) = 0$. Feature ratings and item ratings are normalized to the range $[0,1]$. Unexpectedness $unexp_{ui}$ depends on the number of features new to the user $u$ that appear in item $i$:

$$unexp_{u,i} = \frac{\|F_i\setminus F_u\|}{\|F\setminus F_u\|}, \tag{27}$$

where $F_i$ and $F_u$ are feature sets of item $i$ and of items rated by user $u$, respectively. Dissimilarity $d_{iB}$ indicates how dissimilar item $i$ is to the set of items added to the recommendation list, $B$. Dissimilarity is responsible for diversification of recommendation list *Res* and is calculated as follows:

$$d_{iB} = \frac{1}{\|B\|}\sum_{j\in B} 1 - jacc(i,j), \tag{28}$$

Where $jacc(i,j)$ is the Jaccard similarity between items $i$ and $j$ based on item features:

$$jacc(i,j) = \frac{\|F_i \cap F_j\|}{\|F_i \cup F_j\|}. \tag{29}$$

Feature ratings $\hat{r}_{uf}$ are predicted by the MF algorithm, which receives a user-feature matrix as an input. In the user-feature matrix, each feature rating is an average of ratings given by a user to all items that have this feature:

$$r_{uf} = \frac{1}{\|I_{uf}\|}\sum_{i\in I_{uf}} r_{ui}, \tag{30}$$

where $I_{uf}$ is a set of items that have feature $f$ and that are rated by user $u$. To predict a feature rating, we run the MF algorithm on the user-feature matrix.

The computational complexity of the algorithm is $\mathcal{O}(n^3)$ (excluding precalculation), where $n$ is the number of items in the top-n recommendations $RS_u(n)$.

Our algorithm has three key differences when compared to TD:

- The SOG algorithm takes into account relevance scores instead of positions of items in the top-n recommendations provided by an accuracy-oriented algorithm.
- The SOG algorithm employs not only the damping factor responsible for diversity, but also the serendipity weight.

- The SOG algorithm predicts how likely a user is to like a particular feature of a recommended item.

The SOG algorithm has four main advantages:

1. The objective function of the algorithm corresponds to the common definition of serendipity. The algorithm takes into account relevance by using ratings provided by an accuracy-oriented algorithm, unexpectedness by picking items with features new to users and novelty by picking items that users are likely to be unfamiliar with, because they contain features new to the users.
2. The SOG algorithm takes into account both serendipity and diversity.
3. The SOG algorithm can receive ratings from any accuracy-oriented algorithm, which might be useful for an online recommender system. The reranking can be done on the client side of a client-server application.
4. The SOG algorithm is flexible and can be adjusted with parameters $\Theta_F$ and $\Theta_S$. The parameters can have different values for different users.

We evaluated our algorithm and compared it with the state-of-the-art serendipity-oriented algorithms and TD using two datasets, 100K ML (Harper and Konstan, 2015) and HetRec (Cantador et al., 2011). According to our results, our algorithm outperforms the state-of-the-art serendipity-oriented algorithms in terms of diversity and serendipity, but underperforms them in terms of accuracy. Compared with TD, our algorithm has higher accuracy and serendipity, but slightly lower diversity.

### 3.3.4 A serendipity-oriented dataset

In our user study, we surveyed users regarding serendipity in MovieLens. To inspire future efforts on serendipity in recommender systems, we published the dataset collected during the study. The dataset is available on the GroupLens website[1].

Note that the published dataset is slightly different from the one used in paper PII, because these data were updated, while we were reporting the results.

The dataset contains user answers to our questions and additional information, such as past ratings of these users, recommendations they received before replying to our survey and movie descriptions. The dataset contains user ratings given from November 11, 2009, until January 6, 2018. The dataset was generated on January 15, 2018. Overall, there are 10,000,000 ratings. The dataset contains the following information:

- User answers. We included user answers to our survey. The survey included eight statements (one statement for user satisfaction, one for preference broadening, two for novelty and four for unexpectedness) and one

[1] https://grouplens.org/datasets/serendipity-2018/

question asking how long ago the user watched the movie (section 1.3.2). Overall, we published the answers of 481 users regarding 2,150 ratings (user-movie pairs).

- User ratings. The dataset also contains user ratings given or updated (users can update their ratings at any time) from November 11, 2009, until January 6, 2018. The dataset contains all the ratings of the 481 users given in the specified period of time.
- User recommendations. We also included the last eight recommendations that the 481 users received prior to the user study.
- Movie descriptions. We provided description of each movie used in the dataset. The movie descriptions included release dates, lists of directors, cast, genres and IDs.
- Tag genome description. We included the tag genome description for 11,005 movies in our dataset, because these data were available only for these movies. The tag genome represents a number of features inferred from tags and ratings users assign to movies in MovieLens.

# 4 SUMMARY OF THE ORIGINAL ARTICLES

In this chapter, we provide summaries of articles included in this dissertation.

## 4.1 Article PI: "A survey of serendipity in recommender systems. Knowledge-Based Systems"

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. Knowledge-Based Systems, 111:180-192, 2016. doi: http://dx.doi.org/10.1016/j.knosys.2016.08.014.

**Research problem**

In this paper, we present a literature review on serendipity in recommender systems. We review (a) definitions of serendipity (***RQ1***), (b) the state-of-the-art serendipity-oriented recommendation algorithms (***RQ6***), (c) serendipity metrics (***RQ4***) and (d) future directions of the topic (***RQ8***).

**Results**

We found that there was no agreement on the definition of serendipity in recommender systems, but most researchers indicated that serendipity includes relevance, novelty and unexpectedness. We developed two categorizations for serendipity-oriented recommendation algorithms based on algorithm architecture (reranking, modification and new) and on the paradigm (pre-filtering, modeling and post-filtering). We developed a categorization for serendipity metrics: full metrics and component metrics. Finally, we provided future directions of serendipity in recommender systems: (a) further development of serendipity-oriented algorithms and serendipity evaluation metrics, (b) serendipity in context-aware recommender systems, (c) serendipity in cross-domain recommender systems and (d) serendipity in group recommendations.

**Author's contribution**

The author conducted the literature review and completed the first draft of the article. The coauthors provided their comments regarding the content and organization of the article. The final version of the article was written in close collaboration with the coauthors.

## 4.2 Article PII: "Investigating Serendipity in Recommender Systems Based on Real User Feedback"

Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. Investigating Serendipity in Recommender Systems Based on Real User Feedback. Proceedings of SAC 2018: Symposium on Applied Computing, 2018 (Accepted).

**Research problem**

This paper presents (a) variations of novelty, unexpectedness and serendipity that are supported by the literature review (***RQ1***); (b) the effects of these variations on users (***RQ2***); (c) upper boundary estimation of the proportion of serendipitous items (***RQ3***) and (d) features important for detecting serendipitous items (***RQ5***).

**Results**

To investigate variations of novelty, unexpectedness and serendipity, we conducted the user study in MovieLens, where we surveyed users regarding movies they found serendipitous. The study allowed us to collect and publish the first serendipity-oriented dataset.

We found that among relevant items, different variations of novelty, unexpectedness and serendipity generally have positive effects on preference broadening. Meanwhile, these variations have different effects on preference broadening and user satisfaction in comparison to each other.

Based on our user study, we assessed the proportion of serendipitous items in a typical collaborative filtering-based recommender system. According to our estimation, among all items users rate in a typical collaborative filtering-based recommender system, up to 8.5% are serendipitous according to at least one variation, while among recommendations provided by the system and taken by users, this proportion is up to 69%. For the rarest kind of serendipity, strict serendipity (recommend), these ratios are 1.8% and 14.4%, while for the most frequent kind of serendipity, strict serendipity (find), they are 5.1% and 41.4%, respectively.

We found that features important for detecting serendipitous items are predicted ratings, popularity, content and collaborative similarity to items in a user profile. We also found that serendipitous items have higher predicted ratings than corresponding non-serendipitous ones.

**Author's contribution**

The author analyzed ways to conduct the user study, implemented the survey in MovieLens, conducted the study, analyzed collected data, prepared the collected dataset for publication and completed the first draft of the article. The coauthors provided their comments and guidance on each stage of the research during the whole research process, including but not limited to preparation for the user study, implementation of the survey, analysis and communication of the results of the study. The final version of the article was written in close collaboration with the coauthors. The research was also conducted in collaboration with the GroupLens team and the MineSocMed project members, who are not the coauthors of the article, but who also provided their comments and guidance during this research.

## 4.3 Article PIII: "Challenges of Serendipity in Recommender Systems"

Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. Challenges of serendipity in recommender systems. In Proceedings of the 12th International Conference on Web Information Systems and Technologies, volume 2, pages 251-256. SCITEPRESS, 2016. doi: 10.5220/0005879802510256.

**Research problem**

This paper presents a literature review different from the one provided in paper PI and discusses challenges of serendipity in recommender systems (***RQ7***).

**Results**

We discussed four challenges of serendipity in recommender systems: the disagreement on the definition of serendipity, the presence of the emotional dimension, the lack of serendipitous encounters and the disagreement on serendipity metrics. The first challenge is that there is no consensus on the definition of serendipity, which causes a disagreement on ways to measure serendipity. The second challenge is that serendipity depends on users' emotions more than relevance, since relevance is a component of serendipity. The third challenge is that it is difficult to study serendipity, because serendipitous encounters are very rare. The final challenge is related to measuring serendipity, because there is no agreement on its definition and it is difficult to draw reliable conclusions due to the lack of serendipitous encounters.

**Author's contribution**

The author conducted the literature review, came up with the challenges and completed the first draft of the article. The final version of the article was completed in close collaboration with the coauthors, who provided their comments and contributed to the content.

## 4.4 Article PIV: "A Serendipity-Oriented Greedy Algorithm for Recommendations"

Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. A serendipity-oriented greedy algorithm for recommendations. In Proceedings of the 13th International conference on web information systems and technologies. SCITEPRESS, 2017 (Best paper award). doi: 10.5220/0006232800320040

**Research problem**

This paper presents our serendipity-oriented recommendation algorithm (contribution 3).

**Results**

We described our serendipity-oriented recommendation algorithm and compared it to the state-of-the-art serendipity-oriented algorithms. We measured accuracy, diversity and serendipitiy. According to our results, our algorithm outperforms the baseline algorithms in terms of serendipity and diversity, but underperforms them in terms of accuracy.

**Author's contribution**

The author implemented the baseline algorithms, designed and evaluated the serendipity-oriented algorithm and completed the first draft of the article. The final version of the article was completed in a close collaboration with the coauthors, who provided their comments and contributed to the content.

## 4.5 Article PV: "A Serendipity-Oriented Greedy Algorithm and a Complete Serendipity Metric for Recommendation Purposes"

Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. A Serendipity-Oriented Greedy Algorithm and a Complete Serendipity Metric for Recommendation Purposes. Springer Computing (Submitted for publication).

**Research problem**

This paper presents our serendipity-oriented recommendation algorithm (contribution 3) and serendipity evaluation metric (contribution 2).

**Results**

This article is an extended version of Article PIV. In this article, we present our serendipity-oriented algorithm and serendipity metric. We conduct the same experiments as in Article PIV but provide more details on our serendipity evaluation metric.

**Author's contribution**

The author implemented the baseline algorithms, designed and evaluated the serendipity-oriented algorithm and completed the first draft of the article. The final version of the article was completed in close collaboration with the coauthors, who provided their comments and contributed to the content.

## 4.6 Article PVI: "Cross-Social Network Collaborative Recommendation"

Aleksandr Farseev, Denis Kotkov, Alexander Semenov, Jari Veijalainen, and Tat-Seng Chua. Cross-social network collaborative recommendation. In Proceedings of the ACM Web Science Conference, pages 38:1-38:2. ACM, 2015. doi: 10.1145/2786451.2786504.

**Research problem**

This paper is dedicated to cross-domain recommender systems. We investigate whether it is possible to improve accuracy in the target domain by enriching our dataset with information from source domains when domains differ on an item level and only sets of users overlap (***RQ9***).

**Results**

We utilized a subset of the NUS-MSS dataset (Farseev et al., 2015), which contains data on the same users from Foursquare, Twitter and Instagram. We used data from each online social network to calculate similarities between users for User-Based Collaborative Filtering (UBCF) (section 2.1.5). Each source of information corresponded to one or two feature spaces. Each baseline algorithm was represented by UBCF employing a particular feature space. We also employed two ways of combining results provided by these baselines. We demonstrated that one of the combinations outperformed single-domain baselines, which suggests that source domains can improve recommendation accuracy in the target domain when only users overlap and the domains are different on an item level.

**Author's contribution**

The author evaluated user-based collaborative filtering utilizing data from a single domain and contributed to the content of the paper. The data collection, feature extraction and combination of results of multiple baselines were performed by the coauthors. The final version of the article was completed in close collaboration with the coauthors.

## 4.7 Article PVII: "Cross-domain recommendations with overlapping items"

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. Cross-domain recommendations with overlapping items. In Proceedings of the 12th International conference on web information systems and technologies, volume 2, pages 131-138. SCITEPRESS, 2016. doi: 10.5220/0005851301310138.

**Research problem**

In this paper, we investigate whether the source domain can improve accuracy in the target domain when only items overlap on a system level (***RQ10***).

**Results**

We collected two datasets related to music from two websites: the online social network vk.com and the music recommendation service last.fm. The datasets contained ratings that users assigned to audio recordings on these two websites. We matched the two datasets using song metadata and built three datasets by enriching the dataset collected from vk.com with different portions of data from last.fm. We then measured accuracy of content-based filtering and item-based filtering (section 2.1.5) on these three datasets and found that the source domain can improve accuracy in the target domain when only items overlap on a system level. In fact the higher the portion of overlapping items the higher the improvement.

**Author's contribution**

The author collected the datasets, implemented and evaluated recommendation algorithms and completed the first draft of the article. The coauthors provided their comments regarding the experiments, the content and the organization of the article. The final version of the article was written in close collaboration with the coauthors.

## 4.8 Article PVIII: “Improving Serendipity and Accuracy in Cross-Domain Recommender Systems”

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. Improving Serendipity and Accuracy in Cross-Domain Recommender Systems. Web Information Systems and Technologies, 2017. doi: 10.1007/978-3-319-66468-2_6

**Research problem**

In this paper, we investigate whether the source domain can improve accuracy and serendipity in the target domain when only items overlap on a system level (***RQ10*** and ***RQ11***).

**Results**

This paper is the extended version of Article PIV. In addition to investigating the improvement of accuracy, we also investigated the improvement of serendipity in cross-domain recommender systems. We conducted the same experiments with slightly different portions of overlapping recordings, but also measured serendipity. We found that the source domain can improve serendipity in the target domain when only items overlap on a system level.

**Author’s contribution**

The author collected the datasets, implemented and evaluated recommendation algorithms and completed the first draft of the article. The coauthors provided their comments regarding the content and organization of the article. The final version of the article was written in close collaboration with the coauthors.

# 5 CONCLUSIONS

In this dissertation, we answered eleven research questions and made four contributions. We reviewed definitions of serendipity in recommender systems, ways to assess serendipity and serendipity-oriented recommendation algorithms. We investigated the effects of different variations of serendipity on users and assessed a portion of serendipitous items in a typical collaborative filtering-based recommender system. We also looked at serendipity and accuracy in the context of cross-domain recommender systems. Finally, we discussed challenges and future directions of serendipity in recommender systems.

We conducted a literature review and found that although there was no consensus on the definition of serendipity in recommender systems, most researchers agreed that serendipitous items are relevant, novel and unexpected to a user. We also found that definitions of novelty and unexpectedness vary, which resulted in eight variations of serendipity. In our literature review, we reviewed serendipity evaluation metrics and proposed their categorization. We also reviewed serendipity-oriented recommendation algorithms and proposed two categorizations. Based on the reviewed literature, we discussed the challenges of serendipity in recommender systems and suggested future directions of this topic.

We conducted the user study, in which we surveyed users of the movie recommender system MovieLens. We found that all variations of serendipity, novelty and unexpectedness have positive effects on user preference broadening, but these variations have different effects on preference broadening and user satisfaction when they are compared with each other. We also estimated the proportion of serendipitous items in typical collaborative filtering-based recommender systems. We only provided an upper boundary estimation because our sample was biased toward serendipitous items and it only represented around 0.008% of ratings, 0.8% of users and 0.6% of items in the recommender system. Among items users rate in a recommender system, up to 8.5% are serendipitous according to at least one variation of serendipity, while among recommendations that users receive and follow in this system, this ratio is up to 69%.

We initially investigated serendipity and accuracy in cross-domain recommender systems by conducting experiments on pre-collected datasets. We found that a source domain can improve serendipity and accuracy in the target domain when only items overlap on a system level. We also found that source domains can improve recommendation accuracy in the target domain when the overlap happens in the sets of users and domains are different on an item level.

The four contributions of this dissertation are the definition of serendipity, the serendipity evaluation metric, the serendipity-oriented recommendation algorithm and the serendipity-oriented dataset. Our definition is based on a literature review and evaluated in our user study. According our definition, serendipity includes three components: relevance, novelty and unexpectedness, where novelty has two variations and unexpectedness has four variations. This results in eight variations of serendipity. Our serendipity metric is based on a traditional metric and captures popularity of items and their similarity to items in the user profile. Our Serendipity-Oriented Greedy (SOG) algorithm is a reranking algorithm, which is based on the existing greedy algorithm, Topic Diversification (TD). Our algorithm outperforms other serendipity-oriented recommendation algorithms in terms of serendipity and diversity and underperforms them in terms of accuracy. To the best of our knowledge, our serendipity-oriented dataset is the first publicly available dataset that contains user feedback on serendipitous items.

## 5.1 Limitations

We used three methods for our research (section 1.3): the literature review, the user study and offline evaluation. Each of these methods had a number of limitations.

Our literature review might lack relevant sources for two reasons. First, because we used Google Scholar, Scopus and Web of Science to find articles, our literature review is missing articles that are not indexed by these three search engines. Second, our literature review might also lack sources that discuss serendipity using different terms than those we used in our search queries, such as “non-obviousness” or “luck”.

Our offline evaluation has two limitations. First, to demonstrate our algorithm and serendipity metric, we used datasets collected in MovieLens. The results of evaluation using datasets collected in another recommender system might produce different results. Second, in general, offline evaluation might not correspond to a real-life scenario (Garcin et al., 2014).

In our user study, we only looked at two user metrics: preference broadening and user satisfaction. The decision was based on the literature review, but other metrics, such as user retention or trust (Gunawardana and Shani, 2015), are also important to investigate. The users that we selected for our survey are more active than the rest of the users in the system meaning that they do not represent the whole population of MovieLens users. Compared to the number of ratings in MovieLens, the number of ratings we received user feedback on can be considered insufficient. We analyzed user replies regarding 2,146 ratings, which represented only around 0.008% of ratings, 0.8% of users and 0.6% of movies in MovieLens. Finally, to receive enough observations of serendipitous items for our analysis, we selected items that were more likely to be serendipitous than other items, which limited our findings to the investigation of serendipitous items among relevant ones.

We conducted the user study after designing our algorithm and evaluation metric, meaning that both the algorithm and the metric employ features slightly different from those important for detecting serendipitous items. However, both artifacts can be adjusted according to the discovered features.

## 5.2 Discussion

Our findings indicate that serendipitous items are worth suggesting because they broaden user preference usually without hurting user satisfaction, and given our sample, serendipitous items do not seem to be too rare to make an attempt to recommend them. Depending on the application scenario and context, serendipitous items can be very useful, as mentioned by Smith and Linden (2017):

> *When someone is clearly seeking something specific, recommendations should be narrow to help them quickly find what they need. But when intent is unclear or uncertain, discovery and serendipity should be the goal.*

The findings of our study and experiments mostly correspond to findings reported in other studies and experiments. For example, most studies on serendipity in recommender systems indicate that popularity, content and collaborative similarities of an item to items in the user profile are important features for detecting serendipitous items (Adamopoulos and Tuzhilin, 2014; Kaminskas and Bridge, 2014, 2016; Lu et al., 2012; Murakami et al., 2008; Zhang et al., 2012; Zheng et al., 2015). However, one of our findings suggests that the higher the rating predicted by a collaborative filtering algorithm, the higher the serendipity, which contradicts most studies on the topic. This finding suggests that collaborative algorithms, such as MF or IBCF, are good at suggesting serendipitous items because the ratings they provide correlate with serendipity, at least among relevant items. However, further research is needed to draw a more informed conclusion.

Compared to other topics in recommender systems, such as relevance or diversity, serendipity still remains an underexplored topic, which opens opportunities for researchers to investigate it.

## 5.3 Further research

We indicated future directions of serendipity in recommender systems in our answer to ***RQ8*** (section 3.1.5). We indicated the four following directions: further development of serendipity-oriented algorithms and serendipity metrics that take into account features important for detecting serendipitous items, serendipity in context-aware recommender systems, serendipity in cross-domain recommender systems and serendipity in group recommendations.

In addition to these directions, further research on serendipity in recommender systems should include more experiments involving real users in online recommender systems because serendipity is a complex concept that significantly depends on the emotional responses of users, which are difficult to simulate in an offline experiment. Even offline evaluations regarding relevance often fail to represent real-life scenarios (Garcin et al., 2014). Meanwhile, serendipity not only includes relevance, but also novelty and unexpectedness.

We encourage researchers to conduct more experiments regarding the effects of serendipity on users and consider not only preference broadening and user satisfaction, but also other metrics, such as trust or risk (Gunawardana and Shani, 2015). It would be useful to investigate how serendipity affects user retention and rating activity and when users need serendipity the most.

# YHTEENVETO (FINNISH SUMMARY)

## Onnekkuus suosittelujärjestelmissä

Internetin sähköisten kauppapaikkojen runsas sisältötarjonta (kuten majoituspalvelut tai suoratoistomusiikki) tekee käyttäjille mahdottomaksi tutustua siihen järkevässä ajassa. Suosittelujärjestelmät ovat avustavia järjestelmiä, jotka on alunperin suunniteltu auttamaan sivuston käyttäjiä löytämään sivustolta kiinnostavia tavaroita ja palveluita (nimikkeitä), kun niiden määrä on suuri. Perinteisesti suosittelujärjestelmät on optimoitu saavuttamaan korkea suositustarkkuus, jota mitataan sillä, kuinka usein käyttäjä ostaa tai kuluttaa järjestelmän suosittelemen nimikkeen. Suositustarkkuuden kohottamiseksi suosittelujärjestelmät ehdottavat usein käyttäjille nimikkeitä, jotka ovat yleisesti suosittuja ja sopivassa mielessä samankaltaisia käyttäjän aiemmin ostamien tai kuluttamien nimikkeiden kanssa. Tämän seurauksena käyttäjät usein menettävät kiinnostuksensa käyttää sivuston suosittelujärjestelmää tai jopa koko sivustoa, koska suositellut nimikkeet ovat jo entuudestaan tuttuja tai ne ovat muuten helposti löydettävissä ilman apuakin.

Yksi tapa lisätä käyttäjätyytyväisyyttä ja käyttäjien sitoutumista on suosittaa käyttäjälle ns. onnekkaita nimikkeitä. Sellaisia käyttäjät eivät omatoimisesti löytäisi, eivätkä edes huomaisi etsiä, mutta olisivat tyytyväisiä tutustuttuaan niihin järjestelmän suosituksen pohjalta. Onnekkuuden käsitettä ei ole tutkittu perusteellisesti suosittelujärjestelmien yhteydessä. Tällä hetkellä ei ole edes yhteistä näkemystä siitä, mitä alakäsitteitä onnekkuuden määritelmään pitäisi sisällyttää. Tässä väitöskirjassa onnekkaina pidetään suositeltuja nimikkeitä, jotka ovat käyttäjän näkökulmasta asiaankuuluvia, uusia ja yllätyksellisiä.

Tässä väitöskirjassa: tutustutaan erilaisiin onnekkuuden määritelmiin kirjallisuudessa ja evaluoidaan niiden toimivuutta käyttäjätutkimuksessa, arvioidaan onnekkaiden nimikesuositusten esiintyvyyttä yhdessä suosittelujärjestelmässä, käydään läpi tapoja mitata suositusten onnekkuutta ja parantaa sitä, tutkitaan onnekkuutta monialuesuosittelijoissa, jotka hyödyntävät useampia nimiketyyppejä (esim. elokuvat, musiikki, kirjat) suosituksia tuottaessaan; ja tarkastellaan onnekkaita nimikesuosituksia tuottavien suosittelujärjestelmien haasteita ja kehityssuuntia.

Työssä on sovellettu suunnittelutieteen metodologista kehystä ja kehitetty neljä artefaktia: (1) kahdeksan ehdokasta onnekkuuden määritelmäksi, (2) suositeltujen nimikkeiden onnekkuuden mittari; (3) onnekkaita suosituksia generoiva algoritmi; (4) elokuvasuositusten onnekkuutta mittaava datajoukko, joka on koottu MovieLens-suosittelujärjestelmän käyttäjäpalautekyselyn avulla. Nämä artefaktit on evaluoitu käyttäen tieteellisiä menetelmiä ja kommunikoitu tiedeyhteisölle julkaisujen muodossa.

Avainsanat: suosittelujärjestelmät, onnekkuus, asianmukaisuus, uutuus, yllätyksellisyys, personointi, suosittelualgoritmit, evaluointi, evaluointimetriikat, simuloidut käyttäjäkokeet, käyttäjäpalautekyselyt, onnekkuuden mittarit

## REFERENCES

Adamopoulos, P. & Tuzhilin, A. 2014. On unexpectedness in recommender systems: Or how to better expect the unexpected. ACM Transactions on Intelligent Systems and Technology 5 (4), 1–32.

Adomavicius, G. & Kwon, Y. 2012. Improving aggregate recommendation diversity using ranking-based techniques. IEEE Transactions on Knowledge and Data Engineering 24 (5), 896–911.

Adomavicius, G. & Tuzhilin, A. 2015. Context-aware recommender systems. In Recommender systems handbook. Springer, 191–226.

Amatriain, X., Jaimes, A., Oliver, N. & Pujol, J. 2011. Data mining methods for recommender systems. In Recommender Systems Handbook. Springer US, 39-71. doi:10.1007/978-0-387-85820-3_2. ⟨URL:http://dx.doi.org/10.1007/978-0-387-85820-3_2⟩.

Amatriain, X. 2016. Past, present, and future of recommender systems: An industry perspective. In Proceedings of the 21st International Conference on Intelligent User Interfaces. ACM, 1–1.

Blei, D. M., Ng, A. Y. & Jordan, M. I. 2003. Latent dirichlet allocation. Journal of machine Learning research 3 (Jan), 993–1022.

Boyle, R. 2000. The Three Princes of Serendip. ⟨URL:http://livingheritage.org/three_princes.htm⟩. ([Online; accessed 06-May-2018]).

Brin, S. & Page, L. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. Computer Networks 56 (18), 3825 - 3833. doi:10.1016/j.comnet.2012.10.007.

Cantador, I., Brusilovsky, P. & Kuflik, T. 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In Proceedings of the 5th ACM conference on Recommender systems. New York, NY, USA: ACM. RecSys 2011.

Cantador, I., Fernández-Tobías, I., Berkovsky, S. & Cremonesi, P. 2015. Cross-domain recommender systems. In Recommender Systems Handbook. Springer, 919–959.

Castells, P., Hurley, N. J. & Vargas, S. 2015. Novelty and diversity in recommender systems. In Recommender Systems Handbook. Springer, 881–918.

Celma Herrada, Ò. 2009. Music recommendation and discovery in the long tail. Universitat Pompeu Fabra. Ph. D. Thesis.

Cohn, R. & Russell, J. 2015. The Three Princes of Serendip.

Cremonesi, P., Koren, Y. & Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of RecSys'10. New York, NY, USA: ACM, 39–46. doi:10.1145/1864708.1864721.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. et al. 2010. The youtube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems. ACM, 293–296.

De Pessemier, T., Dooms, S. & Martens, L. 2014. Comparison of group recommendation algorithms. Multimedia tools and applications 72 (3), 2497–2541.

Efthymiou, V., Zervoudakis, P., Stefanidis, K. & Plexousakis, D. 2017. Group recommendations in mapreduce. In Strategic Innovative Marketing. Springer, 495–500.

Ekstrand, M. D., Ludwig, M., Konstan, J. A. & Riedl, J. T. 2011a. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In Proceedings of the 5th ACM Conference on Recommender Systems. New York, NY, USA: ACM, 133–140.

Ekstrand, M. D., Riedl, J. T. & Konstan, J. A. 2011b. Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction 4 (2), 81–173.

Farseev, A., Nie, L., Akbari, M. & Chua, T.-S. 2015. Harvesting multiple sources for user profile learning: a big data study. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval. ACM, 235–242.

Fernández-Tobías, I., Cantador, I., Kaminskas, M. & Ricci, F. 2012. Cross-domain recommender systems: A survey of the state of the art. In Spanish Conference on Information Retrieval.

Francis, J. W. P. M. E. & Booth, R. J. 1993. Linguistic Inquiry and Word Count. Technical Report. Technical Report, Dallas, TX: Southern Methodist University.

Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C. & Huber, A. 2014. Offline and online evaluation of news recommender systems at swissinfo.ch. In Proceedings of RecSys'14. New York, NY, USA: ACM, 169–176. doi:10.1145/2645710.2645745.

Ge, M., Delgado-Battenfeld, C. & Jannach, D. 2010. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In Proceedings of the Fourth ACM Conference on Recommender Systems. ACM, 257–260.

de Gemmis, M., Lops, P., Semeraro, G. & Musto, C. 2015. An investigation on the serendipity problem in recommender systems. Information Processing & Management 51 (5), 695 - 717.

Goldberg, D., Nichols, D., Oki, B. M. & Terry, D. 1992. Using collaborative filtering to weave an information tapestry. Commun. ACM 35 (12), 61–70. doi:10.1145/138859.138867.

Gunawardana, A. & Shani, G. 2015. Evaluating recommender systems. In Recommender Systems Handbook. Springer, 265–308.

Harper, F. M. & Konstan, J. A. 2015. The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems 5 (4), 19:1–19:19.

Herlocker, J. L., Konstan, J. A., Borchers, A. & Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In Proceedings of SIGIR'99. New York, NY, USA: ACM, 230–237. doi:10.1145/312624.312682.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22 (1), 5–53.

Hijikata, Y., Shimizu, T. & Nishida, S. 2009. Discovery-oriented collaborative filtering for improving user satisfaction. In Proceedings of the 14th International Conference on Intelligent User Interfaces. ACM, 67–76.

Iaquinta, L., Semeraro, G., de Gemmis, M., Lops, P. & Molino, P. 2010. Can a recommender system induce serendipitous encounters? InTech.

Järvelin, K. & Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. 20 (4), 422–446. doi:10.1145/582415.582418.

Kaminskas, M. & Bridge, D. 2014. 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).

Kaminskas, M. & Bridge, D. 2016. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Transactions on. Interactive Intelligent Systems 7 (1), 2:1–2:42.

Kapoor, K., Kumar, V., Terveen, L., Konstan, J. A. & Schrater, P. 2015. "i like to explore sometimes": Adapting to dynamic user novelty preferences. In Proceedings of the 9th ACM Conference on Recommender Systems. ACM, 19–26.

Karpus, A., Vagliano, I. & Goczyła, K. 2017. Serendipitous recommendations through ontology-based contextual pre-filtering. In International Conference: Beyond Databases, Architectures and Structures. Springer, 246–259.

Koren, Y. & Bell, R. 2011. Advances in collaborative filtering. In Recommender Systems Handbook. Springer US, 145-186. doi:10.1007/978-0-387-85820-3_5.

Kotkov, D., Konstan, J. A., Zhao, Q. & Veijalainen, J. 2018. Investigating serendipity in recommender systems based on real user feedback. In Proceedings of SAC 2018: Symposium on Applied Computing. ACM.

Kotkov, D., Veijalainen, J. & Wang, S. 2016. Challenges of serendipity in recommender systems. In Proceedings of the 12th International conference on web information systems and technologies., Vol. 2. SCITEPRESS, 251-256.

Kotkov, D., Veijalainen, J. & Wang, S. 2017. A serendipity-oriented greedy algorithm for recommendations. In Proceedings of the 13th International Conference on Web Information Systems and Technologies, Vol. 1. SCITEPRESS, 32–40.

Kotkov, D., Veijalainen, J. & Wang, S. 2018. A serendipity-oriented greedy algorithm and a complete serendipity metric for recommendation purposes. In Computing (Submitted for publication). Springer International Publishing.

Kotkov, D., Wang, S. & Veijalainen, J. 2016. A survey of serendipity in recommender systems. Knowledge-Based Systems 111, 180–192.

Lathia, N., Hailes, S., Capra, L. & Amatriain, X. 2010. Temporal diversity in recommender systems. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, 210–217.

Leino, J. 2014. User Factors in Recommender Systems: Case Studies in e-Commerce, News Recommending, and e-Learning. Tampereen yliopisto. Ph. D. Thesis.

Levy, Y. & Ellis, T. J. 2006. A systems approach to conduct an effective literature review in support of information systems research. Informing Science: International Journal of an Emerging Transdiscipline 9 (1), 181–212.

Liu, T.-Y. 2009. Learning to rank for information retrieval. Found. Trends Inf. Retr. 3 (3), 225–331. doi:10.1561/1500000016.

Lu, Q., Chen, T., Zhang, W., Yang, D. & Yu, Y. 2012. 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. IEEE Computer Society, 258–265.

Martin, D. 2008. Most untranslatable word. ⟨URL:http://www.todaytranslations.com/blog/most-untranslatable-word/⟩. ([Online; accessed 06-May-2018]).

Masthoff, J. 2015. Group recommender systems: aggregation, satisfaction and group attributes. In recommender systems handbook. Springer, 743–776.

McNee, S. M., Riedl, J. & Konstan, J. A. 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In CHI '06 Extended Abstracts on Human Factors in Computing Systems. ACM, 1097–1101.

McSherry, F. & Najork, M. 2008. Computing information retrieval performance measures efficiently in the presence of tied scores. In Advances in Information

Retrieval, Vol. 4956. Springer Berlin Heidelberg. Lecture Notes in Computer Science, 414-421. doi:10.1007/978-3-540-78646-7_38.

Murakami, T., Mori, K. & Orihara, R. 2008. Metrics for evaluating the serendipity of recommendation lists. In Annual Conference of the Japanese Society for Artificial Intelligence.

Narducci, F., Musto, C., Semeraro, G., Lops, P. & De Gemmis, M. 2013. Leveraging encyclopedic knowledge for transparent and serendipitous user profiles. In International Conference on User Modeling, Adaptation, and Personalization. Springer, 350–352.

Netflix 2009. Netflix Prize: Review Rules. http://www.netflixprize.com/rules.html. ([Online; accessed 06-May-2018]).

Nguyen, T. T., Harper, F. M., Terveen, L. & Konstan, J. A. 2017. User personality and user satisfaction with recommender systems. Information Systems Frontiers, 1–17.

Ning, X., Desrosiers, C. & Karypis, G. 2015. A comprehensive survey of neighborhood-based recommendation methods. In Recommender systems handbook. Springer, 37–76.

Peffers, K., Tuunanen, T., Rothenberger, M. A. & Chatterjee, S. 2007. A design science research methodology for information systems research. Journal of management information systems 24 (3), 45–77.

Remer, T. G. 1965. Serendipity and the three princes: From the Peregrinaggio of 1557.

Resnick, P. & Varian, H. R. 1997. Recommender systems. Communications of the ACM 40 (3), 56–58.

Ricci, F., Rokach, L. & Shapira, B. 2015. Recommender systems: introduction and challenges. In Recommender systems handbook. Springer, 1–34.

Rich, E. 1979. User modeling via stereotypes*. Cognitive Science 3 (4), 329–354. doi:10.1207/s15516709cog0304_3.

Said, A., Fields, B., Jain, B. J. & Albayrak, S. 2013. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In Proceedings of the 2013 Conference on Computer Supported Cooperative Work. ACM, 1399–1408.

Schafer, J. B., Konstan, J. & Riedl, J. 1999. Recommender systems in e-commerce. In Proceedings of the 1st ACM Conference on Electronic Commerce. New York, NY, USA: ACM. EC '99, 158–166. doi:10.1145/336992.337035.

Shani, G. & Gunawardana, A. 2011. Evaluating recommendation systems. In Recommender Systems Handbook. Springer US, 257-297. doi:10.1007/978-0-387-85820-3_8. ⟨URL:http://dx.doi.org/10.1007/978-0-387-85820-3_8⟩.

Silveira, T., Zhang, M., Lin, X., Liu, Y. & Ma, S. 2017. How good your recommender system is? a survey on evaluations in recommendation. International Journal of Machine Learning and Cybernetics, 1–19.

Smith, B. & Linden, G. 2017. Two decades of recommender systems at amazon. com. IEEE Internet Computing 21 (3), 12–18.

Smyth, B., Coyle, M. & Briggs, P. 2011. Communities, collaboration, and recommender systems in personalized web search. In Recommender Systems Handbook. Springer US, 579-614. doi:10.1007/978-0-387-85820-3_18.

Smyth, B. & McClave, P. 2001. Similarity vs. diversity. In International Conference on Case-Based Reasoning. Springer, 347–361.

Su, X. & Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. Adv. in Artif. Intell. 2009, 4:2–4:2. doi:10.1155/2009/421425.

Tacchini, E. 2012. Serendipitous mentorship in music recommender systems. Ph. D. Thesis.

Taramigkou, M., Bothos, E., Christidis, K., Apostolou, D. & Mentzas, G. 2013. Escape the bubble: Guided exploration of music preferences for serendipity and novelty. In Proceedings of the 7th ACM Conference on Recommender Systems. ACM, 335–338.

Vargas, S. & Castells, P. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In Proceedings of RecSys'11. New York, NY, USA: ACM, 109–116. doi:10.1145/2043932.2043955.

Vig, J., Sen, S. & Riedl, J. 2012. The tag genome: Encoding community knowledge to support novel interaction. ACM Transactions on Interactive Intelligent Systems 2 (3), 13:1–13:44.

Von Alan, R. H., March, S. T., Park, J. & Ram, S. 2004. Design science in information systems research. MIS quarterly 28, 75–105.

Zhang, Y. C., Séaghdha, D. O., Quercia, D. & Jambor, T. 2012. Auralist: Introducing serendipity into music recommendation. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. ACM, 13–22.

Zheng, Q., Chan, C.-K. & Ip, H. H. 2015. An unexpectedness-augmented utility model for making serendipitous recommendation. In Advances in Data Mining: Applications and Theoretical Aspects, Vol. 9165. Springer International Publishing, 216-230.

Ziegler, C.-N., McNee, S. M., Konstan, J. A. & Lausen, G. 2005. Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on World Wide Web. New York, NY, USA: ACM, 22–32.

## ORIGINAL PAPERS

## PI

## A SURVEY OF SERENDIPITY IN RECOMMENDER SYSTEMS

by

Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen 2016

Knowledge-Based Systems.

Knowledge-Based Systems 111 (2016) 180–192

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# A survey of serendipity in recommender systems

Denis Kotkov*, Shuaiqiang Wang, Jari Veijalainen

*University of Jyvaskyla, Department of Computer Science and Information Systems, P.O.Box 35, FI-40014 University of Jyvaskyla, Finland*

A R T I C L E I N F O

*Article history:*
Received 5 December 2015
Revised 9 August 2016
Accepted 10 August 2016
Available online 11 August 2016

*Keywords:*
Novelty
Serendipity
Recommender systems
Evaluation metrics
Evaluation strategies
Algorithms

A B S T R A C T

Recommender systems use past behaviors of users to suggest items. Most tend to offer items similar to the items that a target user has indicated as interesting. As a result, users become bored with obvious suggestions that they might have already discovered. To improve user satisfaction, recommender systems should offer serendipitous suggestions: items not only relevant and novel to the target user, but also significantly different from the items that the user has rated. However, the concept of serendipity is very subjective and serendipitous encounters are very rare in real-world scenarios, which makes serendipitous recommendations extremely difficult to study. To date, various definitions and evaluation metrics to measure serendipity have been proposed, and there is no wide consensus on which definition and evaluation metric to use. In this paper, we summarize most important approaches to serendipity in recommender systems, compare different definitions and formalizations of the concept, discuss serendipity-oriented recommendation algorithms and evaluation strategies to assess the algorithms, and provide future research directions based on the reviewed literature.

## 1. Introduction

Many online stores offer thousands of different products, such as movies, songs, books and various services. With a wide range of goods, a store is likely to have a product that fits user preferences. However, the growth of choice does not always lead to user satisfaction, as sometimes it is difficult to find a product a user would like to purchase due the overwhelming number of available products [1].

To overcome the flood of information, online stores have widely adopted recommender systems [2]. In this paper, the term *recommender system* (RS) refers to a software tool that suggests items interesting to users [1]. 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 an RS suggests to the user in an interaction through the Web, email or text message” [3]. For example, an item can refer to a movie, a song or even a friend in an online social network. A *user* is generally understood to mean a person who uses an RS for any purpose. For example, a user might be an individual looking for a movie to watch or a company representative looking for goods to purchase.

Recommendation algorithms can be divided into categories: non-personalized and personalized [4]. Non-personalized algorithms suggest the same items to each user. For example, recommendations based on the popularity of items can be considered non-personalized. Personalized algorithms suggest different items depending on the target user. In this case, two users might receive different recommendations. In this paper, we focus on personalized RSs.

As a rule, personalized RSs provide suggestions based on user profile. A user profile might include any information about a user, such as an ID, age, gender and actions the user has performed with items in the past. In this paper, the term *user profile* refers to a set of items rated by the user [3]. Generation of recommendations can be based on (1) ratings a target user and others have given to items, (2) attributes of items rated by the target user or (3) both ratings and attributes [5]. One RS might suggest items that could be interesting to users who like many items in common with the target user. Another might recommend items that have many common attributes with items liked by the target user. For example, if a user rates many comedies, an RS will suggest more comedies.

Currently, RSs are widely adopted for different purposes. The ultimate goal of services that host RSs is to increase turnover [1]. Depending on the application scenarios, ways to achieve the goal might differ. One RS might suggest expensive items over interesting ones for a particular user, as the profit of the hosting service depends on user purchases. The RS of a service that sells subscriptions might suggest items interesting to users to encourage users to visit the service again. The business goal of the RS thus may differ from the goals of the users of the service [6].

* Corresponding author.
*E-mail addresses:* deigkotk@student.jyu.fi (D. Kotkov), shuaiqiang.wang@jyu.fi (S. Wang), jari.veijalainen@jyu.fi (J. Veijalainen).

http://dx.doi.org/10.1016/j.knosys.2016.08.014
0950-7051/© 2016 Elsevier B.V. All rights reserved.

### 1.1. Motivation

Users may interact with an RS for different purposes, such as expressing themselves, influencing others or just browsing the catalog of items. However, the main reason the majority of individuals would use an RS is to discover novel and interesting items. It is demanding to look for items manually among an overwhelming number of options [3,6,7].

Most recommendation algorithms are evaluated based on accuracy, which does not correspond to user needs, as high accuracy indicates that the algorithm has prediction power but neglects the novelty and unexpectedness of suggested items [3,8–10]. To achieve high accuracy, RSs tend to suggest items similar to a user profile [3,11]. Consequently, the user receives recommendations only of items in some sense similar to items the user rated initially when she/he started to use the service (the so-called overspecialization problem). This has been observed to lead to a low user satisfaction [3,6,11,12].

One of the ways to overcome the overspecialization problem is to increase the serendipity of an RS. At this point we tentatively consider an RS to be serendipitous if it is able to suggest novel, interesting and unexpected items to a particular user. We will elaborate various definitions in Section 2.2. Serendipity-oriented algorithms can be used to slightly expand user tastes [13] or they can be applied as an additional "surprise me" option of an RS [14]. It is challenging to suggest serendipitous items for the following reasons [3]:

- There is no consensus on the definition of serendipity in RSs, as many papers present different definitions and formalizations of the concept [3,15].
- It is not clear how to measure serendipity, as there is no agreement on the definition of the concept [12].
- Serendipity includes an emotional dimension, which is very subjective [9,16]. Consequently, it is difficult to infer reasons why a user considers an item to be serendipitous [2].
- Serendipitous suggestions are less frequent than relevant ones, as not only must serendipitous items be relevant, but also novel and unexpected to a user [3]. As a result, it is often more difficult to capture and induce serendipitous recommendations than relevant ones in an experiment [17].

### 1.2. Methods and logistics

In this paper, we focus on serendipity, a poorly investigated concept that remarkably influences user satisfaction [11,18,19]. We aim at answering the following research questions:

RQ1 What is serendipity in recommender systems? What makes certain items serendipitous for particular users?
RQ2 What are the state-of-the-art recommendation approaches that suggest serendipitous items?
RQ3 How can we assess serendipity in recommender systems?
RQ4 What are the future directions of serendipity in RSs?

To address the research questions, we collected articles that mention serendipity using Google Scholar[1], Scopus[2] and Web of Science[3].

1. *Preliminary analysis.* By investigating articles received by entering the query "serendipity in recommender systems," we inferred that researchers employ different definitions of serendipity and different metrics to measure serendipity in RSs [19–21].
2. *Serendipity definitions and metrics.* To answer *RQ1* and *RQ3*, we looked for papers that proposed definitions of serendipity in RSs and methods to measure it. We selected the first 20 articles retrieved by the search engines in response to the search queries "definition of serendipity in recommender systems," "serendipity in recommender systems," "measure surprise in recommender systems" and "unexpectedness in recommender systems." To find papers that propose definitions of the concept and methods to measure it, we paid attention to related works and evaluation metrics used in the articles (forward search [22]). Eventually, we found 18 qualifying articles, six of which presented serendipity evaluation metrics (more detail is provided in Section 5).
3. *Serendipity-oriented algorithms.* To answer *RQ2*, we looked for algorithm proposals that improve the serendipity metrics discovered, searching papers that cite articles with proposals of serendipity metrics (backward search [22]).
4. *Filter.* Many of the articles mention serendipity in RSs, but mainly focus on other topics. We therefore primarily selected relevant high-quality articles.

The goal of the paper is to give a definition of serendipity and an overview of existing recommendation approaches and evaluation metrics, and thus to guide and inspire future efforts on recommendation serendipity. The main contributions of this paper are summarized as follows:

- Review of definitions and formalizations of serendipity and related concepts, such as relevance, novelty and unexpectedness.
- Review and classification of serendipity-oriented recommendation algorithms.
- Review of evaluation strategies to assess serendipity in RSs.
- Future directions to improve serendipity in RSs.

This paper is organized as follows. We answer *RQ1* in Section 2, which is dedicated to the definition of serendipity in RSs. In Section 3, we review serendipity-oriented recommendation algorithms and answer *RQ2*. Sections 4 and 5 describe strategies to assess serendipity and evaluation metrics (*RQ3*), respectively. Finally, we provide future directions (*RQ4*) and conclude in Sections 6 and 7, respectively.

## 2. Concepts

The degree of uncertainty around the terminology related to serendipity requires us to set up vocabulary to be used throughout the paper. In this section we answer *RQ1* by defining serendipity in RSs and describing what kinds of items are serendipitous for particular users.

### 2.1. Components of serendipity

Serendipity is a complex concept which includes other concepts. In this section, we provide those related concepts, such as relevance, novelty and unexpectedness.

#### 2.1.1. Relevance

Depending on a particular application scenario, we may consider different actions that a user performs with items to indicate his/her interest. For example, we might regard an item as relevant for a user if she/he has gaven it a high rating and/or if she/he has purchased it. Therefore, to combine different terms that mean the same in the context of this paper, we define a *relevant item* as an item that a user likes, consumes or is interested in, depending on the application scenario of a particular RS.

[1] https://scholar.google.com/.
[2] http://www.scopus.com/.
[3] https://webofknowledge.com/.

#### 2.1.2. Novelty

The term *novel item* has different meanings that can be summarized as follows [3]:

1. *Item novel to an RS.* A recently added item that users have not yet rated [23] (so-called cold start problem [24]).
2. *Forgotten item.* A user might forget that she/he consumed the item some time ago [23].
3. *Unknown item.* A user has never consumed the item in his/her life [23].
4. *Unrated item.* An unrated by the target user regardless of whether she/he is familiar with the item [7].

In this paper, the term *novel item* corresponds to definition 3 due to the popularity of the definition in studies on serendipity [9,12,25,26]. A novel item is a relevant or irrelevant item that a target user has never seen or heard about in his/her life [23]. However, a user does not normally rate all the items with which she/he is familiar.

It is often necessary to estimate how probable it is that a user has already seen an item. A user is more likely to be familiar with items similar to what she/he consumes and popular among other users [6,27]. In this paper, popularity indicates how likely users are to be familiar with certain items. Popularity might correspond to a number of ratings assigned to an item in a dataset used by an RS. Popular items are widely recognized by an individual, since she/he might have heard about the items from others or mass media, such as TV, radio or popular web-sites. A user might also be aware of items that are similar to what she/he usually consumes. For example, a jazz lover is likely to attend a small jazz concert rather than a rock concert. By attending the jazz concert she/he is likely to discover some unpopular jazz music. Novelty therefore includes two subcomponents: *unpopularity* and *dissimilarity to a user profile* [13]. For example, a jazz lover is likely to consider a very unpopular jazz song and relatively popular classical song as novel. In contrast, the user might have already listened to a classical hit on the radio as well as a relatively popular jazz song in a jazz club.

#### 2.1.3. Unexpectedness

*Unexpectedness* and *surprise* are terms frequently used in the literature [10,20,28,29], but to the best of our knowledge there is no consensus on the terms' definitions. In this paper, the terms *unexpected* and *surprising* refer to items that *significantly differ* from the profile of the user regardless of how novel or relevant those items are. For example, a classical song recommended to a jazz lover is likely to be more unexpected than a jazz song.

There are two differences between novelty and unexpectedness. First, a user may find an item unexpected even if she/he is familiar with the item. Second, to surprise a user, unexpected items have to differ from the user profile more than novel items.

### 2.2. Concept of serendipity

Serendipity is a difficult concept to study [3], as it includes an emotional dimension [9,16]. It is challenging to define serendipity in recommender systems as well as what kind of items are serendipitous and why [3,15], since generally serendipitous encounters are very rare [17].

The term *serendipity* has been recognized as one of the most untranslatable words [30]. The first known use of the term was found in a letter by Horace Walpole to Sir Horace Mann on January 28, 1754. The author described his discovery by referencing a Persian fairy tale, "The Three Princes of Serendip". The story described a journey taken by three princes of the country Serendip to explore the world [31]. In the letter, Horace Walpole indicated that the princes were "always making discoveries, by accidents & sagacity, of things which they were not in quest of" [32].

According to the dictionary [33], serendipity is "the faculty of making fortunate discoveries by accident." The word "discovery" indicates the novelty of serendipitous encounters, while the word "fortunate" indicates that the discovery must be relevant and unexpected.

#### 2.2.1. Serendipity in a computational context

As serendipity is a complicated concept, Van Andel discussed whether a computer can generate serendipitous information [34]:

> *[P]ure serendipity is not amenable to generation by a computer. The very moment I can plan or programme 'serendipity' it cannot be called serendipity anymore. All I can programme is, that, if the unforeseen happens, the system alerts the user and incites him to observe and act by himself by trying to make a correct abduction of the surprising fact or relation.*

The author suggests that computer programs cannot generate but can help a user find serendipitous information. Based on [34], we suppose that as recommender systems are designed to "assist and augment" a process of making "choices without sufficient personal experience of the alternatives" [35], they can also assist a process of making "fortunate discoveries" [33].

Corneli et al. investigated serendipity in a computational context and proposed a framework to describe the concept [36]. The authors considered *focus shift* a key condition for serendipity, which happens when something initially perceived irrelevant, neutral or even negative becomes relevant. According to the authors, the concept of serendipity includes four components:

- *Prepared mind* (focus on something an explorer is looking for). The component indicates that an explorer should be focused on what she/he wants to investigate.
- *Serendipity trigger* (inspiration for a novel result). The act of drawing initial attention to a serendipitous object or phenomenon.
- *Bridge* (inference). Investigation of a discovered object or phenomenon.
- *Result* (an outcome). This can be a new artifact, process, phenomenon, hypothesis or problem.

The framework can be illustrated using the example of the microwave oven invention. Once Percy Spencer was working with a radar set and noticed that a candy bar in his pocket had melted. Later, the researcher investigated this phenomenon and invented the first microwave oven. In the example, a *focus shift* happened when the once uninteresting melted candy bar attracted the researcher's attention. Percy Spencer had a *prepared mind*, as he was initially working with radar. The researcher experienced a *serendipity trigger* when he noticed that the candy bar had melted. A *bridge* occurred when he investigated why it had melted. Finally, the *result* is the invention of the microwave oven.

#### 2.2.2. Serendipity in the context of recommender systems

The framework proposed by Corneli et al. can also apply to RSs. In this case, a *focus shift* happens when initially uninteresting items become interesting. A user has a *prepared mind*, as she/he expects that an RS will suggest items similar to his/her profile [10]. The attention of the user drawn by the items represents a *serendipity trigger*. The investigation of the items by the user is a *bridge*. Finally, the *result* for the user is the obtained knowledge.

According to the framework, an item serendipitous for a user should meet certain requirements. First, the serendipitous item is required to be unexpected and relevant to create a *focus shift*. A user is likely to consider an item irrelevant at first sight if the item is significantly different from his/her profile (unexpected). Meanwhile, the item is required to be relevant to eventually become interesting for the user. To create a *serendipity trigger*, the item must

**Table 1**
Definitions of serendipity in related works.

| Articles | Definition of serendipity | Components |
|---|---|---|
| McNee et al. [29] | "serendipity in a recommender is the experience of receiving an unexpected and fortuitous item recommendation. But even if we remove that component, the unexpectedness part of this concept the novelty of the received recommendations is still difficult to measure" | Relevance, novelty and unexpectedness |
| Adamopoulos and Tuzhilin [10,28] | "serendipity, the most closely related concept to unexpectedness, involves a positive emotional response of the user about a previously unknown (novel) [...] serendipitous recommendations are by definition also novel." | Relevance, novelty and unexpectedness |
| Sridharan [37] | "serendipity is defined as the accident of finding something good or useful while not specifically searching for it. Serendipity is thus closely related to unexpectedness and involves a positive emotional response of the user about a previously unknown item. It measures how surprising the unexpected recommendations are [...] serendipity is concerned with the novelty of recommendations and in how far recommendations may positively surprise users" | Relevance, novelty and unexpectedness |
| Zhang et al. [19] | "serendipity represents the "unusualness" or "surprise" of recommendations" | Novelty and unexpectedness |
| Maksai et al. [39] | "Serendipity is the quality of being both unexpected and useful." | Relevance and unexpectedness |

be novel for the user; otherwise the user would not pay attention to the item. As a result, to be serendipitous for a user, items must be relevant, novel and unexpected.

Many authors indicate that novelty, relevance and unexpectedness are important for serendipity, yet these authors often do not define these concepts, which might cause confusion [8,9,21,28,29,37]. The majority of papers include each of these three components to the definition of serendipity [9,28,29,37]. One example is the definition proposed by Iaquinta et al. [26]:

> *A serendipitous recommendation helps the user to find a surprisingly interesting item that she might not have otherwise discovered (or it would have been really hard to discover). [...] Serendipity cannot happen if the user already knows what is recommended to her, because a serendipitous happening is by definition something new. Thus the lower is the probability that user knows an item, the higher is the probability that a specific item could result in a serendipitous recommendation.*

Iaquinta et al. suggest that serendipitous items are always relevant, novel and unexpected to a user [26]. In contrast, Kaminskas and Bridge consider even items a user is aware of to be serendipitous [21]. Their definition includes only relevance and unexpectedness [21]:

> *Serendipity consists of two components - surprise and relevance [...] we also do not require a surprising item to be novel, but only different from the user's expectations, which are represented by a set of items.*

Another definition was proposed by Chiu et al. [38]. The authors proposed completely different definitions adopted towards online social networks. According to Chiu et al. [38], a serendipitous item is novel for a target user, but familiar to the user's friends, defined as contacts in an online social network: "Items are not yet accessed (implicit) or rated (explicit). But the user's friends (in the social network) have accessed before" [38]. Table 1 provides additional summarized examples of definitions of serendipity.

To answer *RQ1*, in this paper, we refer to serendipity as a property of an RS that indicates how good the RS is at suggesting serendipitous items [7], where *serendipitous items* are relevant, novel and unexpected for a user [3,10,12,26,29,37].

In terms of popularity and similarity to a user profile, serendipitous items are limited by novelty and unexpectedness. First, as serendipity includes novelty, serendipitous items should be unpopular [40] and dissimilar to a user profile [19], otherwise the user is likely to be familiar with them (Section 2.1.2). Second, the unexpectedness component requires serendipitous items to significantly differ from a user profile (Section 2.1.3).

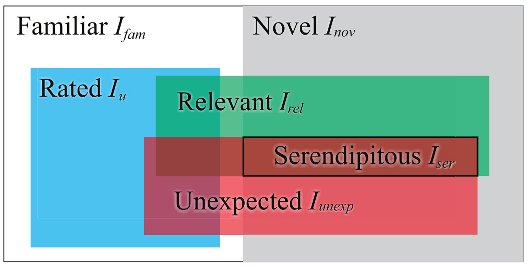


**Fig. 1.** Euler diagram of items from a user's point of view at a given moment of time (snapshot).

Fig. 1 illustrates set intersections of familiar, novel, rated, relevant, unexpected and serendipitous items perceived by a user at a particular moment of time. Suppose that an RS contains $I$ items and a user is familiar with items $I_{fam} \subseteq I$. The items that the user has never heard of are novel $I_{nov} = I \setminus I_{fam}$. The RS has information that the user is aware of items she/he rated $I_u \subseteq I_{fam}$. We assume that the user rates only items with which she/he is familiar. Items interesting to the user are relevant $I_{rel} \subseteq I$. They are distributed among the categories familiar, rated, novel and unexpected. Unexpected items $I_{unexp} \subseteq I$ differ from items the user rated and can be novel or familiar to the user. In rare cases, unexpected items can also be rated. Rated items are likely to be unexpected if they differ from the rest of the rated items. Serendipitous items are relevant, novel and unexpected $I_{ser} = I_{rel} \cap I_{nov} \cap I_{unexp}$. The task of the serendipity-oriented RS is to suggest to the user serendipitous items $I_{ser}$ based on rated items $I_u$.

The proposed restrictions of serendipitous items can have exceptions. For example, a user might consider a popular item novel and serendipitous. We apply these restrictions for simplification and assume they are reasonable in most cases.

In general, the increase of serendipity decreases accuracy for two reasons. First, as users tend to like items similar to items they already find interesting, many items different from the users' profiles are irrelevant. Second, considering the number of low-quality items among unpopular ones [6], increasing serendipity is likely to increase the number of irrelevant items recommended. An algorithm has to offer risky suggestions to increase serendipity.

Depending on user needs, requirements for serendipity-oriented algorithms might differ. A user might want to discover relevant items. In this case, it is important to keep a trade-off between accuracy and serendipity [19,21,41]. However, a user might look for serendipitous items in particular. For example, a "surprise me" algorithm could be offered to the user as an additional option [14]. In this case, it might be more important to increase serendipity regardless of accuracy.

#### 2.2.3. Difference between serendipity and similar concepts

It is important to clarify the difference between novelty and serendipity. First, serendipitous items are relevant, while novel items can be irrelevant. Second, as serendipitous items are unexpected, they significantly differ from items a user rated in an RS, which is not necessary for novel items. For example, let us assume a jazz lover logs into an RS to discover a few songs. A suggestion of a jazz song that the user is unaware of, regardless of whether she/he likes it, is likely to be novel, but not serendipitous.

Another concept which might be confused with serendipity is diversity. The concept reflects how different items are from each other [7,27]. Diversity is a desirable property in an RS, as users often become bored with recommendation lists containing items very similar to each other. For example, a poker player might enjoy a recommendation list of books about poker, combinatorics and math more than a list containing only poker books.

There are two key differences between diversity and serendipity. First, diversity is based on the comparison of recommended items between each other, while serendipity is based on the comparison of recommended items and a user profile [27]. Second, diversity does not include relevance, a necessary component for serendipity.

#### 2.2.4. Conclusions and discussion

In this section, we answered *RQ1* based on the reviewed literature related to serendipity in RSs. Serendipity is a property that indicates how good an RS is at suggesting serendipitous items that are relevant, novel and unexpected for a particular user. Serendipitous items are by definition unpopular and significantly different from the user profile.

It is worth mentioning that serendipity is not always a necessary component in suggesting items. Some services might need an RS to suggest only relevant items, regardless of how serendipitous those items are. For example, an RS that suggests friends might be required to offer only those individuals with whom a target user is familiar. Furthermore, users might prefer obvious suggestions over serendipitous ones [42]. For example, user intention to receive risky recommendations might increase with the growth of experience of using the system [6]. However, in many cases, serendipity is perceived as a purely positive property of an RS [28,29,41].

## 3. Approaches

In the section, we are going to review serendipity-oriented algorithms and answer *RQ2*. Recommendation algorithms can be divided into the following categories [43].

- *Content-based filtering* (CBF) utilizes items' attributes to generate recommendations [43]. This kind of algorithm considers choices a user has made in the past to suggest items the user will consume in the future. For example, if a user has watched a comedy movie, then a CBF algorithm would suggest another comedy movie to the user.
- *Collaborative filtering* (CF) considers only ratings users have gaven to items [43]. For example, if two users have rated many common items similarly, then a CF algorithm would probably recommend items highly rated by the first user that the second user has not yet rated and vice versa.
  - *Memory-based CF* typically uses similarity measures based on user ratings [5,43]. This approach generates recommendations for a target user based on what similar users have rated or based on items similar to items the target user has consumed.
  - *Model-based CF* utilizes a model that receives user ratings as input and generates recommendations [5,43].
- *Hybrid filtering* takes into account both ratings and attributes of items [43–45].

Serendipity-oriented RSs can be classified depending on the data they use (CF, CBF, hybrid) or on the architecture of recommendation algorithms. We divide serendipity-oriented approaches into three categories:

- *Reranking algorithms (Reranking).* To improve serendipity, an algorithm can leverage ratings predicted by an accuracy-oriented approach and rerank the output of an RS. An accuracy-oriented algorithm finds items relevant for a target user, while a reranking algorithm assigns low ranks to obvious suggestions.
- *Serendipity-oriented modification (Modification).* This category represents modifications of accuracy-oriented algorithms. The main difference between modification and reranking is that reranking algorithms can use any accuracy-oriented algorithms that assign ranking scores to items, while modifications can only be applied to certain kinds of algorithms. An example is the k-furthest neighbor approach [18], which is a modification of the k-nearest neighbor algorithm (user-based CF). Instead of suggesting items liked by users similar to a target user, a k-furthest neighbor algorithm recommends items disliked by users dissimilar to a target user.
- *Novel algorithms (New).* This category includes serendipity-oriented algorithms that are not based on any common accuracy-oriented algorithms. Approaches from this category are very diverse and use different techniques, such as clustering [46] and random walk with restarts [8,14]. One of the examples is the TANGENT algorithm, which detects groups of like-minded users and suggests items simultaneously liked by users from the group of the target user and other groups [14]. Recommended items are related to previous choices of the user and likely to be surprising, as these items are chosen by users from a group different than the one of the target user.

We may take actions to improve the serendipity of recommendation at different stages of the recommendation process. Similarly to [47], we suggest three paradigms for serendipity-oriented recommendation algorithms:

- Pre-filtering: A recommendation algorithm preprocesses the input data for an accuracy-oriented algorithm to improve serendipity.
- Modeling: A recommendation algorithm improves serendipity in the phase of generating recommendations.
- Post-filtering: A recommendation algorithm reranks the results of accuracy-oriented algorithms.

Table 2 demonstrates classification of the state-of-the-art serendipity-oriented approaches. The majority of approaches are either modifications or of uncertain basis. In terms of the data that algorithms use, we reviewed at least one algorithm from each category (CF, CBF and hybrid). Serendipity subcomponents include popularity (*pop*) and similarity to a user profile (*sim*), which the algorithms take into account. By assessing the parameter, we examined whether the model takes into account the subcomponents. Among the approaches presented, the algorithms proposed by Zheng et al. [13] and Zhang et al. [19] capture both popularity and similarity to a user profile.

To review serendipity-oriented algorithms, we present notation in Table 3. Let $I$ be a set of available items and $U$ be a set of users. User $u$ rates or interacts with items $I_u$, $I_u \subseteq I$. A recommender system suggests $R_u$ items to user $u$. Each item $i$, $i \in I$ is represented as

**Table 2**
Classification of serendipity-oriented algorithms (MF - matrix factorization, kNN - k-nearest neighbor, unpop - unpopularity, dissim - dissimilarity to a user profile).

| Name | Approach type | Serendipity subcomponents | Paradigm | Recommendation generation |
|---|---|---|---|---|
| Utility Model [10] | Hybrid | dissim | Post-filtering | Reranking |
| Full Auralist [19] | Model-based CF | unpop + dissim | Post-filtering | Reranking |
| K-Furthest Neighbor [18] | Memory-based CF | unpop | Pre-filtering | Modification of kNN |
| kNN with item taxonomy[48] | Hybrid | dissim | Modeling | Modification of kNN |
| Search-time-based-RS [49] | Hybrid | unpop | Modeling | Modification of kNN |
| Unexpectedness-Augmented Utility Model [13] | Model-based CF | unpop + dissim | Modeling | Modification of MF |
| Serendipitous Personalized Ranking [41] | Model-based CF | unpop | Modeling | Modification of MF |
| Distance-based Model [46] | CBF | dissim | Modeling | New |
| TANGENT [14] | Model-based CF | dissim | Modeling | New |
| RWR-KI [8] | CBF | unpop | Modeling | New |

**Table 3**
Notations.

| | |
|---|---|
| $I = (i_1, i_2, \ldots, i_n)$ | the set of items |
| $F = (f_1, f_2, \ldots, f_z)$ | feature set |
| $i = (f_{i,1}, f_{i,2}, \ldots, f_{i,z})$ | representation of item $i$ |
| $U = (u_1, u_2, \ldots, u_m)$ | the set of users |
| $I_u$, $I_u \subseteq I$ | the set of items rated by user $u$ (user profile) |
| $R_u$, $R_u \subseteq I$ | the set of items recommended to user $u$ |
| $rel_u(i)$ | 1 if item $i$ relevant for user $u$ and 0 otherwise |
| $U_i$ | the set of users who rated item $i$ |

a vector $i = (f_{i,1}, f_{i,2}, \ldots, f_{i,z})$ in a multidimensional feature space $F$. For example, a feature can be a genre of a movie on a web-site. If $F = (drama, comedy, romance)$ then the movie "Forrest Gump" can be represented as $i_{Forrest} = (0.4, 0.2, 0.4)$, where the numbers indicate how strongly the movie belongs to the genre.

## 3.1. Reranking

This section is dedicated to serendipity-oriented recommendation algorithms that rerank results of accuracy-oriented ones to increase serendipity.

### 3.1.1. Utility model

Adamopoulos and Tuzhilin proposed a serendipity-oriented algorithm that reranks the results of any accuracy-oriented algorithm that produces relevance scores [10]. The algorithm assigns overall scores to items based on two metrics: (1) unexpectedness utility and (2) quality utility. Unexpectedness utility indicates obviousness of items for a user, while quality utility reflects their relevance for the user. The algorithm consists of the following steps:

1. Computation of the set of expected items $E_u$;
2. Filtering of items that are likely to be irrelevant based on the relevance score provided by an accuracy-oriented algorithm;
3. Filter of items that are too obvious for a user based on expected items $E_u$;
4. Overall utility calculation;
5. Recommendation of items based on the overall utility.

The set of expected items $E_u$ is represented by items previously consumed by a user $I_u$ or by items similar or related to $I_u$, where relatedness is inferred by various data mining techniques. In steps 2 and 3, the algorithm uses a threshold to remove items obvious or irrelevant to a user. On the last step, the algorithm suggests items with the highest overall utility to the user. Overall utility consists of unexpectedness utility and quality utility. Unexpectedness utility depends on the average distance between an item and user expectations $E_u$, where the distance is based on user ratings (collaborative) and on item attributes (content-based). Quality utility is based on ratings provided by an accuracy-oriented algorithm.

The proposed algorithm seems to recommend more items that differ from items a target user usually consumes due to filtering based on user expectations. However, the algorithm might still recommend popular items, as it does not explicitly take popularity into account.

### 3.1.2. Full auralist

Seeking to improve diversity, novelty and serendipity while keeping acceptable accuracy, Zhang et al. proposed a recommendation algorithm called Full Auralist [19]. Full Auralist consists of three algorithms:

- *Basic auralist* is responsible for the relevance of recommendations.
- *Listener diversity* provides a diversified recommendation list.
- *Declustering* is designed to suggest items unexpected for a user.

Each algorithm returns a ranked list of items. *Full Auralist*, in turn, integrates generated ranks using their linear combination.

The algorithm appears to capture both the popularity of items and their similarity to a user profile. However, Full Auralist might be quite difficult to implement, as it contains three different algorithms.

## 3.2. Modification

This section is dedicated to serendipity-oriented recommendation algorithms based on accuracy-oriented ones.

### 3.2.1. K-furthest neighbor

Seeking to improve novelty, Said et al. has proposed k-furthest neighbor (kFN) recommendation algorithm, similar to kNN [18]. As kNN is biased towards popular items, which results in poor personalization, kFN is designed to overcome this problem. Instead of suggesting items that neighborhood users like, kFN forms neighborhoods of users dissimilar to a target user. By selecting items dissimilar users dislike, kFN is supposed to overcome the bias of items liked by the majority [18].

By overcoming the popularity bias, the algorithm seems to suggest more novel items, at least with respect to kNN. However, it is questionable whether it recommends serendipitous items, as it does not explicitly improve the dissimilarity of recommendations to a target user profile.

### 3.2.2. Knn with item taxonomy

Nakatsuji et al. designed a novelty-oriented recommendation algorithm [48]. The authors modified kNN by replacing the similarity measure with *relatedness*. By forming a neighborhood of related users and picking items from their profiles, the algorithm is supposed to improve the novelty of recommendations, over that of a classical kNN.

The algorithm is a modification of kNN, which calculates ratings based on relatedness. Relatedness is inferred by utilizing random walk with restarts (RWR) on a user similarity graph [50]. In RWR a random particle travels from node to node with a probability equivalent to an edge weight. During each step, the particle has a probability to return to its starting node. The particle visits nodes a different number of times depending on the starting node. After a sufficient number of random walks, the ratio of the number of transitions and number of visits of a certain node stabilizes. The obtained probabilities indicate relatedness between starting node and other nodes.

In a user similarity graph, nodes correspond to users, while edges correspond to similarities. User similarities are based on a taxonomy of items. The similarities are calculated considering items and classes of items that a user rated.

By picking users who are dissimilar but related to a target user, the algorithm seems to suggest more items that are dissimilar to the target user profile. Furthermore, user similarity calculation might be especially useful with a rich taxonomy of items. However, even related users might still give high ratings to popular items, which might result in suggestions with which a target user is already familiar.

### 3.2.3. Search-time-based-RS

Kawamae proposed a recommendation algorithm based on estimated search time [49,51]. The author employed the following definition of serendipity: “a serendipitous recommendation helps the user find a surprisingly interesting item he might not have otherwise discovered” [49]. In other words, the more difficult it is to find an item, the more probable that the item is serendipitous to a user.

The algorithm consists of three steps [49]. First, for each target user the algorithm detects similar users (innovators) who have common tastes and who discover recently released items better than the target user. Second, the algorithm measures how likely it is that a target user will consume a particular item from a profile of an innovator. Third, the algorithm combines these two probability parameters into ranking score and sorts a suggestion list.

The advantage of the algorithm is that it takes into account when items were released and consumed by users. In terms of similarity to a user profile and popularity, the algorithm seems to suggest unpopular items, as personal innovator probability gives high scores to recently released items that have not yet become popular. However, as the algorithm searches for users who have rated many common items with the target user, it is likely that a recommended item would be similar to a target user profile.

### 3.2.4. Unexpectedness-augmented utility model

Zheng et al. proposed a recommendation algorithm [13] based on PureSVD (variation of singular value decomposition (SVD)) [52]. The main difference between PureSVD and the proposed algorithm is the optimization task:

$$min \sum_{u \in U} \sum_{i \in I} (\widetilde{r}_{u,i} - p_u \cdot q_i^T)^2 w_{u,i} + \lambda(|p_u|^2 + |q_i|^2) \tag{1}$$

$$w_{u,i} = \left(1 - \frac{|U_i|}{|U|}\right) + \frac{\sum_{j \in I_u} dist(i,j)}{|I_u|}, \tag{2}$$

where $p_u$ and $q_i$ are user and item feature vectors, respectively, while $\widetilde{r}_{u,i}$ corresponds to the ratings user $u$ gave to item $i$ with the absence of a rating represented by zero. The set of users who rated item $i$ is represented by $U_i$. The weight $w_{u,i}$ increases the contribution of unexpected items, which forces a gradient descent algorithm to pay additional attention to the ratings of those items [13].

The proposed algorithm captures both popularity and similarity to a user profile. Consequently, it suggests more novel and serendipitous items than PureSVD.

### 3.2.5. Serendipitous personalized ranking

Lu et al. suggested a serendipitous personalized ranking algorithm (SPR), based on SVD. The objective of SPR is to maximize the serendipitous area under the ROC (receiver operating characteristic) curve (SAUC). SAUC is based on AUC (area under the curve) and defined as follows [41]:

$$SAUC(u) = \frac{1}{|I_u^+||I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in (I \setminus I_u^+)} \sigma(\hat{r}_{u,i} - \hat{r}_{u,j})(pop(j))^{\alpha} \tag{3}$$

where $\sigma$ is a 0–1 loss function: $\sigma(x) = 1$ if $x > 0$, $\sigma(x) = 0$ otherwise, while $\hat{r}_{u,i}$ is a preference prediction of user $u$ for item $i$. Relevant items are represented by $I_u^+$, consequently, $I \setminus I_u^+$ corresponds to unrated and irrelevant items. Popularity weight corresponds to $pop(i)$, $pop(i) \propto |U_i|$.

The proposed algorithm appears to have high novelty due to the popularity parameter. However, the algorithm might not suggest serendipitous items, as it disregards dissimilarity to a user profile.

## 3.3. New

This section is dedicated to serendipity-oriented recommendation algorithms that are not based on any common accuracy-oriented algorithms.

### 3.3.1. Distance-based model

Akiyama et al. proposed a content-based algorithm to improve the serendipity of an RS. First, the recommendation algorithm clusters items from a user profile based on item attributes. Second, the algorithm assigns scores to unrated items based on their distance from discovered clusters and on how unexpected they are for the user. Finally, the algorithm ranks unrated items according to their scores and suggests the recommendation list to the user [46].

The algorithm does not have many parameters, which makes it easy to control. Besides, an algorithm’s content-based nature might positively result in computational time depending on the dataset. However, the algorithm disregards the popularity subcomponent of serendipity.

### 3.3.2. TANGENT

Onuma et al. proposed the TANGENT algorithm to broaden user tastes [14]. The algorithm performs on a bipartite graph, where users and items correspond to nodes, while ratings correspond to edges. TANGENT detects groups of likeminded users and suggests items relevant to users from different groups. For example, if a target user belongs to comedy fans, the algorithm will suggest a movie relevant not only to comedy fans, but also to users from other groups, such as action fans or romance fans.

The proposed algorithm appears to broaden user tastes and retain the accuracy of recommendations. However, TANGENT might suggest popular items that users already know well, as items liked by users from different groups are likely to be popular.

### 3.3.3. RWR-KI

De Gemmis et al. proposed a serendipity-oriented RWR-KI algorithm [8]. RWR-KI stands for *random walk with restarts enhanced with knowledge infusion*. The algorithm suggests items based on the results of RWR that exploits an item similarity graph, where similarity indicates relatedness of items. The authors used WordNet and Wikipedia to calculate item similarities. By using uncommon similarity measure, the algorithm is supposed to suggest serendipitous items.

The algorithm suggests items based on the result of RWR, which receives a graph of item similarities and a set of starting nodes as an input. The item similarity graph contains nodes that correspond to items, and weighted edges, where weights correspond to similarities between connected items. Item similarities are inferred using item descriptions, WordNet and Wikipedia. Given an item similarity graph and a set of starting nodes $I_u$ (items rated by a user), RWR returns relatedness between items and a user profile. Items the user has not yet rated are ranked according to inferred relatedness values and suggested to the user.

The RWR-KI algorithm might suggest serendipitous items, as the algorithm uses an uncommon similarity measure that indicates relatedness of items instead of similarity. Furthermore, as the algorithm does not depend on user ratings, recommendations seem to be unbiased towards popular items. However, suggested items might still be similar to a user profile, since they might have similar descriptions. Besides, RWR-KI requires information enrichment using sources, such as WordNet or Wikipedia, which requires additional effort.

## 4. Evaluation strategies

To address *RQ3*, in this section we discuss evaluation strategies used to assess serendipity in recommender systems (RSs). We review offline and online evaluation strategies.

Depending on the specific domain, it is necessary to choose a suitable objective for an RS. For example, the goal of an RS that suggests friends might be to offer items with which a user is likely to be familiar. However, a music RS might suggest only novel items. It is crucial for the friend RS to have high accuracy. Meanwhile, the music recommendations require both accuracy and novelty [6]. Requirements of an RS therefore influence the choice of evaluation metrics and the decision on a recommendation algorithm [7]. According to different objectives, the evaluation metrics of the recommendation algorithms could be different as well.

Traditionally, evaluation strategies are divided into two categories: offline evaluation and online evaluation. In offline evaluation, researchers conduct experiments using pre-collected or simulated datasets. In the experiments, researchers hide some ratings and let an algorithm predict them based on the rest of the data. The performance of algorithms is usually presented by evaluation metrics. Based on results, researchers may choose a suitable algorithm [2,7,53].

Online evaluation involves real users interacting with a recommender system. For example, users might explicitly indicate whether they find recommended items novel or interesting. Online experiments can be conducted by using an RS prototype or a deployed functioning RS [2,7].

Usually, online evaluation is demanding and complicated, as it involves real users. Researchers, therefore, conduct offline experiments prior to the implementation of an online RS [7].

### 4.1. Offline evaluation

In offline experiments, researchers use pre-collected implicit or explicit user preferences to simulate the behavior of users and assess recommendation algorithms. Even though this kind of experiment only lets researchers answer a limited number of questions normally related to the prediction power of algorithms, it is often necessary to conduct offline experiments prior to online evaluation to decrease the number of candidate algorithms [7]. Regarding the evaluation of serendipity, many papers provide both qualitative and quantitative analysis of recommendation algorithms [7,21].

#### 4.1.1. Quantitative analysis

In offline experiments, researchers select a number of test users and hide some of their ratings. The hidden data are regarded as a test set and the rest as a training set. The candidate algorithms receive the training data as an input and predict the test set. The researchers filter candidate algorithms by their predictive ability [2,7].

Current datasets contain data regarding items relevant to a particular user, but lack data regarding serendipitous items. To assess serendipity in offline experiments, researchers make assumptions regarding serendipitous items and measure serendipity using the metrics described in Section 5.

#### 4.1.2. Qualitative analysis

It is possible to manually assess the serendipity of recommendations by comparing recommended items and user profiles. For example, Kaminskas and Bridge presented a qualitative analysis of suggestion lists offered by three popular recommendation algorithms [21]. One of the goals in the paper was to propose evaluation metrics for unexpectedness. To support the results of quantitative analysis, the authors compared a target user profile and recommended items for the user, including values of the proposed evaluation metrics. The items in user profiles and the recommended items referred to music artists.

### 4.2. Online evaluation

We regard online evaluation as evaluation that involves users interacting with the recommender system. This kind of evaluation may have different forms, such as questionnaires and experiments where users interact with a deployed RS [7].

Conducting online experiments is especially important for an assessment of novelty and serendipity, since it allows asking a user if she/he finds a recommended item unexpected or novel. On the one hand, these kinds of experiments are more representative than offline experiments, since they involve real users interacting with the RS [29]. On the other hand, online experiments are more demanding to conduct [7].

The main challenge in assessing serendipity in an online experiment is that it is difficult for a user to detect serendipitous items, since they are new to a user. Many papers have used different methods to investigate serendipity and novelty in RSs. In this section, we review the most common approaches mentioned in papers dedicated to serendipity.

#### 4.2.1. Questionnaire

Many RSs provide suggestions and explicitly ask users whether they find an item serendipitous [18–20,54]. A user might detect relevant and novel items, but it is difficult for the user to determine whether an item is unexpected, since the concept of unexpectedness is quite vague and might depend on many factors, such as mood, time of day and experimental setting. It is important to formulate a question about serendipity that a user would understand. For example, in [18] the authors asked participants whether they thought the recommendations were serendipitous. The results of the experiment were not statistically significant. The authors assumed this happened because most users were not sure what serendipity meant, as the term is difficult to translate to other languages. Table 4 reviews questions that were used in questionnaires to assess the serendipity of recommended items.

Regarding novelty, it is possible to utilize the setting suggested in [25]. In the context of music recommendation, a user might be offered a list of songs without any metadata. The user would listen to each song for 30 s and then inform a researcher whether she/he knows and likes the song.

**Table 4**
Serendipity assessment in experiments.

| Article | Application | Question |
|---|---|---|
| [19] | Music songs | "Exactly what I listen to normally... Something I would never have listened to otherwise" (Likert scale) |
| [54] | Music artists | "Did you find artists you wouldn't have found easily on your own and which you would like to listen to from now on?" |
| [20] | TV shows | "We inquired about serendipity for recommended shows, which means whether the recommended shows had hardly ever been heard of before by viewers but seemed to be interesting or whether they were already known but were unexpected" |
| [18] | Movies | "Are the recommendations serendipitous?" |
| [55] | IBM Connections | Participants indicated whether an item is surprising to them. |
| [8] | Movies | Analysis of facial expressions |

### 4.2.2. *Qualitative analysis*

It might be beneficial to ask users to give feedback on the recommendation list as a whole. Users might indicate what they liked or disliked the most or what they would improve and why [19,56]. For example, Zhang et al. presented a qualitative analysis based on users' comments on an RS [19]. The authors proposed a recommendation approach which was evaluated online. In the experiment, different RSs suggested music artists to users. Users not only rated suggested artists but also left comments on provided recommendation lists. The authors analyzed the comments and inferred factors that affected user perception of recommendation lists in the experiment.

### 4.2.3. *Deployed RS*

Sometimes it is necessary to improve an algorithm inside an already existing and deployed RS. Users can be divided into groups and introduced to the output of different recommendation algorithms. In the settings, researchers assume that users are not aware of the experiment. By analyzing user behavior, researchers make a decision between algorithms [7].

Assessing the impact of serendipitous items on user satisfaction in an online experiment with a deployed RS might be challenging for two reasons. First, by introducing serendipitous items, an RS might affect users differently depending on their level of experience [6]. Second, it might be difficult to detect an impact of serendipitous items on user behavior due to the emotional dimension of serendipity [9,16]. However, to the best of our knowledge, the number of experiments on serendipity in RSs that involve deployed RS is very limited.

## 5. Formal definitions of serendipity

To answer *RQ3*, in this section, we are going to review evaluation metrics that measure serendipity in recommender systems (RSs). The section provides a comparison of different metrics that have been proposed [19–21], the section provides their comparison, including their advantages and disadvantages.

Serendipity metrics can be divided into two categories: component metrics and full metrics. Component metrics measure different components of serendipity, such as novelty or unexpectedness, while full metrics measure serendipity as a whole.

### 5.1. *Component metrics*

Each component of serendipity can be measured in different ways. In this section, we are going to review serendipity component metrics that measure novelty and unexpectedness.

#### 5.1.1. *Novelty*

Vargas and Castells proposed two novelty metrics [27]. The first metric is based on an items distance from items a user has consumed (user profile) and has two variations:

$$nov_{va}^{dist1}(i,u) = \min_{j \in I_u} dist(i,j) \tag{4}$$

$$nov_{va}^{dist2}(i,u) = \frac{1}{|I_u|}\sum_{j \in I_u} dist(i,j) \tag{5}$$

where $dist(i, j)$ indicates the distance between items $i$ and $j$ and is formalized as follows:

$$dist(i,j) = 1 - sim(i,j), \tag{6}$$

where $sim(i, j)$ is any kind of similarity between items $i$ and $j$ ($sim(i, j) \in [0, 1]$). For example, it might be content-based cosine distance [57]. The second novelty metric is based on popularity and has two variations:

$$nov_{va}^{pop1}(i,u) = 1 - \frac{|U_i|}{|U|}, \tag{7}$$

$$nov_{va}^{pop2}(i,u) = -\log_2 \frac{|U_i|}{|U|}. \tag{8}$$

Nakatsuji et al. proposed a novelty metric similar to Eq. (4) [48]. They considered a taxonomy of items, where an item belongs to one of classes $CLS = (cls_1, cls_2, \ldots, cls_{\mathfrak{k}})$. The novelty metric is based on a minimum distance between an item and a user profile in an item taxonomy.

$$nov_{na}(i,u) = \min_{j \in I_u}(dis(cls_j, cls_i)), \tag{9}$$

where $cls_k$ is a class of item $k$ in an item taxonomy, while $dis(cls_j, cls_i)$ is the distance between classes $cls_j$ and $cls_i$ in the taxonomy.

#### 5.1.2. *Unexpectedness*

Kaminskas and Bridge suggested that serendipity includes two components: unexpectedness and relevance [21]. The authors indicated that unexpectedness reflects how dissimilar a suggested item is to a user profile, while relevance can be measured by accuracy metrics [2]. To measure unexpectedness, Kaminskas and Bridge suggested two pair-wise similarity metrics [21]: (1) point-wise mutual information and (2) content-based similarity.

Point-wise mutual information indicates how similar two items are based on the numbers of users who have rated both items and each item separately:

$$PMI(i,j) = -\log_2 \frac{p(i,j)}{p(i)p(j)} / \log_2 p(i,j), \tag{10}$$

where $p(i)$ is the probability that any user has rated item $i$, while $p(i, j)$ is the probability that items $i$ and $j$ are rated together. *PMI* ranges from $-1$ to 1, where $-1$ indicates that two items are never rated together, while 1 indicates that two items are always rated together. Based on point-wise mutual information, unexpectedness metrics have two variations:

$$unexp_{kam}^{co-occ1}(i,u) = \max_{j \in I_u} PMI(i,j) \tag{11}$$

$$unexp_{kam}^{co-occ2}(i,u) = \frac{1}{|I_u|}\sum_{j \in I_u} PMI(i,j) \tag{12}$$

**Table 5**
Primitive systems used in serendipity metrics.

| Metric | Article | Primitive system |
|---|---|---|
| $ser_{ad}$ | Adamopoulos and Tuzhilin [10,28] | Used two systems: one based on users' profiles and another based on the highest rating among the most popular items. |
| | de Gemmis et al. [8] | Used two systems: popularity and average rating. |
| $ser_{ge}$ | Ge et al. [25] | Suggested a system based on popularity. |
| | Lu et al. [41] | Used a system based on popularity. |
| | Sridharan [37] | Used a systems based on popular and highly rated items. |
| $ser_{mur}$ | Murakami et al. [20] | Used models based on users' profiles. |
| | Maksai et al. [39] | Used a systems based on popular items. |

The content-based unexpectedness metric is based on item attributes and has two variations. Both variations correspond to novelty metrics proposed by Vargas and Castells [27] (Eq. ([4,5])), where similarity is represented by Jaccard similarity.

### 5.2. Full metrics

Murakami et al. proposed a serendipity metric based on a primitive recommender system [20]:

$$ser_{mur}(u) = \sum_{i \in R_u} max(Pr_u(i) - Prim_u(i), 0) \cdot rel_u(i), \tag{13}$$

where $Pr_u(i)$ and $Prim_u(i)$ is the confidence of recommending item $i$ to user $u$ by the examined and the primitive models (RSs), respectively. The list of recommendations is represented by $R_u$, while the relevance of an item corresponds to $rel_u(i)$ (Table 3). A primitive RS is chosen arbitrarily and required to provide suggestions with low serendipity. Furthermore, Murakami et al. proposed the rank sensitive metric [20]:

$$ser\_r_{mur}(u) = \frac{1}{|R_u|}\sum_{j=1}^{|R_u|} max(Pr_u(i_j) - Prim_u(i_j), 0) \cdot rel_u(i_j) \cdot \frac{count_u(j)}{j}, \tag{14}$$

where $count_u(j)$ is the number of relevant items for user $u$ with a rank lower or equal to $j$. Later Ge et al. modified proposed metric (13) [25]:

$$ser_{ge}(u) = \frac{1}{|R_u|}\sum_{i \in (R_u \setminus PM)} rel_u(i), \tag{15}$$

where $PM$ is a set of items generated by a primitive recommender model. Adamopoulos and Tuzhilin then modified metric (15) by using an additional set of items [28]:

$$ser_{ad}(u) = \frac{1}{|R_u|}\sum_{i \in (R_u \setminus (E_u \cup PM))} rel_u(i), \tag{16}$$

where $E_u$ is a set of items that matches the interests of user $u$.

Table 5 demonstrates the classification of papers in terms of primitive RS. The serendipity metric $ser_{ad}$ includes popularity, similarity to a user profile and relevance. Other metrics $ser_{ge}$, $ser_{mur}$ and $ser\_r_{mur}$ would also capture these components if the primitive model simultaneously captured user interests and the popularity of items.

De Pessemier et al. generalized the serendipity metric originally proposed in [7]. The metric is calculated as follows [58]:

$$ser_{pe}(i, u) = \frac{1 + n_{I_u,max} - n_{I_u,f(i)}}{1 + n_{I_u,max}} \cdot rel_u(i), \tag{17}$$

$$n_{I_u,max} = \max_{f \in F}(n_{I_u,f}), \tag{18}$$

where $n_{I_u,f}$ is the number of items with attribute $f$ from user profile $I_u$, while $n_{I_u,max}$ is the maximum number of items with the same attribute from the user profile. The attribute of item $i$ is represented by $f(i)$.

### 5.3. Analysis of the evaluation metrics

Both component metrics and full metrics have their advantages and disadvantages [3]. Component metrics could be useful in measuring different aspects, such as popularity or similarity to a user profile. However, these metrics could be mistaken. For example, an algorithm suggests items 1, 2, 3, and 4, where items 1 and 2 are irrelevant, novel and unexpected, while items 3 and 4 are relevant but familiar and expected. The algorithm does not suggest any serendipitous items, but it might have high novelty, unexpectedness and accuracy.

Full metrics measure serendipity as a whole, but they also have disadvantages. Most full metrics are sensitive to primitive recommender systems [3,21,58]. By changing this parameter, one might obtain completely different and even contradictory results. Full metrics also disregard multiple levels of serendipity. As the relevance component of serendipity can be assessed using multiple judgments [59], one item might be more serendipitous than another. Full metrics that are not based on a primitive recommender system often disregard some serendipity subcomponents. For example, $ser_{pe}(i, u)$ measures relevance and dissimilarity of items to a user profile and ignores unpopularity.

To answer *RQ3*, in this section we classified and reviewed proposed metrics to assess serendipity in RSs. We did not include serendipity metrics designed to measure serendipity in particular domains such as friend recommendation [12], as they might not be applicable in other domains.

## 6. Future directions

In this section we are going to answer *RQ4* by providing ways to improve serendipity in recommender systems (RSs) that have not yet been widely adopted but that seem promising. Based on the reviewed literature, we are going to indicate four directions: (1) popularity and similarity in RSs, (2) context-aware RSs, (3) cross-domain RSs and (4) group RSs.

### 6.1. Popularity and similarity in RSs

According to the reviewed literature, most novelty-oriented and serendipity-oriented recommendation algorithms as well as evaluation metrics consider either popularity of items [8,18,41,49] or their similarity to a user profile [10,46,48]. One of the promising directions is the development of algorithms and evaluation metrics that capture both subcomponents of serendipity.

As we mentioned in Section 2.2, popularity is an important subcomponent of serendipity, as a user is likely to be familiar with globally popular items. A user might become aware of these items through different channels of information, such as radio, TV or friends [11].

Similarity of recommended items to a user profile is also important, since a user tends to look for items similar to what she/he likes, with the aim of finding more items she/he would enjoy. A user therefore might be aware of items similar to his/her profile [27].

It is challenging to suggest items that are relevant, unpopular and dissimilar to a user profile for two reasons. First, as most unpopular items are of low quality, a user is likely to consider them irrelevant [6]. Second, as a user tends to enjoy items similar to what she/he already likes, by increasing the number of items dissimilar to the user profile, an RS is likely to suggest items the user simply dislikes [42].

One of the ways to suggest items that are relevant, unpopular and dissimilar to a user profile is to combine each of these requirements into an optimization objective. For example, Zheng et al. suggested an optimization function of an SVD algorithm that combines relevance, popularity and similarity to a user profile [13]. Another way is to split a recommendation algorithm into different stages and handle one requirement per stage. For example, Full Auralist consists of three algorithms, each of which is responsible for one of the objectives [19].

### *6.2. Context-aware RSs*

In this paper, we assume that user tastes do not change with time; neither do they depend on the environment. However, many factors, such as weather, mood or location, can influence user preference for items [47,60,61]. For example, a user might prefer listening to soft music in the evening and energetic music in the morning. Suggesting songs according to the time of day would result in the improvement of user satisfaction.

Many recommender systems aim at predicting ratings users would give to items based on available ratings [47,61]. In contrast, context-aware RSs consider additional information, such as time, mood or location that might be relevant for generating recommendations [61,62].

The definition of serendipity should include the context, as each component of the concept is context-dependent [3]. An item that was serendipitous yesterday might not be serendipitous tomorrow. Serendipity thus could consist of relevance, novelty and unexpectedness in a given context.

Leveraging the context can be beneficial for suggesting serendipitous items, as context-aware RSs aim at suggesting items relevant to a user in a particular context. By using context, an RS can recommend items different from what the user usually consumes but also relevant, since the suggestions would fit a particular situation. For example, if a user who does not usually listen to rock and roll drives to Las Vegas, then the novel for the user song “Viva Las Vegas” by Elvis Presley would likely be considered serendipitous.

### *6.3. Cross-domain RSs*

Most RSs generate recommendations based on a single domain. In this paper, the term domain refers to “a set of items that share certain characteristics that are exploited by a particular recommender system” [63]. Cross-domain RSs take advantage of data from multiple domains [64]. For example, a cross-domain RS might leverage information from a book domain to improve recommendation performance in a movie domain [65].

Several types of cross-domain RSs can improve the serendipity of recommendations:

- User modeling. By using information from multiple domains, this kind of system can infer items with which users are already familiar. Leveraging additional information may also help discover additional interests of users, which might result in serendipitous suggestions.
- Relevance. Use of additional domains may help to recommend more items relevant for a user, improving serendipity, as relevance is one of serendipity’s components.
- Context. Additional domains may contain information regarding context [64,66]. By using this kind of data, one may apply context-aware recommendation algorithms and improve serendipity, as indicated in Section 6.2.

Cross-domain RSs can suggest combinations of items from different domains. For example, a user could receive a recommendation of a venue to visit and a song to listen to in that particular venue. Combinations of items affect the definition of the concept, as it is not clear whether a combination or each item in the combination has to be novel, unexpected and relevant [3].

### *6.4. Group RSs*

Most recommender systems suggest items to an individual user [67]. However, in some cases it is important to generate recommendations for a group of users. For example, a few friends might need to choose a movie to watch together.

Suggesting items serendipitous to a group of users is more challenging than suggesting these kinds of items to an individual, as items serendipitous to the group must be novel, unexpected and relevant to each individual in the group. Items serendipitous to the group could be represented by an intersection of sets containing items serendipitous to each user in the group. The cardinality of the intersection is less than or equal to the cardinality of each user’s set of serendipitous items. The serendipity of a group recommendation could be assessed by measuring serendipity for each user in the group.

The presence of different individuals in the group could be considered as the context of the recommendation. An item not serendipitous to a particular user might become serendipitous when it is consumed around certain individuals. To the best of our knowledge, efforts on recommendation of serendipitous items to a group of users are very limited.

## **7. Conclusion**

In this paper, we investigated serendipity in recommender systems (RSs) and answered our research questions.

*RQ1. What is serendipity in recommender systems? What makes certain items serendipitous for particular users?*

In Section 2, we defined serendipity as a property that reflects how good an RS is at suggesting serendipitous items that are relevant, novel and unexpected. Novelty and unexpectedness require serendipitous items to be relatively unpopular and significantly different from a user profile.

*RQ2. What are the state-of-the-art recommendation approaches that suggest serendipitous items?*

We overviewed serendipity-oriented algorithms and provided two classifications of them in Section 3. The first classification is based on the architecture of the algorithms, while the second classifies algorithms depending on the phase of recommendation that is responsible for serendipity.

*RQ3. How can we assess serendipity in recommender systems?*

We overviewed evaluation strategies to assess serendipity in RSs in Section 4. We discussed qualitative and quantitative methods used in experimental studies related to serendipity.

We also presented two categories of serendipity metrics in Section 5: component metrics and full metrics. Component metrics measure serendipity components, while full metrics measure serendipity as a whole.

*RQ4. What are the future directions of serendipity in RSs?*

In Section 6, we indicated four future directions of serendipity-oriented algorithms. The first direction indicates that serendipity-oriented algorithms and evaluation metrics should take into account both item popularity and similarity to a user profile. The second direction involves context-aware recommender systems. As context-aware RSs can pick items based on the context rather than user tastes, this kind of system might suggest more serendipitous items. The third direction includes cross-domain RS. Having information from additional domains, an RS can infer which items are familiar to a user, suggest items relevant to the user and select items that fit the context. The fourth direction is dedicated to group recommendations. It is difficult to suggest items serendipitous to each user in a group of individuals, as the tastes of each user must be considered.

We hope not only that this paper will make the reader aware of serendipity and related concepts in RSs, but also that it will help one conduct one's own experiments by picking suitable algorithms, evaluation strategies and metrics. We also hope that, as a result, the presented overview will contribute to the development and improvement of recommendation algorithms focused on user satisfaction rather than on accuracy metrics.

## Acknowledgement

The research at University of Jyväskylä was performed in the MineSocMed project, partially supported by the Academy of Finland, grant #268078. The research was also partially supported by Natural Science Foundation of China #71402083.

## References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender Systems Handbook, Springer US, pp. 1–35.
[2] M.D. Ekstrand, J.T. Riedl, J.A. Konstan, Collaborative filtering recommender systems, Found. Trends Hum. Comput. Int. 4 (2) (2011) 81–173.
[3] D. Kotkov, J. Veijalainen, S. Wang, Challenges of serendipity in recommender systems, in: Proceedings of the 12th International Conference on Web Information Systems and Technologies., vol. 2, SCITEPRESS, 2016, pp. 251–256.
[4] J.B. Schafer, J. Konstan, J. Riedl, Recommender systems in e-commerce, in: Proceedings of the 1st ACM Conference on Electronic Commerce, ACM, New York, NY, USA, 1999, pp. 158–166.
[5] J. Bobadilla, F. Ortega, A. Hernando, A. Gutierrez, Recommender systems survey, Knowl. Based Syst. 46 (2013) 109–132.
[6] Ò. Celma Herrada, Music Recommendation and Discovery in the Long Tail, Ph.D. thesis, Universitat Pompeu Fabra, 2009.
[7] G. Shani, A. Gunawardana, Recommender Systems Handbook, vol. 51, Springer US, pp. 695–717.
[8] M. de Gemmis, P. Lops, G. Semeraro, C. Musto, An investigation on the serendipity problem in recommender systems, Inf. Process. Manage. 51 (2015) 695–717.
[9] L. Iaquinta, M. de Gemmis, P. Lops, G. Semeraro, M. Filannino, P. Molino, Introducing serendipity in a content-based recommender system, in: Proceedings of the 8th International Conference on Hybrid Intelligent Systems, 2008, pp. 168–173.
[10] P. Adamopoulos, A. Tuzhilin, On unexpectedness in recommender systems: or how to better expect the unexpected, ACM Trans. Intell. Syst. Technol. 5 (4) (2014) 1–32.
[11] E. Tacchini, Serendipitous Mentorship in Music Recommender Systems, Ph.D. thesis, Università degli Studi di Milano, 2012.
[12] M. Manca, L. Boratto, S. Carta, Behavioral data mining to produce novel and serendipitous friend recommendations in a social bookmarking system, Inf. Syst. Front. (2015) 1–15.
[13] Q. Zheng, C.-K. Chan, H.H. Ip, An unexpectedness-augmented utility model for making serendipitous recommendation, in: Advances in Data Mining: Applications and Theoretical Aspects, vol. 9165, Springer International Publishing, 2015, pp. 216–230.
[14] K. Onuma, H. Tong, C. Faloutsos, Tangent: a novel, 'surprise me', recommendation algorithm, in: Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2009, pp. 657–666.
[15] Q. Meng, K. Hatano, Visualizing basic words chosen by latent dirichlet allocation for serendipitous recommendation, in: Proceedings of the 3rd International Conference on Advanced Applied Informatics, 2014, pp. 819–824.
[16] A. Foster, N. Ford, Serendipity and information seeking: an empirical study, J. Doc. 59 (3) (2003) 321–340.
[17] P. André, M.C. Schraefel, J. Teevan, S.T. Dumais, Discovery is never by chance: designing for (un)serendipity, in: Proceedings of the Seventh ACM Conference on Creativity and Cognition, ACM, New York, NY, USA, 2009, pp. 305–314.
[18] A. Said, B. Fields, B.J. Jain, S. Albayrak, User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, 2013, pp. 1399–1408.
[19] Y.C. Zhang, D.O. Séaghdha, D. Quercia, T. Jambor, Auralist: introducing serendipity into music recommendation, in: Proceedings of the 5th ACM International Conference on Web Search and Data Mining, ACM, New York, NY, USA, 2012, pp. 13–22.
[20] T. Murakami, K. Mori, R. Orihara, Metrics for evaluating the serendipity of recommendation lists, in: New Frontiers in Artificial Intelligence, vol. 4914, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 40–46.
[21] M. Kaminskas, D. Bridge, 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.
[22] Y. Levy, T.J. Ellis, A systems approach to conduct an effective literature review in support of information systems research, Inf. Sci. 9 (2006) 181–212.
[23] K. Kapoor, V. Kumar, L. Terveen, J.A. Konstan, P. Schrater, "I like to explore sometimes": adapting to dynamic user novelty preferences, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2015, pp. 19–26.
[24] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2002, pp. 253–260.
[25] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the 4th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2010, pp. 257–260.
[26] L. Iaquinta, G. Semeraro, M. de Gemmis, P. Lops, P. Molino, Can a recommender system induce serendipitous encounters?, IN-TECH, 2010.
[27] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: Proceedings of the 5th ACM Conference on Recommender Systems, New York, NY, USA, 2011, pp. 109–116.
[28] P. Adamopoulos, A. Tuzhilin, On unexpectedness in recommender systems: or how to expect the unexpected, in: Workshop on Novelty and Diversity in Recommender Systems, at the 5th ACM International Conference on Recommender Systems, 2011, pp. 11–18.
[29] S.M. McNee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: Proceedings of the Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 2006, pp. 1097–1101.
[30] Most Untranslatable Word, (http://www.todaytranslations.com/blog/most-untranslatable-word/). [Online; accessed 20-November-2015].
[31] The Three Princes of Serendip, (http://livingheritage.org/three_princes.htm). [Online; accessed 20-November-2015].
[32] T.G. Remer, Serendipity and the Three Princes: From the Peregrinaggio of 1557, University of Oklahoma Press, p. 20.
[33] Serendipity - Definition of Serendipity by the Free Dictionary, (http://www.thefreedictionary.com/serendipity). [Online; accessed 20-November-2015].
[34] P. Van Andel, Anatomy of the unsought finding. serendipity: origin, history, domains, traditions, appearances, patterns and programmability, Brit. J. Philos. Sci. 45 (1994) 631–648.
[35] P. Resnick, H.R. Varian, Recommender systems, Commun. ACM 40 (1997) 56–58.
[36] J. Corneli, A. Pease, S. Colton, A. Jordanous, C. Guckelsberger, Modelling serendipity in a computational context, Comput. Res. Repos. abs/1411.0440 (2014).
[37] S. Sridharan, Introducing Serendipity in Recommender Systems Through Collaborative Methods, Ph.D. thesis, University of Rhode Island, 2014.
[38] Y.-S. Chiu, K.-H. Lin, J.-S. Chen, A social network-based serendipity recommender system, in: Proceedings of the International Symposium on Intelligent Signal Processing and Communications Systems, 2011, pp. 1–5.
[39] A. Maksai, F. Garcin, B. Faltings, Predicting online performance of news recommender systems through richer evaluation metrics, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2015, pp. 179–186.
[40] P. Bieganski, J.A. Konstan, J.T. Riedl, System, method and article of manufacture for making serendipity-weighted recommendations to a user, 2001, Patent US6334127 B1.
[41] Q. Lu, T. Chen, W. Zhang, D. Yang, Y. Yu, 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, volume 1
[42] L.W. James, Nudging Music Serendipity, Ph.D. thesis, University of Cambridge, 2013.
[43] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. Artif. Intell. 2009 (2009) 1–20.
[44] A. Tejeda-Lorente, C. Porcel, J. Bernabé-Moreno, E. Herrera-Viedma, Refore: a recommender system for researchers based on bibliometrics, Appl. Soft Comput. 30 (2015) 778–791.
[45] C. Martinez-Cruz, C. Porcel, J. Bernabe-Moreno, E. Herrera-Viedma, A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling, Inf. Sci. 311 (2015) 102–118.

[46] T. Akiyama, K. Obara, M. Tanizaki, Proposal and evaluation of serendipitous recommendation method using general unexpectedness, in: Proceedings of the Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies at 4th ACM Conference on Recommender Systems, Barcelona, Spain, 2010, pp. 3–10.
[47] G. Adomavicius, A. Tuzhilin, Recommender Systems Handbook, Springer US, Boston, MA, pp. 217–253.
[48] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, K. Fujimura, T. Ishida, Classical music for rock fans? Novel recommendations for expanding user interests, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ACM, New York, NY, USA, 2010, pp. 949–958.
[49] N. Kawamae, Serendipitous recommendations via innovators, in: Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2010, pp. 218–225.
[50] H. Tong, C. Faloutsos, J.-Y. Pan, Random walk with restart: fast solutions and applications, Knowl. Inf. Syst. 14 (3) (2008) 327–346.
[51] N. Kawamae, H. Sakano, T. Yamada, Personalized recommendation based on the personal innovator degree, in: Proceedings of the 3rd ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2009, pp. 329–332.
[52] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the 4th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2010, pp. 39–46.
[53] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22Nd Annual International Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 1999, pp. 230–237.
[54] M. Taramigkou, E. Bothos, K. Christidis, D. Apostolou, G. Mentzas, Escape the bubble: guided exploration of music preferences for serendipity and novelty, in: Proceedings of the 7th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2013, pp. 335–338.
[55] I. Guy, R. Levin, T. Daniel, E. Bolshinsky, Islands in the stream: a study of item recommendation within an enterprise social stream, in: Proceedings of the 38th International ACM Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2015, pp. 665–674.
[56] M.N. Jelassi, S.B. Yahia, E.M. Nguifo, Towards more targeted recommendations in folksonomies, Soc. Netw. Anal. Min. 5 (2015) 1–18.
[57] P. Lops, M. de Gemmis, G. Semeraro, Recommender Systems Handbook, Springer US, Boston, MA, pp. 73–105.
[58] T. Pessemier, S. Dooms, L. Martens, Comparison of group recommendation algorithms, Multimed. Tools Appl. 72 (2013) 2497–2541.
[59] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (2002) 422–446.
[60] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, E. Duval, Context-aware recommender systems for learning: a survey and future challenges, IEEE Trans. Learn. Technol. 5 (4) (2012) 318–335.
[61] F.V. Steeg, Context-Aware Recommender Systems, Utrecht University, Master's Thesis, 2015.
[62] L. Baltrunas, Context-aware collaborative filtering recommender systems, Ph.D. thesis, 2011.
[63] I. Fernández-Tobías, I. Cantador, M. Kaminskas, F. Ricci, Cross-domain recommender systems: a survey of the state of the art, in: Proceedings of the 2nd Spanish Conference on Information Retrieval, 2012, pp. 187–198.
[64] I. Cantador, P. Cremonesi, Tutorial on cross-domain recommender systems, in: Proceedings of the 8th ACM Conference on Recommender Systems, New York, NY, USA, 2014, pp. 401–402.
[65] P. Winoto, T. Tang, If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? A study of cross-domain recommendations, New Generat. Comput. 26 (3) (2008) 209–225.
[66] I. Cantador, P. Cremonesi, Cross-domain Recommender Systems, 2014, (http://recsys.acm.org/wp-content/uploads/2014/10/recsys2014-tutorial-cross_domain.pdf) [Online; accessed 20-November-2015].
[67] J. Masthoff, Recommender Systems Handbook, Springer US, Boston, MA, pp. 677–702.

PII

# INVESTIGATING SERENDIPITY IN RECOMMENDER SYSTEMS BASED ON REAL USER FEEDBACK

by

Denis Kotkov, Joseph A. Konstan, Qian Zhao, Jari Veijalainen 2018

Proceedings of SAC 2018: Symposium on Applied Computing (Accepted)

# Investigating Serendipity in Recommender Systems Based on Real User Feedback

Denis Kotkov
Faculty of Information Technology,
University of Jyvaskyla, Finland
deigkotk@student.jyu.fi

Joseph A. Konstan
University of Minnesota
Minneapolis, MN, USA
konstan@umn.edu

Qian Zhao
University of Minnesota
Minneapolis, MN, USA
zhaox331@umn.edu

Jari Veijalainen
Faculty of Information Technology,
University of Jyvaskyla, Finland
jari.veijalainen@jyu.fi

## ABSTRACT

Over the past several years, research in recommender systems has emphasized the importance of serendipity, but there is still no consensus on the definition of this concept and whether serendipitous items should be recommended is still not a well-addressed question. According to the most common definition, serendipity consists of three components: relevance, novelty and unexpectedness, where each component has multiple variations. In this paper, we looked at eight different definitions of serendipity and asked users how they perceived them in the context of movie recommendations. We surveyed 475 users of the movie recommender system, MovieLens regarding 2146 movies in total and compared those definitions of serendipity based on user responses. We found that most kinds of serendipity and all the variations of serendipity components broaden user preferences, but one variation of unexpectedness hurts user satisfaction. We found effective features for detecting serendipitous movies according to definitions that do not include this variation of unexpectedness. We also found that different variations of unexpectedness and different kinds of serendipity have different effects on preference broadening and user satisfaction. Among movies users rate in our system, up to 8.5% are serendipitous according to at least one definition of serendipity, while among recommendations that users receive and follow in our system, this ratio is up to 69%.

## CCS CONCEPTS

• **Information systems → Recommender systems**; *Personalization*;

## KEYWORDS

recommender systems; serendipity; relevance; novelty; unexpectedness

**ACM Reference Format:**
Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. 2018. Investigating Serendipity in Recommender Systems Based on Real User Feedback. In *Proceedings of SAC 2018: Symposium on Applied Computing (SAC 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3167132.3167276

## 1 INTRODUCTION

Recommender systems are designed to help users find interesting items, when the number of these items is overwhelming [24]. In this paper, the term *item* refers to a piece of information, which is a reference to an object, such as a good, service or process that a recommender system suggests to the user [17]. An item can refer to any object, such as a movie, song or book.

Traditionally, recommendation algorithms have been optimized for accuracy [15], which indicates the predictive power of these algorithms. However, recently the focus of the recommender systems community started shifting towards factors beyond accuracy [15], as accuracy alone does not always result in user satisfaction. One of the factors of recommender systems beyond accuracy is serendipity [15].

According to the dictionary, serendipity is "the faculty of making fortunate discoveries by accident"[1]. The term serendipity was first introduced in the context of recommender systems in early 2000s [10]. Many researchers employed their definitions of this concept, but there is no consensus on the definition of serendipity yet [17, 19]. The most common definitions of the concept include three components: relevance, novelty and unexpectedness [13, 19, 21], while these components have multiple definitions [19].

It is unclear whether serendipitous items should be recommended to users. According to most claims from the literature on serendipity in recommender systems, there are two main reasons for collaborative recommender systems [8] to suggest serendipitous items: they broaden user preferences [10, 29, 30] and increase user satisfaction [1, 20, 22, 29]. However, the studies showing

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

that serendipitous items are in any way better than relevant non-serendipitous ones are very limited and often have a small number of samples [26, 27, 29].

Novelty and unexpectedness have multiple definitions [16], but it is unclear whether different variations of the same component have different effects on our metrics. In this paper, the term *metrics* refers to preference broadening and user satisfaction.

Researchers often indicate that serendipitous items are very rare [4, 17]. However, it is unclear exactly how rare these items are, as it might not be worth of suggesting them due to their rareness and a high risk of suggesting irrelevant items, while optimizing for serendipity [19].

This paper presents the first study that looks across multiple definitions of serendipity. It compares these definitions and their components in terms of their value for a user in a user study. We employ the most common definition of the concept, which requires serendipity to include three components: relevance, novelty and unexpectedness, where each component has multiple definitions resulting in eight definitions of serendipity. We detect the most important features for predicting serendipitous items and estimate the ratio of these items among items rated in a typical collaborative-filtering-based recommender system [8]. We conduct this study in the online movie recommender system, MovieLens[2], where we ask users retrospectively about movies they have rated. In this paper, we address the following research questions (RQs):

*RQ1. What are the effects of variations of serendipity components on broadening user preferences and user satisfaction?*

*RQ2. What are the effects of different kinds of serendipity on broadening user preferences and user satisfaction?*

*RQ3. What are the effective features for detecting serendipitous movies? What are the value ranges of these features for typical serendipitous movies?*

*RQ4. How rare are serendipitous movies among movies rated by the users in a typical collaborative-filtering-based recommender system? To what extent does this kind of system help users find these movies?*

Most serendipity-oriented algorithms are evaluated based on publicly available datasets that lack user feedback regarding serendipitous items. To label certain items serendipitous to users, researchers tend to make assumptions regarding serendipity, such as serendipitous items are unpopular [20, 22, 30] or dissimilar to items users rated in the past [1]. These assumptions might not correspond to real life scenarios. We therefore publish our collected dataset to allow other researchers conduct experiments related to serendipity in recommender systems[3]. This paper has the following contributions:

- We conduct literature review and operationalize common definitions of serendipity.
- We compare different definitions of serendipity and their components in terms of preference broadening and user satisfaction.
- We find a subset of features that are effective for detecting movies that are serendipitous according to certain definitions, which might be useful for suggesting these these serendipitous movies in an off-line evaluation of recommendation algorithms.
- We estimate the ratio of serendipitous movies, which might help to decide whether it is worth optimizing for serendipity.
- We publish the first dataset that includes user feedback regarding serendipitous movies to allow other researchers conduct their experiments.

In this study, we surveyed 475 users and found that most kinds of serendipity and all the variations of serendipity components broaden user preferences, but one variation of unexpectedness hurts user satisfaction. We found effective features for detecting movies that are serendipitous according to definitions that do not include this variation of unexpectedness. These features are predicted rating, popularity, content-based and collaborative similarity to movies users watched in the past. We also found that different variations of unexpectedness and different kinds of serendipity have different effects on preference broadening and user satisfaction. Among movies users rate in a typical collaborative-filtering-based system, up to 8.5% are serendipitous according to at least one definition of serendipity, while among recommendations that users receive and follow in the system, this ratio is up to 69%. We only provide an upper bound estimation due to the bias of our dataset (we selected relatively unpopular movies).

The paper is organized as follows: in section 2, we briefly review related work. Section 3 describes our survey and the method to invite users to our study. Section 4 describes the dataset we collected. In section 5, we analyze the collected data and answer our research questions. In section 6, we discuss the results, while in section 7 we discuss limitations of our study and future work. Finally, we conclude in section 8.

## 2 RELATED WORK

Many authors focused on different aspects of serendipity and used different definitions of the concept. In this section, we focus on (1) the value of serendipitous items for users due to the objective of our research, (2) definitions of serendipity, as this is the key concept of our research and (3) inquiring for serendipity, as we conduct a survey where we ask users to indicate serendipitous movies.

### 2.1 Why Serendipitous Items?

Most previous studies on this topic indicate three reasons to recommend serendipitous items. Researchers have claimed that serendipitous items help overcome the overspecialization problem (for content-based filtering algorithms) [1, 13], broaden user preferences [10, 29, 30] and increase user satisfaction [1, 20, 22, 29].

However, studies that provide evidence for the benefit of recommending serendipitous items are very limited. The only study we found that measured the benefit of serendipitous recommendations was conducted by Zhang at el. [29]. In the study, 21 users were offered recommendations from serendipity-oriented and accuracy-oriented algorithms. Although users gave lower ratings to recommendations provided by the serendipity-oriented algorithm than those provided by the accuracy-oriented algorithm, the majority of users preferred using the serendipity-oriented one [29].

To the best of our knowledge, there are no studies that compare items corresponding to different definitions of serendipity in terms

[2]movielens.org
[3]The dataset is available on the GroupLens website: https://grouplens.org/datasets/movielens/

of their value for users. In this paper, we compare different definitions of serendipity in terms of preference broadening and user satisfaction. We do not consider the overspecialization problem, as in MovieLens, users mostly receive recommendations generated by collaborative filtering algorithms [7], while the overspecialization problem is more prominent for content-based filtering algorithms [13].

## 2.2 The Definitions of Serendipity

There is no consensus on the definition of serendipity in recommender systems [17, 19]. However, most authors indicate that serendipitous items must be relevant, novel and unexpected to a user [17]. An item is relevant to a user if the user expresses or will express their preference for the item in the future by liking or consuming the item depending on the application scenario [18]. Novelty of an item to a user depends on how familiar the user is with the item. An item can be novel to a user in different ways:

(1) The user has never heard about the item [16].
(2) The user has heard about the item, but has not consumed it.
(3) The user has consumed the item and forgot about it [16].

Studies on serendipity in recommender systems often neglect the definition of unexpectedness. We present a number of definitions corresponding to the component. An item can be unexpected to the user if:

(1) The user does not expect this item to be relevant to them [1].
(2) The user does not expect this item to be recommended to them.
(3) The user would not have found this item on their own [1, 9–11, 27].
(4) The item is significantly dissimilar to items the user usually consumes [14, 19, 29].
(5) The user does not expect to find this item, as the user is looking for other kinds of items [1].

In this paper, we investigate serendipity according to different definitions: serendipitous items are relevant, novel and unexpected, where unexpectedness corresponds to definitions 1–4 and novelty corresponds to all the definitions listed above (we merged definitions 1 and 3 together). We do not consider definition 5 of unexpectedness, as many users in MovieLens do not normally look for particular kinds of movies. Furthermore, if they know what they are looking for, they are unlikely to remember what kinds of movies they were looking for after they have watched the movie they found.

## 2.3 Inquiry About Serendipity

There are two ways of inquiring about serendipity in surveys: posing questions concerning serendipity while viewing it as atomic, or exposing its components and posing suitable questions concerning them. The former way of inquiring requires less effort from a user and simplifies the analysis of user answers. However, asking one question does not allow to investigate components of serendipity and is likely to be confusing for users due to the complexity of the concept [25]. For example, Said et al. compared results of collaborative filtering algorithms in terms of serendipity in an online experiment [25]. The authors directly asked users whether they found recommendations serendipitous and received statistically insignificant results in terms of serendipity when they compared performance of the algorithms. The authors noted that this insignificance was caused by the complexity of the concept especially for non-native speakers [25].

Inquiring about each component of serendipity requires the users to answer several questions, where one or more questions measure one concept at a time. For example, Zhang et al. considered an item serendipitous to a user, when that user gave an item a high rating and indicated that this item was novel and unexpected to them [29]. Although this way of inquiring about serendipity is more demanding for users, it allows to investigate each component of serendipity and measure serendipity more precisely than asking just the one question.

It is also possible to use implicit user feedback on items to assess serendipity. For example, de Gemmis et al. analyzed facial expressions of users to detect the movies that were serendipitous to these users [6].

In this paper, we conduct a survey, where we inquire about serendipity by asking users one question per component of serendipity according to each definition employed in this research, as our goal of investigating each definition of serendipity requires precise assessment of the concept and its components.

## 3 THE SURVEY DESIGN

The main functionality of MovieLens allows users to rate movies they watched on the scale from 0.5 to 5 stars with the granularity of 0.5 star and receive recommendations generated based on the ratings. MovieLens does not allow users to indicate how long ago they had watched a particular movie. Users might rate a movie in a while after they had watched it. MovieLens also allows users to perform other actions, such as adding a movie to the list of movies to watch (a watch list), assigning keywords (tags) to movies, and adding new movies to the system.

The ideal way to measure serendipity in a movie domain would be to inquire a user about novelty and unexpectedness before the user has watched the movie and inquire the user about the relevance of this movie afterwards. MovieLens allows us to implement this experimental setting by conducting two surveys: the first one, when a user adds movies to their watch list and the second one, when the user rates movies from their watch list. However, this setting has two main disadvantages: (a) users mostly add movies they expect to enjoy watching to their watch lists, and (b) only a few users use the functionality of adding movies into watch lists and even fewer users rate movies from their watch lists. We therefore decided to ask users about their experience retrospectively.

We invited users via emails to complete an online survey regarding movies they rated during the last three months before the experiment. We chose three months, because it is likely that users still remember their experience of rating those movies when the users take our survey. Our inclusion criteria for users was as follows: we selected users who rated at least five movies with a rating of at least 3.5 stars from December 30, 2016 till March 30, 2017 (the experiment started on April 1, 2017) and at least one month after their registrations (for users who joined MovieLens after November

30, 2016). We assumed that users rate movies that they watched before the registration during the first month after their registration. In our survey, we picked five movies rated during the three months before the experiment and asked users to answer five questions and rate forty statements about the five movies we picked for each user (one question and eight statements per movie). We picked the least popular movies (i.e. those with the smallest number of ratings in the system) among movies users rated during that period of time. We expected that users discovered these movies in our system, as users are likely to hear about popular movies from other sources, such as friends, family and TV.

We emailed 2305 users who met our inclusion criteria and received a response from 522 users, but only 475 users rated all the statements and answered the question about at least one movie. In total, these users rated all the statements and answered all the questions about 2166 movies.

Table 1 demonstrates statements we asked users to rate. Serendipity components that correspond to the statements and the definitions of serendipity. We asked each user to rate eight statements using the following scale: “strongly agree”, “agree”, “neither agree nor disagree”, “disagree”, “strongly disagree”, “don’t remember”. Each definition of serendipity consists of three components: relevance, novelty and unexpectedness. As we only asked users about movies they rated with at least 3.5 stars, we assumed that all the movies we asked users about are relevant to these users. We picked four definitions of unexpectedness and two definitions of novelty:

- Unexpectedness to be relevant (unexp_rel) corresponds to the original definition of unexpectedness 1 from the literature review section (section 2.2).
- Unexpectedness to be found (unexp_find) corresponds to the original definition of unexpectedness 3.
- Implicit unexpectedness (unexp_imp) corresponds to the original definition of unexpectedness 4.
- Unexpectedness to be recommended (unexp_rec) corresponds to the original definition of unexpectedness 2.
- Strict novelty (s_nov) corresponds to the original definitions of novelty 1 and 3.
- Motivationally novelty (m_nov) corresponds to the original definition of novelty 2.

This resulted in eight sets of serendipitous movies. For example, we considered a movie motivationally serendipitous (implicit) if a user rated statements 2 and 5 (m_nov and unexp_imp) with replies “strongly agree” or “agree”. One movie can belong to several definitions simultaneously.

In this paper, the term *movie* refers to a user-movie pair. For example, a relevant movie corresponds to a user-movie pair, where the user considers the movie relevant, while other users might not consider this movie relevant.

## 4 SUMMARY STATISTICS OF THE DATASET

Figure 1 demonstrates the distribution of the answers to the question of how long ago users watched movies we picked for the survey. Users watched around 60% of the movies we asked them about less than 6 months before the survey and therefore it is likely that they still remember their watching experience for the movies. We removed movies that users indicated they did not watch (20 movies or 1%) from our dataset.

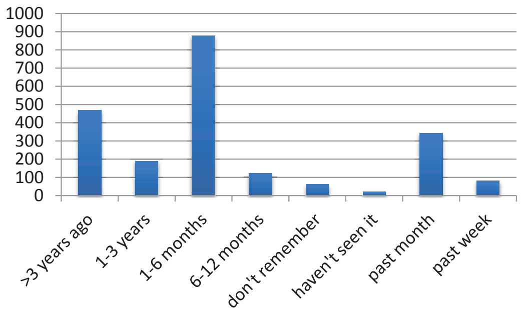


**Figure 1: Distribution of answers to the question “When did you watch this movie for the first time?”**

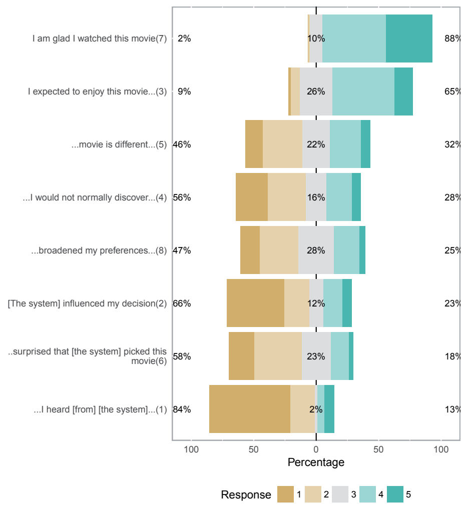


**Figure 2: Distributions of answers (“1” - strongly diagree, “2” - disagree, “3” - neither agree nor disagree, “4” - agree, “5” - strongly agree)**

Figure 2 demonstrates distributions of the user responses. Users indicated that they were glad they watched the majority of movies we asked them about, which might have resulted from our inclusion criteria (we picked movies users rated at least 3.5 stars in MovieLens).

Table 2 demonstrates the numbers of movies that are serendipitous according to the different definitions along with all the movies we picked for the survey. The sets of different kinds of serendipitous

**Table 1: Statements 1-6 correspond to components of serendipity, statements 7 and 8 correspond to our metrics and "+" indicates inclusion of a component to the definition of serendipity, i.e. the user checks "agree" or "strongly agree" to the corresponding statement, except for Statement 3 where inclusion means checking "disagree", "strongly disagree" or "neither agree nor disagree".**

| Statement # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Statement | The first time I heard of this movie was when MovieLens suggested it to me. | MovieLens influenced my decision to watch this movie. | I expected to enjoy this movie before watching it for the first time. | This is the type of movie I would not normally discover on my own; I need a recommender system like MovieLens to find movies like this one. | This movie is different (e.g., in style, genre, topic) from the movies I usually watch. | I was (or, would have been) surprised that MovieLens picked this movie to recommend to me. | I am glad I watched this movie. | Watching this movie broadened my preferences. Now I am interested in a wider selection of movies. |
| Name | s_nov | m_nov | unexp_rel | unexp_find | unexp_imp | unexp_rec | Satisfaction | Preference broadening |
| Description | A novelty component (strict novelty) | A novelty component (motivational novelty) | An unexpectedness component (unexpectedness (relevance)) | An unexpectedness component (unexpectedness (find)) | An unexpectedness component (unexpectedness (implicit)) | An unexpectedness component (unexpectedness (recommend)) | Our satisfaction metric | Our preference broadening metric |
| s_ser_rel | + | | + | | | | | |
| s_ser_find | + | | | + | | | | |
| s_ser_imp | + | | | | + | | | |
| s_ser_rec | + | | | | | + | | |
| m_ser_rel | | + | + | | | | | |
| m_ser_find | | + | | + | | | | |
| m_ser_imp | | + | | | + | | | |
| m_ser_rec | | + | | | | + | | |

movies overlap. For example, 48 m_ser_imp movies are at the same time m_ser_rec.

## 5 ANALYSIS

In this section, we explain how we analyzed the collected data set. All the movies we picked are considered relevant by users due to our inclusion criteria. In the following text, we omit indicating that the movies are relevant for brevity.

We employed a cumulative link mixed-effect regression model [5]. We used this model to predict a dependent ordinal variable $Y$ with independent binary variables $x_1, x_2, ..., x_n$. Consider an observation from the $k$th user. The model fits probability of the dependent variable to fall in $j = 1, 2, ..., J$ categories as follows:

$$log\left(\frac{P(Y \leqslant j)}{1 - P(Y \leqslant j)}\right) = \alpha_j + \beta_1 x_1 + ... + \beta_n x_n + u_k, \quad (1)$$

where $P(Y \leqslant j)$ is a cumulative probability that $Y \leqslant j$, while $\beta_1, \beta_2, ..., \beta_n$ are the coefficients of the model, where $n$ depends on the number of independent variables included in the model. Parameter $\alpha_j$ corresponds to the intercept for category $j$, while $u_k$ is a random intercept for the $k$th user and $u_k \sim Gaussian(0, \sigma)$ where $\sigma$ is an additional parameter describing the dispersion of the random intercept effect. See [2] for more details on this analytical model for ordinal response data. We use the R *ordinal* package (a standard implementation of the model) to conduct the analysis [5].

In our analysis, the dependent variables correspond to the likert-scale responses users gave to the statements 7 and 8 (preference broadening or user satisfaction, separately). They take values from 1 to 5. Our independent variables correspond to whether a movie satisfies a particular definition of serendipity or a variation of serendipity component.

We conducted statistical tests for each coefficient of the regression models, where the null hypothesis was that the coefficient equals zero meaning that changes in the independent variable were not associated with changes in the dependent variable. To control false discoveries, we used Bonferroni correction procedure, which adjusted our critical p-value from 0.05 to 0.0005 [12].

### 5.1 Effects of the Serendipity Components

To answer RQ1 and investigate the effects of variations of serendipity components separately, we ran twelve cumulative link mixed-effect regression models, two models per component variation. In each model, we predicted the dependent variable (user responses to the preference broadening or user satisfaction question) with a binary independent variable, i.e. whether a movie belongs to a particular variation of the component. The results are summarized in Table 3. It shows that:

(1) Movies that are novel according to either definition of novelty have a positive effect on preference broadening compared with the corresponding non-novel movies.

**Table 2: General characteristics of the dataset. Strict serendipity is the union of all kinds of strict serendipity (movies corresponding to at least one definition of serendipity that requires strict novelty (s_nov)), motivational serendipity is the union of all kinds of motivational serendipity, and serendipity is the union of all kinds of serendipity.**

| Concept | Movies | Users |
|---|---|---|
| All | 2146 | 475 |
| Strictly serendipitous (relevant) (s_ser_rel) | 77 | 61 |
| Strictly serendipitous (find) (s_ser_find) | 181 | 119 |
| Strictly serendipitous (implicit) (s_ser_imp) | 115 | 80 |
| Strictly serendipitous (recommend) (s_ser_rec) | 63 | 50 |
| Strictly serendipitous | 205 | 131 |
| Motivationally serendipitous (relevant) (m_ser_rel) | 91 | 64 |
| Motivationally serendipitous (find) (m_ser_find) | 163 | 101 |
| Motivationally serendipitous (implicit) (m_ser_imp) | 128 | 88 |
| Motivationally serendipitous (recommend) (m_ser_rec) | 71 | 49 |
| Motivationally serendipitous | 218 | 122 |
| Serendipitous | 302 | 173 |

**Table 3: The fixed effects (coefficients) of the twelve cumulative link mixed-effect regression models. Each cell corresponds to a coefficient of an ordinal regression with a single independent variable. Dependent variables are our metrics (broadening or satisfaction), while independent variables are variations of serendipity components (metric ∼ component). Significance codes: "*" < 0.0005.**

| Component | Broadening | Satisfaction |
|---|---|---|
| s_nov | 0.7412* | 0.1259 |
| m_nov | 0.7827* | 0.307 |
| unexp_rel | 0.5972* | -0.9133* |
| unexp_find | 2.1161* | 0.1934 |
| unexp_imp | 1.8698* | -0.03029 |
| unexp_rec | 1.0889* | -0.451 |

(2) Movies that are unexpected according to each definition of unexpectedness have a positive effect on preference broadening compared with the corresponding non-unexpected movies.
(3) unexp_rel movies have a negative effect on user satisfaction compared with the non unexp_rel movies.

All variations of novelty and unexpectedness broaden user preferences (observations 1 and 2), but unexpectedness (relevant) (movies that are unexpected to be relevant) hurts user satisfaction (observation 3).

Next, we conducted direct comparisons between the variations of novelty and unexpectedness. For the unexpectedness component, we ran twelve cumulative link mixed-effect regression models, two models per comparison. We ran each model on a subset of the collected dataset where we included only observations belonging to the two variations that were being compared. In the dataset, we repeated observations belonging to the two variations simultaneously. Table 4 summarizes the results for the variations of the unexpectedness component. It shows that:

(1) unexp_find and unexp_imp movies have positive effects on preference broadening and user satisfaction, when compared with unexp_rel movies.
(2) unexp_find movies have a positive effect on user satisfaction, when compared with unexp_imp movies.

Variations of unexpectedness components turned out to differ in terms of our metrics. Unexpectedness (relevant) broadens user preferences less and results in a lower user satisfaction than unexpectedness (find and implicit) (observation 1), while unexpectedness (find) outperforms unexpectedness (implicit) in terms of user satisfaction (observation 2).

We omitted the results for the variations of novelty, because we did not find any statistically significant results in comparisons between strict and motivational novelty in terms of preference broadening and user satisfaction.

**Table 4: The fixed effects (coefficients) of the twelve cumulative link mixed-effect regression models. Each cell corresponds to a coefficient of an ordinal regression with a single independent variable run on a dataset consisting of instances belonging to variations of unexpectedness indicated in the left column and top row. Dependent variables are our metrics (broadening or satisfaction), while independent variables are variations of unexpectedness indicated in the left column (metric ∼ component). Significance codes: "*" < 0.0005**

| Broadening | | | |
|---|---|---|---|
| Component | unexp_rel | unexp_find | unexp_imp |
| unexp_find | 0.8206* | | |
| unexp_imp | 0.6325* | -0.1343 | |
| unexp_rec | 0.4079 | -0.3932 | -0.1857 |
| **Satisfaction** | | | |
| Component | unexp_rel | unexp_find | unexp_imp |
| unexp_find | 0.6373* | | |
| unexp_imp | 0.4634* | -0.1463* | |
| unexp_rec | 0.1436 | -0.4529 | -0.3270 |

## 5.2 Effects of Serendipity

To address RQ2, we ran sixteen cumulative link mixed-effect regression models, two models per serendipity definition. In each model, the dependent variable corresponds to the metric (preference broadening or satisfaction), while the independent variable is a binary variable, which equals true if the movie is serendipitous according to a particular definition of serendipity and false otherwise.

Table 5 summarizes the results and shows that movies that are serendipitous according to seven definitions of serendipity (s_ser_rel, s_ser_find, s_ser_imp, s_ser_rec, m_ser_find, m_ser_imp,

**Table 5: The fixed effects (coefficients) of the sixteen cumulative link mixed-effect regression models. Each cell corresponds to a coefficient of an ordinal regression with a single independent variable. Dependent variables are the metrics (preference broadening or satisfaction), while the independent variables correspond to whether a movie is serendipitous according to a particular definition (metric ~ serendipity). Significance codes: "*" < 0.0005**

| | s_ser_rel | s_ser_find | s_ser_imp | s_ser_rec | m_ser_rel | m_ser_find | m_ser_imp | m_ser_rec |
|---|---|---|---|---|---|---|---|---|
| Broadening | 0.979* | 1.471* | 1.581* | 1.605* | 0.663 | 1.667* | 1.354* | 1.307* |
| Satisfaction | -0.347 | 0.322 | 0.284 | 0.276 | -0.166 | 0.486 | 0.164 | 0.265 |

m_ser_rec) broaden user preferences more than the corresponding non-serendipitous ones.

To compare different kinds of serendipity with each other, we conducted direct comparisons between them. Similarly, to compare variations of serendipity components, we ran each comparison on an altered dataset, which included only observations of the two kinds of serendipity that were being compared and repeated observations belonging to both kinds simultaneously. Overall, we ran fifty-six cumulative link mixed-effect regression models (twenty-eight models per metric). We omitted the results for preference broadening, as we did not find any statistically significant results. Table 6 demonstrates the fixed effects (coefficients) of the regression models run for user satisfaction. The following observations can be noticed:

- m_ser_find movies are more enjoyable than s_ser_imp movies.
- s_ser_imp movies are more enjoyable than m_ser_rel movies.
- s_ser_rec movies are more enjoyable than m_ser_rec movies.

Motivational serendipity (find) outperforms strict serendipity (implicit), which, in turn, outperforms motivational serendipity (relevant) in terms of user satisfaction (observations 1 and 2). Meanwhile, strict serendipity (recommend) outperforms motivational serendipity (recommend) (observation 3).

## 5.3 Detecting Serendipitous Items

To answer RQ3, we come up with different features of movies and selected the effective subset for detecting serendipitous movies corresponding to the union of the six kinds of serendipity definitions, i.e. all except the two definitions s_ser_rel and m_ser_rel. We excluded these two from the union because (a) these two definitions include unexp_rel, for which we have evidence showing that it hurts user satisfaction and (b) we do not have evidence showing that m_ser_find broadens user preferences more than non m_ser_find. We predicted the union of the six serendipity definitions that broaden user preferences more than their corresponding non-serendipitous items.

To support feature calculation, we first define an average similarity of a movie to a user profile or to the recommendations this user previously received. In this paper, the term *user profile* refers to ratings this user assigned to items in the past. We define the average similarity of a movie to a user profile as follows:

$$sim\_prof_{u,i} = \frac{1}{||I_u||} \sum_{j \in I_u, j \neq i} sim_{i,j} \tag{2}$$

where $I_u$ is the set of movies rated by user $u$, while $sim_{i,j}$ is the similarity between movies $i$ and $j$. The way we calculate similarity depends on the movie representation. For example, to calculate genre similarity, we modeled movies as sets of genres and used the Jaccard similarity. For collaborative similarity, we modeled movies as rating vectors, where each value corresponded to a user rating, and cosine similarity was used. We define the average similarity of a movie to the recommendations a user previously received as follows:

$$sim\_rec_{u,i} = \frac{1}{||R_u||} \sum_{j \in R_u, j \neq i} sim_{i,j} \tag{3}$$

where $R_u$ is the set of the eight last movies recommended to user $u$ by MovieLens. In summary, we came up with the following features:

- Popularity (logpop). We used popularity because it is one of the most common attributes used in studies dedicated to serendipity in recommender systems [19, 20, 30]. We calculated popularity as follows: $logpop_i = ln(U_i)$, where $U_i$ is the number of ratings received by movie $i$ during the last year (2016) in MovieLens. We picked the number of ratings during the last year instead of the overall number of ratings, because many old movies received many ratings if they were released a long time ago. However, these movies were likely to be unfamiliar to the active users in the system. The most famous movies, such as "The Shawshank Redemption", "Toy Story" and "The Matrix" are still among the most popular movies according to our last-year popularity metric.
- Predicted rating (predicted_rating). We used this feature because the expectation of users might be affected by the system's predictions, while they are browsing movie pages (note that in MovieLens, the predicted ratings are displayed along with the movie information). The algorithm that predicts the rating depends on the choice of the user because MovieLens offers several recommendation algorithms, among which item-based collaborative filtering and matrix factorization are used by the majority of the users.
- Release year (year). We picked this attribute because recency of movies might affect users' familiarity with them. Users might be more familiar with recently released movies than the older ones.
- Average tag-based similarity to the user profile (tag_sim_prof). Similarly to popularity, we picked this feature because content-based similarity is commonly considered in the literature [15, 19, 29, 30]. To calculate the average tag-based distance we employed the tagging model, tag genome [28], which is based on tags users assign to movies. We calculated the distance according to Equation 2, where $sim_{i,j}$ is the similarity measure of weighted cosine distance in [28].

**Table 6: The fixed effects (coefficients) of the twenty-eight cumulative link mixed-effect regression models. Each cell corresponds to a coefficient of an ordinal regression with the independent variable run on a dataset consisting of observations belonging to the kinds of serendipity indicated in the left column and the top row. The dependent variable corresponds to user satisfaction, while independent variables correspond to the variations of unexpectedness indicated in the left column (satisfaction ∼ serendipity). Significance codes: "*" < 0.0005**

| Serendipity | s_ser_rel | s_ser_find | s_ser_imp | s_ser_rec | m_ser_rel | m_ser_find | m_ser_imp |
|---|---|---|---|---|---|---|---|
| s_ser_find | 0.570 | | | | | | |
| s_ser_imp | 0.503 | -0.045 | | | | | |
| s_ser_rec | 0.071 | 0.092 | -0.052 | | | | |
| m_ser_rel | 0.185 | -0.536 | -0.445* | -0.520 | | | |
| m_ser_find | 0.733 | 0.047 | 0.140* | 0.063 | 0.522 | | |
| m_ser_imp | 0.514 | -0.244 | -0.091 | 0.017 | 0.274 | -0.309 | |
| m_ser_rec | 1.069 | -0.135 | -0.145 | 0.355* | 0.600 | 0.005 | 0.153 |

- Average tag-based similarity to recommendations the user received from MovieLens (tag_sim_rec). We picked this feature because users' expectation might depend on the recommendations our system generates. We calculated this feature according to Equation 3 using the similarity measure of weighted cosine distance in [28].
- Average genre-based similarity to the user profile (genre_sim_prof). We picked this feature as an additional content-based similarity and calculated it according to Equation 2, where $sim_{i,j}$ is the Jaccard similarity between the sets of genres of the movies $i$ and $j$.
- Average genre-based similarity to recommendations the user received from the system (genre_sim_rec). We picked this feature as an additional content-based similarity and calculated it according to Equation 3 using the Jaccard similarity.
- Average collaborative similarity to user profile (c_sim_prof). We picked this feature because this is a common similarity measure in the literature on serendipity [14, 30]. We calculated this feature according to Equation 2, where $sim_{i,j}$ is the cosine similarity between movie rating vectors $i$ and $j$.
- Average collaborative similarity to recommendations the user received from the system (c_sim_rec). We calculated this feature according to Equation 3 using the cosine similarity.

We detected effective features for predicting serendipitous movies by running a logistic regression model on our dataset. We used the logistic regression model for the sake of interpretability. In our dataset, we labeled each movie based on whether this movie was serendipitous to a user and performed the 10-fold cross validation.

To select effective features for the prediction of serendipity, we employed the forward search strategy, where we iteratively picked features based on the performance of the logistic regression model when gradually adding these features into the model. To compare models, we used the metric: Area Under the ROC Curve (AUC), which is a commonly used for assessing performance of binary classifiers. We also reported AIC (Akaike Information Criterion), which evaluates the quality of a statistical model (the lower the value, the better the model) [3].

Table 7 demonstrates the results of the forward feature selection strategy. We only included the first four features, because further incorporating more features decreases AUC. According to the obtained results, the most effective features for serendipitous movies according to at least one of the six definitions are predicted rating, popularity, the average tag-based similarity to the user profile and the average collaborative similarity to the user profile.

**Table 7: The results of feature selection with logistic regression, where the dependent variable is a binary variable indicating whether a movie belongs to the union of the six kinds of serendipity (excluding s_ser_rel and m_ser_rel).**

| Features | AIC | AUC |
|---|---|---|
| predicted_rating | 1468.628 | 0.609 |
| predicted_rating + logpop | 1464.008 | 0.621 |
| predicted_rating + logpop + tag_sim_prof | 1459.707 | 0.624 |
| predicted_rating + logpop + tag_sim_prof + c_sim_prof | 1459.840 | 0.627 |

**Table 8: The coefficients of the logistic regression model, where the dependent variable is a binary variable indicating whether a movie is serendipitous according to the union of the six definitions, while the four independent variables correspond to the selected features**

| Feature | Parameter | Standard Error |
|---|---|---|
| predicted_rating | 2.954* | 0.624 |
| logpop | -1.223 | 0.392 |
| tag_sim_prof | 0.508 | 0.260 |
| c_sim_prof | -0.816 | 1.057 |

Table 8 shows the coefficients of the final logistic regression model. It shows that movies that are serendipitous according to at least one of the six definitions have higher predicted ratings than corresponding non-serendipitous movies. Other coefficients are not statistically significant after correction, but the model shows a trend that serendipitous movies tend to be less popular compared with non-serendipitous ones.

### 5.4 How Rare Are Serendipitous Items?

According to Table 2, among 2146 movies that users gave their feedback on, 302 (14%) are serendipitous according to at least one definition. The entire database of MovieLens contains 25,650,696 ratings and 15,854,339 (or 61%) of them are higher than 3.5, which suggests that up to 8.5% ($0.14 * 0.61 \approx 0.085$) are serendipitous. Our samples include 437 movies that our system encouraged users to watch, which can be considered as the number of recommendations that users took. This suggests that up to 69% of recommendations provided by our system that users watch are serendipitous according to at least one definition.

The dataset includes 275 movies that correspond to the union of the six definitions of serendipity, which have a positive effect on preference broadening. This suggests that our system contains up to 5.9% of these movies and 47.4% of them among the recommendations. For the smallest kind of serendipity, strict serendipity (recommend), these ratios are 1.8% and 14.4%, while for the largest kind of serendipity, strict serendipity (find), they are 5.1% and 41.4%, respectively.

## 6 DISCUSSION

We reviewed a set of techniques for operationalizing serendipity, finding that different definitions have different effects on preference broadening and user satisfaction, but confirming that in general serendipitous recommendations broaden preferences (usually without hurting satisfaction).

We found that there are sufficient serendipitous items to recommend (particularly across the span of definitions), making it feasible to recommend serendipitous items in contexts where preference broadening would be useful. We do not look explicitly at which contexts may benefit most, but leave that to others.

The results of our study regarding features effective for the detection of serendipitous items mostly correspond to the prior literature. Content-based similarity to a user profile, collaborative similarity to a user profile and popularity have been acknowledged as important features [1, 14, 15, 20, 22, 29, 30]. However, most studies employ popularity and disregard similarity to a user profile in offline evaluations of recommendation algorithms [20, 22, 30].

Surprisingly, our results showed that ratings provided by recommender systems are a good predictor for serendipity. In fact, the higher the rating the more likely an item to be perceived serendipitous. This might suggest that even recommendation algorithms optimized for accuracy assist users to encounter serendipitous items at least within the limitations of our dataset. This contradicts to the common claim that recommender systems narrow users' interests and trap them in filter bubbles [23, 27, 29]. However, the design of our experiment does not allow us to support or reject this claim.

## 7 LIMITATIONS AND FUTURE WORK

Conducting a study of movie recommender system users based on their previously-rated movies has several limitations. First, we were limited in the reasons we could explore for movie performance on our metrics. For example, serendipitous movies might broaden user preferences more than non-serendipitous ones due to other reasons than that these movies are serendipitous. Second, our study is limited in domain to movies. While we hope our results are generalizable at least in related domains, further study is needed to evaluate user impact, even in this domain. In our future work, we are going to design a serendipity-oriented algorithm using the collected dataset and evaluate it in an experimental setting with real users, where we control for serendipity with the novel algorithm.

The specific design of our study had other limitations. We only looked at performance of different kinds of items in terms of preference broadening and user satisfaction, which was based on the literature review. Future work should consider other metrics. We selected only relatively unpopular relevant movies for our survey to increase the chance of asking users about serendipitous movies, which only allowed us to compare unpopular serendipitous movies and unpopular relevant non-serendipitous ones. As a result, our sample is biased, and may not represent average performance. Finally we limited our study to active users (duration of use of at least a month, minimum number of ratings), which may not reflect the experience of one-time or very infrequent users.

## 8 CONCLUSION

In this paper, we conducted a survey asking 475 real users about 2146 movies with questions designed based on different serendipity definitions synthesized from the prior literature. Through this survey, we collected the first dataset that has real user evaluation on the serendipity of the items. We only asked about relevant movies to (i.e. highly rated by) those users and therefore the effects we found in this work are all relative to items that are relevant but not serendipitous. The following research questions are addressed.

*RQ1. What are the effects of various serendipity components on broadening user preferences and user satisfaction?*

We found that all variations of the unexpectedness and novelty components broaden user preferences, but one type of unexpectedness (unexpected to be relevant) hurts user satisfaction. Movies that users found novel and unexpected according to any definition employed in this paper broaden user preferences more than movies users found non-novel and non-unexpected, respectively. Movies that users did not expect to like and be recommended are less enjoyable than movies users expected to like and be recommended (or had no expectations), respectively.

Variations of the unexpectedness component are different in terms of our metrics. Two variations of unexpected movies: (a) movies that users did not expect to find and (b) movies that users thought were different from movies these users usually watch are better than the variation: (c) movies users did not expect to like in terms of both preference broadening and user satisfaction. Meanwhile, movies that users did not expect to find are more enjoyable than the ones that users found different from movies these users usually watch.

*RQ2. What are the effects of different kinds of serendipity on broadening user preferences and user satisfaction?*

We found that serendipitous movies generally broaden user preferences more than non-serendipitous ones, but we did not find any effects of serendipity on user satisfaction. In particular, movies that are serendipitous according to seven definitions of serendipity broaden user preferences more than corresponding non-serendipitous ones.

We also found that different kinds of serendipity differ in terms of user satisfaction. In particular, motivational serendipity (find) outperforms strict serendipity (implicit), which, in turn, outperforms motivational serendipity (relevant), while strict serendipity (recommend) outperforms motivational serendipity (recommend).

*RQ3. What are the effective features for detecting serendipitous movies? What are the value ranges of these features for typical serendipitous movies?*

We found features most important for detecting movies that are serendipitous according to six definitions of serendipity that do not include unexpectedness (relevant), which hurts user satisfaction. These features are predicted ratings, popularity, content-based and collaborative similarity to a user profile. Our results also show that these serendipitous movies have higher predicted ratings than corresponding non-serendipitous ones.

*RQ4. How rare are serendipitous movies among movies rated by the users in a typical collaborative-filtering-based recommender system? To what extent does this kind of system help users find these movies?*

We discovered that in the best case scenario, among movies users rate in a typical movie recommender system, up to 8.5% are serendipitous according to at least one definition, while among movies recommended by the system that users watch, this ratio is up to 69%. We only provide an upper bound estimation due to the bias of our dataset.

## ACKNOWLEDGMENTS

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

We would like to thank F. Maxwell Harper and researchers at the GroupLens team and the MineSocMed project for their feedback and assistance in conducting this research.

## REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On Unexpectedness in Recommender Systems: Or How to Better Expect the Unexpected. *ACM Transactions on Intelligent Systems and Technology* 5, 4 (2014), 1–32.
[2] Alan Agresti and Maria Kateri. 2011. Categorical data analysis. In *International encyclopedia of statistical science*. Springer, 206–208.
[3] Hirotugu Akaike. 1974. A new look at the statistical model identification. *IEEE transactions on automatic control* 19, 6 (1974), 716–723.
[4] Paul André, M.C. Schraefel, Jaime Teevan, and Susan T. Dumais. 2009. Discovery is Never by Chance: Designing for (Un)Serendipity. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*. ACM, New York, NY, USA, 305–314.
[5] Rune Haubo Bojesen Christensen. 2010. ordinal-regression models for ordinal data. *R package version* 22 (2010).
[6] Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Cataldo Musto. 2015. An investigation on the serendipity problem in recommender systems. *Information Processing & Management* 51, 5 (2015), 695 – 717.
[7] Michael D Ekstrand, Daniel Kluver, F Maxwell Harper, and Joseph A Konstan. 2015. Letting users choose recommender algorithms: An experimental study. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 11–18.
[8] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction* 4, 2 (2011), 81–173.
[9] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*. ACM, 257–260.
[10] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22, 1 (2004), 5–53.
[11] Yoshinori Hijikata, Takuya Shimizu, and Shogo Nishida. 2009. Discovery-oriented Collaborative Filtering for Improving User Satisfaction. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*. ACM, 67–76.
[12] Yosef Hochberg and Yoav Benjamini. 1990. More powerful procedures for multiple significance testing. *Statistics in medicine* 9, 7 (1990), 811–818.
[13] Leo Iaquinta, Giovanni Semeraro, Marco de Gemmis, Pasquale Lops, and Piero Molino. 2010. *Can a recommender system induce serendipitous encounters?* InTech.
[14] Marius Kaminskas and Derek Bridge. 2014. 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)*.
[15] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on. Interactive Intelligent Systems* 7, 1, Article 2 (2016), 42 pages.
[16] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A. Konstan, and Paul Schrater. 2015. "I Like to Explore Sometimes": Adapting to Dynamic User Novelty Preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 19–26.
[17] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2016. Challenges of Serendipity in Recommender Systems. In *Proceedings of the 12th International conference on web information systems and technologies.*, Vol. 2. SCITEPRESS, 251–256.
[18] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2017. A Serendipity-Oriented Greedy Algorithm for Recommendations. In *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, Vol. 1. ScitePress, 32–40.
[19] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.
[20] Qiuxia Lu, Tianqi Chen, Weinan Zhang, Diyi Yang, and Yong Yu. 2012. 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. IEEE Computer Society, 258–265.
[21] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. ACM, 1097–1101.
[22] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. 2008. Metrics for Evaluating the Serendipity of Recommendation Lists. In *Annual Conference of the Japanese Society for Artificial Intelligence*.
[23] Eli Pariser. 2011. *The filter bubble: What the Internet is hiding from you*. Penguin UK.
[24] Paul Resnick and Hal R. Varian. 1997. Recommender Systems. *Commun. ACM* 40, 3 (1997), 56–58.
[25] Alan Said, Ben Fields, Brijnesh J. Jain, and Sahin Albayrak. 2013. User-centric Evaluation of a K-furthest Neighbor Collaborative Filtering Recommender Algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. ACM, 1399–1408.
[26] Eugenio Tacchini. 2012. *Serendipitous mentorship in music recommender systems*. Ph.D. Dissertation.
[27] Maria Taramigkou, Efthimios Bothos, Konstantinos Christidis, Dimitris Apostolou, and Gregoris Mentzas. 2013. Escape the Bubble: Guided Exploration of Music Preferences for Serendipity and Novelty. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 335–338.
[28] Jesse Vig, Shilad Sen, and John Riedl. 2012. The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. *ACM Transactions on Interactive Intelligent Systems* 2, 3, Article 13 (2012), 44 pages.
[29] Yuan Cao Zhang, Diarmuid Ô Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: Introducing Serendipity into Music Recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*. ACM, 13–22.
[30] Qianru Zheng, Chi-Kong Chan, and Horace H.S. Ip. 2015. An Unexpectedness-Augmented Utility Model for Making Serendipitous Recommendation. In *Advances in Data Mining: Applications and Theoretical Aspects*. Vol. 9165. Springer International Publishing, 216–230.

# PIII

# CHALLENGES OF SERENDIPITY IN RECOMMENDER SYSTEMS

by

Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang 2016

Proceedings of the 12th International conference on web information systems and technologies.

# Challenges of Serendipity in Recommender Systems

Denis Kotkov, Jari Veijalainen and Shuaiqiang Wang
*University of Jyvaskyla, Dept. of Computer Science and Information Systems, P.O.Box 35, FI-40014, Jyvaskyla, Finland*

Keywords: Relevance, Unexpectedness, Novelty, Serendipity, Recommender Systems, Evaluation Metrics, Challenges.

Abstract: Most recommender systems suggest items similar to a user profile, which results in boring recommendations limited by user preferences indicated in the system. To overcome this problem, recommender systems should suggest serendipitous items, which is a challenging task, as it is unclear what makes items serendipitous to a user and how to measure serendipity. The concept is difficult to investigate, as serendipity includes an emotional dimension and serendipitous encounters are very rare. In this paper, we discuss mentioned challenges, review definitions of serendipity and serendipity-oriented evaluation metrics. The goal of the paper is to guide and inspire future efforts on serendipity in recommender systems.

## 1 INTRODUCTION

With the growth of information on the Internet it becomes difficult to find content interesting to a user. Hopefully, recommender systems are designed to solve this problem. In this paper, the term *recommender system* refers to a software tool that suggests items of use to users (Ricci et al., 2011). 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 (Ricci et al., 2011). For example, an item can be a reference to a movie, a song or even a friend in an online social network.

Recommender systems and search engines are different kinds of systems that aim at satisfying user information needs. Traditionally, a search engine receives a query and, in some cases, a user profile as an input and provides a set of the most suitable items in response (Smyth et al., 2011). In contrast, a recommender system does not receive any query, but a user profile and returns a set of items users would enjoy (Ricci et al., 2011). The term *user profile* refers to actions a user performed with items in the past. A user profile is often represented by ratings a user gave to items.

Recommender systems are widely adopted by different services to increase turnover (Ricci et al., 2011). Meanwhile, users need a recommender system to discover novel and interesting items, as it is demanding to search items manually among the overwhelming number of them (Shani and Gunawardana, 2011; Celma Herrada, 2009).

Most recommendation algorithms are evaluated based on accuracy that indicates how good an algorithm is at offering interesting items regardless of how obvious and familiar to a user the suggestions are (de Gemmis et al., 2015).To achieve high accuracy, recommender systems tend to suggest items similar to a user profile (Tacchini, 2012). As a result, the user receives recommendations only of items similar to items the user rated initially. Accuracy-based algorithms limit the number of items that can be recommended to the user (so-called *overspecialization* problem), which lowers user satisfaction (Celma Herrada, 2009; Tacchini, 2012). To overcome overspecialization problem and broaden user preferences, a recommender system should suggest serendipitous items.

Suggesting serendipitous items is challenging (Foster and Ford, 2003). Currently, there is no consensus on definition of serendipity in recommender systems (Maksai et al., 2015; Iaquinta et al., 2010). It is difficult to investigate serendipity, as the concept includes an emotional dimension (Foster and Ford, 2003) and serendipitous encounters are very rare (André et al., 2009). As different definitions of serendipity have been proposed (Maksai et al., 2015; Iaquinta et al., 2010), it is not clear how to measure serendipity in recommender systems (Murakami et al., 2008; Zhang et al., 2012).

In this paper we are going to discuss mentioned challenges to guide and inspire future efforts on

Kotkov, D., Veijalainen, J. and Wang, S.
Challenges of Serendipity in Recommender Systems.
In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 2*, pages 251-256
ISBN: 978-989-758-186-1

serendipity in recommender systems. We review definitions of serendipity. We also review and classify evaluation metrics to measure serendipity and indicate their advantages and disadvantages.

## 2 CHALLENGES OF SERENDIPITY IN RECOMMENDER SYSTEMS

Suggesting serendipitous items involves certain challenges. We are going to present the most important of them. Designing a serendipity-oriented recommendation algorithm requires to choose suitable objectives. It is therefore necessary to investigate how to assess serendipity in recommender systems, which requires a definition of the concept.

### 2.1 Definition

It is challenging to define what serendipity is in recommender systems, what kind of items are serendipitous and why, since serendipity is a complex concept (Maksai et al., 2015; Iaquinta et al., 2010).

According to the dictionary[1], serendipity is "the faculty of making fortunate discoveries by accident". The term was coined by Horace Walpole in the letter to Sir Horace Mann in 1754. The author described his unexpected discovery by referencing the fairy tale, "The Three Princes of Serendip". Horace Walpole in his letter explained that the princes were "always making discoveries, by accidents and sagacity, of things which they were not in quest of" (Remer, 1965).

One of the examples of serendipity is the discovery of penicillin. On September 3, 1928, Alexander Fleming was sorting petri dishes and noticed a dish with a blob of mold. The mold in the dish killed one type of bacteria, but did not affect another. Later, the active substance from the mold was named penicillin and used to treat a wide range of diseases, such as pneumonia, skin infections or rheumatic fever. The discovery of penicillin can be regarded as serendipitous, as it led to the result positive for the researcher and happened by accident.

To introduce discoveries similar to the discovery of penicillin in recommender systems, it is necessary to define and strictly formalize the concept of serendipity. We therefore reviewed definitions employed in publications on recommender systems.

Corneli et al. investigated serendipity in a computational context including recommender systems and proposed the framework to describe the concept (Corneli et al., 2014). The authors considered an essential key condition, *focus shift*. A focus shift happens when something that initially was uninteresting, neutral or even negative becomes interesting.

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

One of definitions used in recommender systems was employed by Zhang et al.: "Serendipity represents the "unusualness" or "surprise" of recommendations" (Zhang et al., 2012). The definition does not require serendipitous items to be interesting to a user, but surprising.

In contrast, Maksai et al. indicated that serendipitous items must be not only unexpected (surprising), but also useful to a user: "Serendipity is the quality of being both unexpected and useful" (Maksai et al., 2015).

Adamopoulos and Tuzhilin used another definition. The authors mentioned the following components related to serendipity: unexpectedness, novelty and a positive emotional response, which can be regarded as relevance of an item for a user:

> *Serendipity, the most closely related concept to unexpectedness, involves a positive emotional response of the user about a previously unknown (novel) [...] serendipitous recommendations are by definition also novel.* (Adamopoulos and Tuzhilin, 2014).

A similar definition was employed by Iaquinta et al. According to (Iaquinta et al., 2010), serendipitous items are interesting, unexpected and novel to a user:

> *A serendipitous recommendation helps the user to find a surprisingly interesting item that she might not have otherwise discovered (or it would have been really hard to discover). [...] Serendipity cannot happen if the user already knows what is recommended to her, because a serendipitous happening is by definition something new. Thus the lower is the probability that user knows an item, the higher is the probability that a specific item could result in a serendipitous recommendation* (Iaquinta et al., 2010).

Definitions used in (Iaquinta et al., 2010) and (Adamopoulos and Tuzhilin, 2014) seem to correspond to the dictionary definition and the framework proposed by Corneli et al. As a serendipitous item is novel and unexpected, the item can be perceived as uninteresting, at first sight, but eventually the item will be regarded as interesting, which creates a focus shift, a necessary condition for serendipity (Corneli et al., 2014). The definition also corresponds to the dictionary definition, as novel, unexpected and interesting to a user item is likely to be a "fortunate discovery".

Publications dedicated to serendipity in recommender systems do not often elaborate the components of serendipity (Iaquinta et al., 2010; Maksai et al., 2015; Zhang et al., 2012). It is not entirely clear in what sense items should be novel and unexpected to a user.

Kapoor et al. indicated three different definitions of novelty in recommender systems (Kapoor et al., 2015):

1. *Novel to a recommender system item.* A recently added item that users have not yet assessed.
2. *Forgotten item.* A user might forget that she consumed the item some time ago in the past.
3. *Unknown item.* A user has never seen or heard about the item in her life.

In addition, Shani and Gunawardana suggested that we may regard a novel item as one not rated by the target user regardless of whether she is familiar with the item (Shani and Gunawardana, 2011).

Unexpectedness might also have different meanings depending on expectations of a user. A user might expect a recommender system to suggest items similar to her profile, popular among other users or both similar and popular (Kaminskas and Bridge, 2014; Zheng et al., 2015).

#### 2.1.1 Serendipity in a Context

Most recommender systems do not consider any contextual information, such as time, location or mood of a user (Adomavicius and Tuzhilin, 2011). Meanwhile, the context may significantly affect the relevance of items for a user (Adomavicius and Tuzhilin, 2011). An item that was relevant for a user yesterday might not be relevant tomorrow. A context may include any information related to recommendations. For example, a recommender system may consider current weather to suggest a place to visit. Context-aware recommender systems use contextual information to suggest items interesting to a user.

Serendipity depends on a context, as each of its components is context-dependant. An item that was relevant, novel and unexpected to a user in one context might not be perceived the same in another context. The inclusion of a context affects the definition of serendipity. For example, contextual unexpectedness would indicate how unexpected an item is in a given context, which might be different from unexpectedness in general. Serendipity might consist of novelty, unexpectedness and relevance in a given context.

As the context has a very broad definition (Dey, 2001), it is challenging to estimate what contextual information is the most important in a particular situation. For example, weather is an important factor for most outdoor activities, while user mood is important for music suggestion (Kaminskas and Ricci, 2012).

#### 2.1.2 Serendipity in Cross-domain Recommender Systems

Most recommender systems suggest items from a single domain, where the term domain refers to "a set of items that share certain characteristics that are exploited by a particular recommender system" (Fernández-Tobías et al., 2012). These characteristics are items' attributes and users' ratings. Different domains can be represented by movies and books, songs and places, MovieLens[2] movies and Netflix[3] movies (Cantador and Cremonesi, 2014).

Recommender systems that suggest items using multiple domains are called cross-domain recommender systems. Cross-domain recommender systems can use information from several domains, suggest items from different domains or both consider different domains and suggest items from them (Cantador and Cremonesi, 2014). For example, a cross-domain recommender system may take into account movie preferences of a user and places that the user visits to recommend watching a particular movie in a cinema suitable for the user.

Consideration of additional domains affects the definition of serendipity, as cross-domain recommender systems may suggest combinations of items. It is questionable whether in this case items in the recommended combination must be novel and unexpected.

#### 2.1.3 Discussion

According to literature review, to date, there is no consensus on definition of serendipity in recommender systems (Maksai et al., 2015; Iaquinta et al., 2010). We suggest that the definition of serendipity should include combinations of items from different domains and a context, which might encourage researchers to propose serendipity-oriented recommendation algorithm that would be more satisfying to users. For example, suppose, two young travelers walk in a cold rain in a foreign city without much money. A recommendation of a hostel would be obvious in this situation, as the travelers would look for a hostel on their own. A suggestion of sleeping in a local cinema, which would cost less than a hostel, is likely to be serendipitous in that situation. The recommendation

[2]https://movielens.org/
[3]https://www.netflix.com/

Table 1: Notation

| | |
|---|---|
| $I = (i_1, i_2, ..., i_n)$ | the set of items |
| $F = (f_1, f_2, ..., f_z)$ | feature set |
| $i = (f_{i,1}, f_{i,2}, ..., f_{i,z})$ | representation of item $i$ |
| $U = (u_1, u_2, ..., u_n)$ | the set of users |
| $I_u, I_u \subseteq I$ | the set of items rated by user $u$ (user profile) |
| $R_u, R_u \subseteq I$ | the set of items recommended to user $u$ |
| $rel_u(i)$ | 1 if item $i$ relevant for user $u$ and 0 otherwise |

would be even more satisfying to the travelers if the recommender system also suggested the longest and cheapest movie in that cinema and an energetic song to cheer up the travelers.

## 2.2 Emotional Dimension

Relevance of an item for a user might depend on user mood (Kaminskas and Ricci, 2012). This contextual information is difficult to capture without explicitly asking the user. As serendipity is a complex concept, which includes relevance (Iaquinta et al., 2010; Adamopoulos and Tuzhilin, 2014), this concept depends on the current user mood in a higher degree. An emotional dimension makes serendipity unstable and therefore difficult to investigate (Foster and Ford, 2003).

## 2.3 Lack of Serendipitous Encounters

As serendipitous items must be relevant, novel and unexpected to a user, they are rare (André et al., 2009) and valuable. Due to the lack of observations it is difficult to make assumptions regarding serendipity that would be reasonable in most cases.

## 2.4 Evaluation Metrics

We are going to review evaluation metrics that measure serendipity in recommender systems. As different metrics have been proposed (Murakami et al., 2008; Kaminskas and Bridge, 2014; Zhang et al., 2012), the section provides their comparison, including advantages and disadvantages. To review evaluation metrics, we first present notation in table 1.

The following evaluation metrics consider a recommender system with $I$ available items and $U$ users. User $u$ rates or interacts with items $I_u, I_u \subseteq I$. A recommender system suggests $R_u$ items to user $u$. Each item $i, i \in I$ is represented as a vector $i = (f_{i,1}, f_{i,2}, ..., f_{i,z})$ in a multidimensional feature space $F$. For example, a feature can be a genre of a movie on a web-site. If $F = (drama, crime, action)$ then the movie "*The Shawshank Redemption*" can be represented as $i_{Shawshank} = (0.4, 0.4, 0.1)$.

Seeking to measure serendipity of a recommender system, researchers proposed different evaluation metrics. Based on reviewed literature we classify them into three categories: content-based unexpectedness, collaborative unexpectedness and primitive recommender-based serendipity.

### 2.4.1 Content-based Unexpectedness

Content-based unexpectedness metrics are based on attributes of items. These metrics indicate the dissimilarity of suggestions to a user profile.

One of the content-based unexpectedness metrics was proposed by Vargas and Castells (Vargas and Castells, 2011). Later, the metric was adopted by Kaminskas and Bridge to measure unexpectedness (Kaminskas and Bridge, 2014). The authors suggested that serendipity consists of two components: relevance and unexpectedness. Content-based unexpectedness metrics can be used to measure unexpectedness, while accuracy metrics such as root mean square error (RMSE), mean absolute error (MAE) or precision (Ekstrand et al., 2011) can be used to assess relevance. The metric is calculated as follows:

$$unexp_c(i,u) = \frac{1}{|I_u|} \sum_{j \in I_u} 1 - sim(i,j) \qquad (1)$$

where $sim(i,j)$ is any kind of similarity between items $i$ and $j$. For example, it might be content-based cosine distance (Lops et al., 2011).

### 2.4.2 Collaborative Unexpectedness

Collaborative unexpectedness metrics are based on ratings users gave to items. Kaminskas and Bridge proposed a metric that can measure unexpectedness based on user ratings (Kaminskas and Bridge, 2014). User ratings can indicate similarities between items. Items can be considered similar if they are rated by the same set of users. The authors therefore proposed a co-occurrence unexpectedness metric, which is based on normalized point-wise mutual information:

$$unexp_r(i,u) = \frac{1}{|I_u|} \sum_{j \in I_u} -\log_2 \frac{p(i,j)}{p(i)p(j)} / \log_2 p(i,j) \qquad (2)$$

where $p(i)$ is the probability that users have rated item $i$, while $p(i,j)$ is the probability that the same users have rated items $i$ and $j$.

#### 2.4.3 Primitive Recommender-based Serendipity

The metric is based on suggestions generated by a primitive recommender system, which is expected to generate recommendations with low unexpectedness. Originally the metric was proposed by Murakami (Murakami et al., 2008) and later modified (Ge et al., 2010; Adamopoulos and Tuzhilin, 2014). Modification proposed by Adamopoulos and Tuzhilin is calculated as follows:

$$ser_{pm}(u) = \frac{1}{|R_u|} \sum_{i \in (R_u \setminus (E_u \cup PM))} rel_u(i), \qquad (3)$$

where *PM* is a set of items generated by a primitive recommender system, while $E_u$ is a set items that matches interests of user *u*. In the experiments conducted by Adamopoulos and Tuzhilin, the primitive recommender system generated non-personalized recommendations consisting of popular and highly rated items. Meanwhile, $E_u$ contained items similar to what user *u* consumes.

#### 2.4.4 Analysis of the Evaluation Metrics

Content-based and collaborative metrics capture the difference between recommended items and a user profile, but have a disadvantage. These metrics measure unexpectedness separately from relevance. The high score of both metrics can be obtained by suggesting many unexpected irrelevant and expected relevant items that would probably not be serendipitous.

Depending on a primitive recommender system, the metrics based on a primitive recommender system capture item popularity and dissimilarity to a user profile, but also have a disadvantage. Primitive recommender-based metrics are sensitive to a primitive recommender system (Kaminskas and Bridge, 2014). By changing this parameter, one might obtain contradictory results.

Designing a serendipity-oriented algorithm that takes into account a context and combinations of items from different domains requires a corresponding serendipity definition and serendipity metric. An item might be represented by a combination of items from different domains and considered serendipitous, depending on a particular situation. The reviewed metrics disregard a context and additional domains due to the lack of serendipity definitions that consider this information. One of the reasons might be that recommender systems do not usually have the information on the context. Another reason might be the disadvantages of offline evaluation.

Even offline evaluation of only relevance without considering the context or additional domains may not correspond to results of experiments involving real users (Said et al., 2013; Garcin et al., 2014). Offline evaluation may help choose candidate algorithms (Shani and Gunawardana, 2011), but online evaluation is still necessary, especially in assessing serendipity, as serendipitous items are novel by definition (Iaquinta et al., 2010; Adamopoulos and Tuzhilin, 2014) and it is difficult to assess whether a user is familiar with an item without asking her.

## 3 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we discussed challenges of serendipity in recommender systems. Serendipity is challenging to investigate, as it includes an emotional dimension, which is difficult to capture, and serendipitous encounters are very rare, since serendipity is a complex concept that includes other concepts.

According to the reviewed literature, currently there is no consensus on definition of serendipity in recommender systems, which makes it difficult to measure the concept. The reviewed serendipity evaluation metrics can be divided into three categories: content-based unexpectedness, collaborative unexpectedness and primitive recommender-based serendipity. The main disadvantage of content-based and collaborative unexpectedness metrics is that they measure unexpectedness separately from relevance, which might cause mistakes. The main disadvantage of primitive recommender-based serendipity metrics is that they are sensitive to a primitive recommender.

In our future work, we are going to propose a definition of serendipity in recommender systems, develop serendipity metrics and design recommendation algorithms that suggest serendipitous items. We are also planning to conduct experiments using pre-collected datasets and involving real users. We hope that this paper will guide and inspire future research on recommendation algorithms focused on user satisfaction.

## ACKNOWLEDGEMENT

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

## REFERENCES

Adamopoulos, P. and Tuzhilin, A. (2014). On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology*, 5(4):54:1–54:32.

Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer US.

André, P., schraefel, m., Teevan, J., and Dumais, S. T. (2009). Discovery is never by chance: Designing for (un)serendipity. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, pages 305–314, New York, NY, USA. ACM.

Cantador, I. and Cremonesi, P. (2014). Tutorial on cross-domain recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 401–402, New York, NY, USA. ACM.

Celma Herrada, Ò. (2009). *Music recommendation and discovery in the long tail*. PhD thesis, Universitat Pompeu Fabra.

Corneli, J., Pease, A., Colton, S., Jordanous, A., and Guckelsberger, C. (2014). Modelling serendipity in a computational context. *CoRR*, abs/1411.0440.

de Gemmis, M., Lops, P., Semeraro, G., and Musto, C. (2015). An investigation on the serendipity problem in recommender systems. *Information Processing & Management*, 51(5):695 – 717.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173.

Fernández-Tobías, I., Cantador, I., Kaminskas, M., and Ricci, F. (2012). Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*.

Foster, A. and Ford, N. (2003). Serendipity and information seeking: an empirical study. *Journal of Documentation*, 59(3):321–340.

Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., and Huber, A. (2014). Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 169–176, New York, NY, USA. ACM.

Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 257–260, New York, NY, USA. ACM.

Iaquinta, L., Semeraro, G., de Gemmis, M., Lops, P., and Molino, P. (2010). *Can a recommender system induce serendipitous encounters?* INTECH Open Access Publisher.

Kaminskas, M. and Bridge, D. (2014). Measuring surprise in recommender systems. In *Workshop on Recommender Systems Evaluation: Dimensions and Design*.

Kaminskas, M. and Ricci, F. (2012). Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(23):89 – 119.

Kapoor, K., Kumar, V., Terveen, L., Konstan, J. A., and Schrater, P. (2015). "i like to explore sometimes": Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 19–26, New York, NY, USA. ACM.

Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer US.

Maksai, A., Garcin, F., and Faltings, B. (2015). Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 179–186, New York, NY, USA. ACM.

Murakami, T., Mori, K., and Orihara, R. (2008). Metrics for evaluating the serendipity of recommendation lists. In *New Frontiers in Artificial Intelligence*, volume 4914 of *Lecture Notes in Computer Science*, pages 40–46. Springer Berlin Heidelberg.

Remer, T. G. (1965). *Serendipity and the three princes: From the Peregrinaggio of 1557*, page 20. Norman, U. Oklahoma P.

Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*. Springer US.

Said, A., Fields, B., Jain, B. J., and Albayrak, S. (2013). User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 1399–1408, New York, NY, USA. ACM.

Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer US.

Smyth, B., Coyle, M., and Briggs, P. (2011). Communities, collaboration, and recommender systems in personalized web search. In *Recommender Systems Handbook*, pages 579–614. Springer US.

Tacchini, E. (2012). *Serendipitous mentorship in music recommender systems*. PhD thesis, Ph. D. thesis., Computer Science Ph. D. School–Università degli Studi di Milano.

Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, pages 109–116, New York, NY, USA. ACM.

Zhang, Y. C., Séaghdha, D. O., Quercia, D., and Jambor, T. (2012). Auralist: Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 13–22, New York, NY, USA. ACM.

Zheng, Q., Chan, C.-K., and Ip, H. (2015). An unexpectedness-augmented utility model for making serendipitous recommendation. In *Advances in Data Mining: Applications and Theoretical Aspects*, volume 9165 of *Lecture Notes in Computer Science*, pages 216–230. Springer International Publishing.

# PIV

# A SERENDIPITY-ORIENTED GREEDY ALGORITHM FOR RECOMMENDATIONS

by

Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang 2017

Proceedings of the 13rd International conference on web information systems and technologies (WEBIST) (Best Paper Award)

# A Serendipity-Oriented Greedy Algorithm for Recommendations

Denis Kotkov[1], Jari Veijalainen[1] and Shuaiqiang Wang[2*]

[1]*University of Jyvaskyla, Faculty of Information Technology, P.O.Box 35, FI-40014 University of Jyvaskyla, Jyvaskyla, Finland*

[2]*Manchester Business School, The University of Manchester, Manchester, U.K.*

Keywords: Recommender Systems, Learning to Rank, Serendipity, Novelty, Unexpectedness, Algorithms, Evaluation.

Abstract: Most recommender systems suggest items to a user that are popular among all users and similar to items the user usually consumes. As a result, a 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 algorithm, which improves serendipity through feature diversification and helps overcome the overspecialization problem. To evaluate our algorithm and compare it with others, we employ a serendipity metric that captures each component of serendipity, unlike the most common metric.

## 1 INTRODUCTION

Recommender systems are software tools that suggest items of use to users (Ricci et al., 2011; Kotkov et al., 2016a). 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” (Kotkov et al., 2016a). 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 (1) popular, as these items are consumed by many individuals and often of high quality in many domains (Celma Herrada, 2009) and (2) similar to which the user has assigned high ratings, as these items correspond to user’s preferences (Tacchini, 2012; Kotkov et al., 2016a; Kotkov et al., 2016b). 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 (Celma Herrada, 2009) and (2) users often lose interest in using the system when they are offered only items similar to highly rated ones from their profiles (the so-called overspecialization problem) (Tacchini, 2012; Kotkov et al., 2016a; Kotkov et al., 2016b). Here the term *user profile* refers to the set of items rated by the target user, though it might include information, such as name, ID 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 (Zhang et al., 2012), while others suggest that serendipitous items are relevant and unexpected (Maksai et al., 2015). Although there is no agreement on the definition of serendipity (Kotkov et al., 2016b), in this paper, the term *serendipitous items* refers to items relevant, novel and unexpected to a user (Kotkov et al., 2016a; Kotkov et al., 2016b):

- 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 (Kotkov et al., 2016a; Kotkov et al., 2016b). In different scenarios, ways to express preference might vary. For example, we might regard a movie relevant to a user if the user gave it more than 3 stars out of 5 (Zheng et al., 2015; Lu et al., ), while we might regard a song 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 this

[*]The research was conducted while the author was working for the University of Jyvaskyla, Finland

Kotkov, D., Veijalainen, J. and Wang, S.
A Serendipity-Oriented Greedy Algorithm for Recommendations.
DOI: 10.5220/0006232800320040
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 32-40
ISBN: 978-989-758-246-2

relevance otherwise.

- An item is *novel* to a user if the user has not consumed it yet (Kotkov et al., 2016a; Kotkov et al., 2016b). 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 in a recommender system (Kotkov et al., 2016a; Kotkov et al., 2016b; Celma Herrada, 2009). 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 (Kotkov et al., 2016a; Kotkov et al., 2016b).
- An item is *unexpected* to a user if the user does not anticipate this item to be recommended to him/her (Kotkov et al., 2016a; Kotkov et al., 2016b). 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 (Tacchini, 2012; Kotkov et al., 2016a; Kotkov et al., 2016b). Consequently, an item dissimilar to the rated ones is regarded as unexpected (Kotkov et al., 2016a; Kotkov et al., 2016b). The measure of dissimilarity could be based on user ratings or item attributes depending on the application scenario of a recommender system (Kaminskas and Bridge, 2014).

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 (Zhang et al., 2012; Lu et al., ; Kotkov et al., 2016b).

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. We also show that the serendipity metric we use in the experiments includes each of the three components of serendipity, unlike the most common serendipity metric.

Our serendipity-oriented algorithm reranks recommendations provided by an accuracy-oriented algorithm and improves serendipity through feature diversification. The proposed algorithm is based on an existing reranking algorithm and outperforms this algorithm in terms of accuracy and serendipity. Our algorithm also outperforms the state-of-the-art serendipity-oriented algorithms in terms of serendipity and diversity.

The paper has the following contributions:

- We propose a serendipity-oriented recommendation algorithm.

Table 1: Notations.

| | |
|---|---|
| $I = \{i_1, i_2, ..., i_{n_I}\}$ | the set of items |
| $I_u, I_u \subseteq I$ | the set of items rated by user $u$ (user profile) |
| $F = \{f_1, f_2, ..., f_{n_F}\}$ | the set of features |
| $F_i, F_i \subseteq F$ | the set of features of item $i$ |
| $U = \{u_1, u_2, ..., u_{n_U}\}$ | 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 $u$ |
| $r_{u,i}$ | the rating given by user $u$ to item $i$ |
| $\hat{r}_{u,i}$ | the prediction of the rating given by user $u$ to item $i$ |

- We evaluate existing serendipity-oriented recommendation alorithms.

The rest of the paper is organized as follows. Section 2 describes the proposed algorithm. Section 3 is dedicated to experimental setting, while section 4 reports the results of the experiments. Finally, section 5 draws conclusions and indicates future work.

## 2 A SERENDIPITY-ORIENTED GREEDY ALGORITHM

To describe the proposed algorithm, we present the notation in Table 1. Let $I$ be a set of available items and $U$ be a set of users. User $u$ rates or interacts with items $I_u, I_u \subseteq I$. A recommender system suggests $RS_u(n)$ items to user $u$. Each item can have a number of features $F_i = \{f_{i,1}, f_{i,2}, ..., f_{i,n_{F,i}}\}$. The rating user $u$ has gaven to item $i$ is represented by $r_{u,i}$, while the predicted rating is represented by $\hat{r}_{u,i}$.

### 2.1 Description

We propose a serendipity-oriented greedy (SOG) algorithm, which is based on a topic diversification algorithm (TD) (Ziegler et al., 2005). 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 (Castells et al., 2015). According to the classification provided in (Kotkov et al., 2016b), we propose a hybrid reranking algorithm following the post-filtering paradigm and considering unpopularity and dissimilarity.

Algorithm 1 describes the proposed approach. An accuracy-oriented algorithm predicts item ratings $\hat{r}_{u,i}$

```
Input  : $RS_u(n)$: top–n recommendation set,
         $\Theta_F$: damping factor
Output: $Res$: picked item list
$B'$: candidate set,
$\hat{r}_{u,i}$: predicted rating of an item,
$\hat{r}_{u,f}$: predicted rating of a feature;
$Res[0] \leftarrow i$ with max $\hat{r}_{u,i}$;
for $z \leftarrow 1$ to $n$ do
    $B \leftarrow set(Res)$;// set converts a list to a set
    $B' \leftarrow RS_u(n) \backslash B$;
    calculate $c_{u,B,i}, i \in B'$;
    normalize $c_{u,B,i}$, $\hat{r}_{u,f}$ and $\hat{r}_{u,i}, i \in B'$ to $[0,1]$;
    forall the $i \in B'$ do
        calculate $score_{u,i}$
    end
    $Res[z] \leftarrow i$ with max $score_{u,i}$;
end
```

Algorithm 1: Description of SOG.

and generates top–n suggestions $RS_u(n)$ for user $u$. SOG iteratively picks items from set $RS_u(n)$ to fill diversified list *Res*. In each iteration the algorithm generates a candidate set $B'$ which contains top–n recommendations $RS_u(n)$ except picked items from list *Res* (or from set $B$). A candidate item with the highest score is added to diversified list *Res*. The score is calculated as follows:

$$score_{u,i} = (1-\Theta_F) \cdot \hat{r}_{u,i} + \Theta_F \cdot c_{u,B,i}, \qquad (1)$$

$$c_{u,B,i} = d_{u,B} + \Theta_S \cdot (\max_{f \in (F_i \backslash F_u)} (\hat{r}_{u,f}) + unexp_{u,i}), \qquad (2)$$

where $\Theta_S$ is a serendipity weight, while $\Theta_F$ is a damping factor, which is responsible for diversity of reranked recommendation list *Res*. The predicted rating of feature $f$ for user $u$ is represented by $\hat{r}_{u,f}$, $\hat{r}_{u,f} \in [0,1]$. Feature rating indicates how likely a user is to like an item that has a particular feature. As an item might have several features, we select the rating of a feature that is novel and most relevant to a user. If an item does not have any novel features $F_i \backslash F_u = \emptyset$ then $\max_{f \in (F_i \backslash F_u)}(\hat{r}_{u,f}) = 0$. Unexpectedness is based on the number of new features of an item for a user:

$$unexp_{u,i} = \frac{||F_i \backslash F_u||}{||F \backslash F_u||}, \qquad (3)$$

where $F$ corresponds to the set of all features, $F_i$ corresponds to features of item $i$, and $F_u$ corresponds to features of items rated by user $u$. Suppose selected features correspond to movie genres $F = \{comedy, drama, horror, adventure, crime\}$, the movie "The Shawshank Redemption" could be represented as follows $F_{shawshank} = \{drama, crime\}$, while the user might rate comedies and dramas $F_u = \{comedy, drama\}$. For

**user-item matrix**

| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | 5 | 4 | | |
| u2 | 2 | 2 | | 5 |
| u3 | 4 | | 4 | |
| u4 | 4 | 3 | | |
| u5 | 2 | | 5 | 5 |

;

**user-feature matrix**

| | f1 | f2 | f3 |
|---|---|---|---|
| u1 | 4.5 | 5 | |
| u2 | 2 | 2 | 5 |
| u3 | 4 | 4 | |
| u4 | 3.5 | 4 | |
| u5 | 2 | 3.5 | 5 |

;

$F_{i1}=\{f1, f2\}$
$F_{i2}=\{f1\}$
$F_{i3}=\{f2\}$
$F_{i4}=\{f3\}$

Figure 1: An example of user-item and user-feature matrices.

user $u$ the movie "The Shawshank Redemption" has the following unexpectedness: $unexp_{u,shawshank} = \frac{||\{drama,crime\} \backslash \{comedy,drama\}||}{||\{comedy,drama,horror,adventure,crime\} \backslash \{comedy,drama\}||} = \frac{1}{3}$. If the user rates items of all features $F$, we regard the feature that is the least familiar to the user as novel. If the user has not rated any features $F_u = \emptyset$, all the features are regarded as novel.

Dissimilarity of an item to picked items is calculated as follows:

$$d_{i,B} = \frac{1}{||B||} \sum_{j \in B} 1 - sim_{i,j}, \qquad (4)$$

where similarity $sim_{i,j}$ can be any kind of similarity measure. In our experiments we used content-based Jaccard similarity:

$$sim_{i,j} = \frac{||F_i \cap F_j||}{||F_i \cup F_j||}. \qquad (5)$$

To predict feature ratings $\hat{r}_{u,f}$, we apply an accuracy-oriented algorithm to a user-feature matrix. A user-feature matrix is based on user-item matrix, where a rating given by a user to a feature corresponds to the mean rating given to items having this feature by this user:

$$r_{u,f} = \frac{1}{||I_{u,f}||} \sum_{i \in I_{u,f}} r_{u,i}, \qquad (6)$$

where $I_{u,f}$ is a set of items having feature $f$ and rated by user $u$.

Figure 1 demonstrates an example of user-item and user-feature matrices. Suppose users have rated items $i1$, $i2$, $i3$ and $i4$ on the scale from 1 to 5 (user-item matrix). Each item has a number of features. For example, item $i1$ has features $f1$ and $f2$, while item $i2$ only has feature $f1$. User-feature matrix contains feature ratings based on user-item matrix (eq. 6). For example, the rating of feature $f1$ for user $u1$ is 4.5, as items $i1$ and $i2$ have feature $f1$ and the user gave a 5 to item $i1$ and a 4 to item $i2$.

## 2.2 Analysis

Our algorithm considers each component of serendipity:

- Ratings $\hat{r}_{u,i}$ and $\hat{r}_{u,f}$ correspond to relevance.
- $unexp_{u,i}$ corresponds to unexpectedness.
- Novelty is handled implicitly. SOG suggests items with features novel to users, leading to the relative unpopularity of the items, as they have unpopular features.

Although SOG is based on TD, our algorithm has three key differences with respect to TD:

- SOG considers item scores instead of positions of items in lists, which leads to more accurate scores ($score_{u,i}$).
- SOG employs a serendipity weight that controls how likely the algorithm is to suggest an item with a novel feature.
- SOG predicts features a user will like.

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 can be conducted on the client's side of a client-server application).
- Our algorithm has two parameters, which adjust the algorithm according to different requirements. The parameters could be different for each user and be adjusted as the user becomes familiar with the system.

The computational complexity of the algorithm is $O(n^3)$ (excluding pre-calculation), where $n$ is a number of items in input set $RS_u(n)$ (in our experiments $n = 20$).

## 3 EXPERIMENTS

To evaluate existing algorithms and test the proposed serendipity metric, we conducted experiments using two datasets: HetRec (Harper and Konstan, 2015) and 100K MovieLens. The HetRec dataset contains 855,598 ratings given by 2,113 users to 10,197 movies (density 3.97%). The 100K MovieLens (100K ML) dataset contains 100,000 ratings given by 943 users to 1,682 movies (density 6.3%). The HetRec dataset is based on the MovieLens10M dataset published by grouplens[2]. Movies in the HetRec dataset are linked with movies on IMDb[3] and Rotten Tomatoes[4] websites.

In our experimental setting, we hid 20 ratings of each evaluated user and regarded the rest of the ratings as training data. We performed a 5-fold cross-validation. Each evaluated algorithm ranked test items for a particular user based on training data.

This experimental setting was chosen due to the evaluation task. Other settings either let an algorithm rank all the items in the system or a limited number of them assuming that items unknown by a user are irrelevant. This assumption is not suitable for evaluation of serendipity, as serendipitous items are novel by definition (Iaquinta et al., 2010; Adamopoulos and Tuzhilin, 2014; Kotkov et al., 2016a). The experiments were conducted using Lenskit framework (Ekstrand et al., 2011).

### 3.1 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 which ranks items according to generated scores (Zheng et al., 2015). The objective function of the algorithm is the following:

$$min \sum_{u \in U} \sum_{i \in I_u} (r_{u,i} - p_u q_i^T)^2 + \beta(||p_u||^2 + ||q_i||^2), \tag{7}$$

  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.
- **SPR** (serendipitous personalized ranking) is an algorithm based on SVD that maximizes the serendipitous area under the ROC (receiver operating characteristic) curve (Lu et al., ):

$$max \sum_{u \in U} f(u), \tag{8}$$

$$f(u) = \sum_{i \in I_u^+} \sum_{j \in I_u \backslash I_u^+} z_u \cdot \sigma(0, \hat{r}_{u,i} - \hat{r}_{u,j})(||U_j||)^{\alpha}, \tag{9}$$

  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 $\theta$ (in our experiments $\theta = 3$). Normalization term $z_u$ is calculated as follows: $z_u = \frac{1}{||I_u^+|| ||I_u \backslash I_u^+||}$. We used hinge loss function to calculate $\sigma(x)$ and set popularity weight $\alpha$ to 0.5, as the

[2]http://www.grouplens.org

[3]http://www.imdb.com

[4]http://www.rottentomatoes.com

algorithm performs the best with these parameters according to (Lu et al., ).

- **Zheng's** is an algorithm based on SVD that considers observed and unobserved ratings and weights the error with unexpectedness (Zheng et al., 2015):

$$\begin{aligned} min \quad & \sum_{u \in U} \sum_{i \in I_u} (\tilde{r}_{u,i} - p_u q_i^T)^2 \cdot w_{u,i} + \\ & + \beta(||p_u||^2 + ||q_i||^2), \end{aligned} \tag{10}$$

  where $\tilde{r}_{u,i}$ corresponds to observed and unobserved ratings a user $u$ gave to item $i$. The unobseved ratings equal to 0. The weight $w$ is calculated as follows:

$$w_{ui} = \left(1 - \frac{||U_i||}{max_{j \in I}(||U_j||)}\right) + \frac{\sum_{j \in I_u} diff(i,j)}{||I_u||}, \tag{11}$$

  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 follows $diff(i,j) = 1 - \rho_{i,j}$, where similarity $\rho_{i,j}$ corresponds to the Pearson correlation coefficient:

$$\rho_{i,j} = \frac{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in S_{i,j}} (r_{uj} - \overline{r}_u)^2}}, \tag{12}$$

  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 $u$.

- **TD** is a topic diversification algorithm, where dissimilarity corresponds to eq. 4. Similarly to (Ziegler et al., 2005), we set $\Theta_F = 0.9$.

- **SOG** is the proposed serendipity-oriented greedy algorithm ($\Theta_F = 0.9$, $\Theta_S = 0.6$). We set $\Theta_F$ similarly to (Ziegler et al., 2005) and $\Theta_S$ slightly higher than 0.5 to emphasize the difference between SOG and TD. To predict feature ratings we used SVD, which received user-genre matrix. User ratings in the matrix correspond to mean ratings given by a user to items with those genres.

- **SOGBasic** is SOG algorithm without predicting genre ratings ($\hat{r}_{uf} = 0$).

## 3.2 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 four properties of recommender systems: (1) accuracy, as it is a common property (Kotkov et al., 2016b), (2) serendipity, as SPR, Zhengs, SOG and SOGBasic are designed to improve this property (Lu et al., ; Zheng et al., 2015), (3) diversity, as this is one of the objectives of TD, SOG and SOGBasic (Ziegler et al., 2005) and (4) novelty, as SPR, Zhengs, SOG and SOGBasic are designed to improve this property (Lu et al., ; Zheng et al., 2015).

To measure serendipity, we employed two metrics: traditional serendipity metric and our serendipity metric. The traditional serendipity metric disregards unexpectedness of items to a user, while our serendipity metric takes into account each component of serendipity. We provided results for both metrics to demonstrate their difference. Overall, we used five metrics:

- To measure a ranking ability of an algorithm, we use normalized discounted cumulative gain (NDCG), which, in turn, is based on discounted cumulative gain (DCG) (Järvelin and Kekäläinen, 2000):

$$DCG_u@n = rel_u(1) + \sum_{i=2}^{n} \frac{rel_u(i)}{log_2(i)}, \tag{13}$$

  where $rel_u(i)$ indicates relevance of an item with rank $i$ for user $u$, while $n$ indicates the number of top recommendations selected. The NDCG metric is calculated as follows:

$$NDCG_u@n = \frac{DCG_u@n}{IDCG_u@n}, \tag{14}$$

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

- The traditional serendipity metric is based on a primitive recommender system which suggests items known and expected by a user. Evaluated recommendation algorithms are penalized for suggesting items that are irrelevant or generated by a primitive recommender system. Similarly to (Zheng et al., 2015), we used a slight modification of the serendipity metric:

$$SerPop_u@n = \frac{RS_u(n) \backslash PM \cap REL_u}{n}, \tag{15}$$

  where $PM$ is a set of items generated by the primitive recommender system. We selected the 100 most popular items for $PM$ following one of the most common strategies (Zheng et al., 2015; Lu et al., ). Items relevant to user $u$ are represented by $REL_u, REL_u = \{i \in TestSet_u | r_{ui} > \theta\}$, where $TestSet_u$ is a ground truth for user $u$, while $\theta$ is the threshold rating, which in our experiments equals to 3.

- Our serendipity metric is based on the traditional one. Although the traditional serendipity metric successfully captures relevance and novelty of recommendations by setting a threshold for ratings and taking into account the number of ratings assigned to an item, the metric disregards unexpectedness. In this paper, we consider an item unexpected to a user if the item has at least one feature novel to the user e.g. a feature, of which the user has not yet rated an item. We therefore calculate the serendipity metric as follows:
$$Ser_u@n = \frac{RS_u(n) \backslash PM \cap REL_u \cap UNEXP_u}{n}, \quad (16)$$
where $UNEXP_u$ is a set of items that have at least one feature novel to user $u$. In our experiments, a movie with at least one genre, which the user has not rated a movie of is considered unexpected.
- To measure diversity, we employed an intra-list dissimilarity metric (Zheng et al., 2015):
$$Div_u@n = \frac{1}{n \cdot (n-1)} \sum_{i \in RS_u(n)} \sum_{j \neq i \in RS_u(n)} 1 - sim_{i,j}, \quad (17)$$
where similarity $sim_{i,j}$ is based on Jaccard similarity using item sets based on movie genres (eq. 5).
- Novelty is based on the number of ratings assigned to an item (Zhang et al., 2012):
$$nov_i = 1 - \frac{||U_i||}{max_{j \in I}(||U_j||)}. \quad (18)$$

## 4 RESULTS

Table 2 demonstrates performance of baselines measured with different evaluation metrics. The following observations can be observed for both datasets:

1. SOG outperforms TD in terms of accuracy and slightly underperforms it in terms of diversity. For example, the improvement of $NDCG@10$ is 5.3% on the HetRec dataset, while on the 100K ML dataset the improvement is 5.9%. The decrease of $Div@10$ is less than 2%.
2. SOG outperforms other algorithms in terms of our serendipity metric and the state-of-the-art serendipity-oriented algorithms in terms of diversity, while the highest value of traditional serendipity metric belongs to SPR. TD achieves the highest diversity among the presented algorithms.
3. SOG outperforms SOGBasic in terms of our serendipity metric. For example, the improvement of $Ser@10$ is 5.9% on the HetRec dataset, while on the 100K ML dataset the improvement is 10.9%.
4. SOG slightly outperforms SOGBasic in terms of $NDCG@n$ ($< 1\%$) and the traditional serendipity metric $SerPop@n$ ($< 1\%$) and underperforms in terms of $Div@n$ ($< 1\%$ for the HetRec dataset and $1-2\%$ for the 100K ML dataset).
5. Popularity baseline outperforms SOGBasic, SOG and TD in terms of $NDCG@n$, but underperforms all the presented algorithms in terms of serendipity.

Observation 1 indicates that our algorithm improves TD in terms of both serendipity and accuracy, having an insignificant decrease in diversity. TD provides slightly more diverse recommendations than SOG, as these algorithms have different objectives. The main objective of TD is to increase diversity (Ziegler et al., 2005), while SOG is designed not only to diversify recommendations, but also expose more recommendations with novel genres to a user. SOG therefore picks movies less expected and slightly less diverse than TD. Surprisingly, suggesting movies with genres novel to a user increases accuracy, which might be caused by diverse user preferences regarding movies. The improvements of accuracy and serendipity seem to overcompensate for the insignificant decrease of diversity.

Observation 2 suggests that our algorithm provides the most serendipitous recommendations among the presented baselines, which is partly due to $\Theta_F = 0.9$ for our algorithm. This parameter also causes the high diversity of TD. We set $\Theta_F$ the same as (Ziegler et al., 2005) to emphasize the difference between SOG and TD. SPR achieves the highest traditional serendipity due to its objective function (Lu et al., ). This algorithm is designed to suggest relevant unpopular items to users.

The prediction of genres a user is likely to find relevant improves serendipity, according to observation 3. Meanwhile, accuracy, traditional serendipity and diversity remain almost the same, as observation 4 suggests. SOG appears to increase the number of relevant, novel and unexpected movies and supplant some relevant and expected movies from recommendation lists with respect to SOGBasic, as novel and unexpected movies suggested by SOG are more likely to also be relevant to users than those suggested by SOGBasic.

According to observation 5, our algorithm underperforms the non-personalized baseline in terms of $NDCG@n$. Accuracy of POP is generally relatively

Table 2: Performance of algorithms.

| HetRec dataset | | | | 100K ML dataset | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | NDCG@5 | NDCG@10 | NDCG@15 | Algorithm | NDCG@5 | NDCG@10 | NDCG@15 |
| TD | 0,761 | 0,800 | 0,840 | TD | 0,736 | 0,776 | 0,823 |
| SOGBasic | 0,824 | 0,841 | 0,873 | SOGBasic | 0,800 | 0,821 | 0,859 |
| SOG | 0,825 | 0,842 | 0,873 | SOG | 0,801 | 0,822 | 0,859 |
| POP | 0,824 | 0,848 | 0,878 | POP | 0,804 | 0,833 | 0,869 |
| SPR | 0,854 | 0,873 | 0,894 | SPR | 0,821 | 0,839 | 0,868 |
| Zheng's | 0,857 | 0,874 | 0,898 | Zheng's | 0,836 | 0,859 | 0,887 |
| SVD | 0,871 | 0,894 | 0,916 | SVD | 0,844 | 0,868 | 0,897 |
| Algorithm | SerPop@5 | SerPop@10 | SerPop@15 | Algorithm | SerPop@5 | SerPop@10 | SerPop@15 |
| POP | 0,224 | 0,393 | 0,476 | POP | 0,039 | 0,178 | 0,297 |
| Zheng's | 0,323 | 0,440 | 0,497 | Zheng's | 0,215 | 0,284 | 0,328 |
| TD | 0,457 | 0,482 | 0,493 | TD | 0,318 | 0,324 | 0,328 |
| SOGBasic | 0,493 | 0,494 | 0,501 | SOGBasic | 0,332 | 0,331 | 0,333 |
| SOG | 0,493 | 0,495 | 0,501 | SOG | 0,333 | 0,332 | 0,333 |
| SVD | 0,501 | 0,544 | 0,546 | SVD | 0,338 | 0,353 | 0,357 |
| SPR | 0,431 | 0,550 | 0,563 | SPR | 0,255 | 0,373 | 0,394 |
| Algorithm | Ser@5 | Ser@10 | Ser@15 | Algorithm | Ser@5 | Ser@10 | Ser@15 |
| POP | 0,072 | 0,112 | 0,136 | POP | 0,010 | 0,055 | 0,114 |
| Zheng's | 0,100 | 0,122 | 0,135 | Zheng's | 0,061 | 0,094 | 0,127 |
| SVD | 0,159 | 0,156 | 0,151 | SVD | 0,129 | 0,136 | 0,144 |
| SPR | 0,128 | 0,162 | 0,158 | SPR | 0,086 | 0,150 | 0,165 |
| TD | 0,192 | 0,177 | 0,158 | TD | 0,166 | 0,171 | 0,156 |
| SOGBasic | 0,284 | 0,204 | 0,168 | SOGBasic | 0,239 | 0,193 | 0,166 |
| SOG | 0,305 | 0,216 | 0,174 | SOG | 0,278 | 0,214 | 0,174 |
| Algorithm | Div@5 | Div@10 | Div@15 | Algorithm | Div@5 | Div@10 | Div@15 |
| Zheng's | 0,782 | 0,795 | 0,798 | SPR | 0,788 | 0,792 | 0,796 |
| SVD | 0,787 | 0,795 | 0,799 | Zheng's | 0,783 | 0,794 | 0,799 |
| SPR | 0,797 | 0,797 | 0,796 | SVD | 0,787 | 0,795 | 0,800 |
| POP | 0,794 | 0,803 | 0,803 | POP | 0,813 | 0,810 | 0,808 |
| SOG | 0,944 | 0,891 | 0,850 | SOG | 0,938 | 0,891 | 0,853 |
| SOGBasic | 0,948 | 0,893 | 0,851 | SOGBasic | 0,959 | 0,901 | 0,856 |
| TD | 0,952 | 0,894 | 0,852 | TD | 0,964 | 0,903 | 0,857 |

high, as users on average tend to give high ratings to popular movies (Amatriain and Basilico, 2015). However, accuracy in this case is unlikely to reflect user satisfaction, as users are often already familiar with popular movies suggested. Despite being relatively accurate, POP suggests familiar and obvious movies, which is supported by both serendipity metrics. The relatively low accuracy of our algorithm was caused by the high damping factor $\Theta_F$.

## 4.1 Serendipity Weight Analysis

Figure 2 indicates performance of SOG and SOGBasic algorithms with the change of serendipity weight $\Theta_S$ from 0 to 1 with the step of 0.1 (without cross-validation) on the HetRec dataset (on the 100K ML dataset the observations are the same). The two following trends can be observed: (1) serendipity declines with the increase of accuracy, as with the growth of $\Theta_S$ our algorithms tend to suggest more movies that users do not usually rate, and (2) diversity declines with the increase of serendipity, as with the growth of $\Theta_S$ our algorithms tend to suggest more movies of genres novel to the user limiting the number of genres recommended.

As SOG predicts genres a user likes, its $Ser@10$ is slightly higher than that of SOGBasic for the same values of $NDCG@10$. SOG tends to suggest more movies of genres not only novel, but also interesting to a user, which slightly hurts diversity, but improves serendipity.

## 4.2 Qualitative Analysis

Table 3 demonstrates the recommendations provided for a randomly selected user, who rated 25 movies in

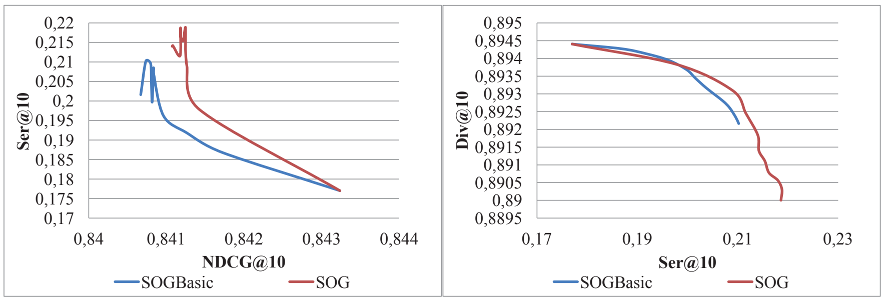


Figure 2: Performance of SOG and SOGBasic algorithms on the Hetrec dataset.

Table 3: Suggestions generated to a random user from the Hetrec dataset (novel genres are in bold).

| User profile | | | |
|---|---|---|---|
| Name | Genres | Rating | *nov* |
| Major League | Comedy | 3.5 | 0.882 |
| The Shawshank Redemption | Drama | 4.5 | 0.137 |
| Stargate | Action Adventure Sci-Fi | 4.5 | 0.572 |
| Robin Hood: Men in Tights | Comedy | 3.5 | 0.684 |
| The Three Musketeers | Action Adventure Comedy | 3.5 | 0.789 |
| SPR recommendations | | | |
| Name | Genres | Rating | *nov* |
| V for Vendetta | **Thriller** | 5 | 0.438 |
| Enemy of the State | Action **Thriller** | 3.5 | 0.620 |
| The Count of Monte Cristo | Drama | 3.5 | 0.787 |
| Free Enterprise | Comedy **Romance** Sci-Fi | 4.5 | 0.989 |
| First Knight | Action Drama **Romance** | 3.5 | 0.962 |
| SOG recommendations | | | |
| Name | Genres | Rating | *nov* |
| V for Vendetta | **Thriller** | 5 | 0.438 |
| The Untouchables | **Thriller Crime** Drama | 4.5 | 0.604 |
| Monsters. Inc. | **Animation Children** Comedy **Fantasy** | 4 | 0.331 |
| Minority Report | Action **Crime Mystery** Sci-Fi **Thriller** | 3 | 0.260 |
| Grease | Comedy **Musical Romance** | 3 | 0.615 |

the HetRec dataset. The algorithms received 5 ratings as the training set and regarded 20 ratings as the test set for the user. We provided recommendations generated by two algorithms: (1) SOG due to high *Ser@n*, and (2) SPR due to high *SerPop@n*.

Although SPR suggested less popular movies than SOG, our algorithm outperformed SPR at overcoming the overspecialization problem, as it introduced more novel genres (8 genres) to the user than SPR (2 genres). Our algorithm also provided a more diversified recommendation list than SPR.

The suggestion of the movie "Monsters Inc." seems to significantly broaden user tastes, as the suggestion is relevant and introduces three new genres to the user. Provided that the movie is serendipitous to the user, it is likely to inspire the user to watch more cartoons in the future.

Analysis of tables 2 and 3 suggests that the traditional serendipity metric captures only novelty and relevance by penalizing algorithms for suggesting popular and irrelevant items, while *Ser@n* takes into account each component of serendipity, which allows assessment of the ability of the algorithm to overcome overspecialization.

## 5 CONCLUSIONS AND FUTURE RESEARCH

We proposed a serendipity-oriented greedy (SOG) recommendation algorithm. We also provided evaluation results of our algorithm and state-of-the-art algorithms using different serendipity metrics.

SOG is based on the topic diversification (TD) algorithm (Ziegler et al., 2005) and improves its accuracy and serendipity for the insignificant price of diversity.

Our serendipity-oriented algorithm outperforms the state-of-the-art serendipity-oriented algorithms in terms of serendipity and diversity, and underperforms them in terms of accuracy.

Unlike the traditional serendipity metric, the serendipity metric we employed in this study captures each component of serendipity. The choice of this metric is supported by qualitative analysis.

In our future research, we intend to further investigate serendipity-oriented algorithms. We will also involve real users to validate our results.

## ACKNOWLEDGEMENTS

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

## REFERENCES

Adamopoulos, P. and Tuzhilin, A. (2014). On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology*, 5(4):1–32.

Amatriain, X. and Basilico, J. (2015). Recommender systems in industry: A netflix case study. In *Recommender Systems Handbook*, pages 385–419. Springer.

Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 881–918. Springer.

Celma Herrada, Ò. (2009). *Music recommendation and discovery in the long tail*. PhD thesis, Universitat Pompeu Fabra.

Ekstrand, M. D., Ludwig, M., Konstan, J. A., and Riedl, J. T. (2011). Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 133–140, New York, NY, USA. ACM.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19.

Iaquinta, L., Semeraro, G., de Gemmis, M., Lops, P., and Molino, P. (2010). *Can a recommender system induce serendipitous encounters?* InTech.

Järvelin, K. and Kekäläinen, J. (2000). 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*, pages 41–48, New York, NY, USA. ACM.

Kaminskas, M. and Bridge, D. (2014). 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)*.

Kotkov, D., Veijalainen, J., and Wang, S. (2016a). Challenges of serendipity in recommender systems. In *Proceedings of the 12th International conference on web information systems and technologies.*, volume 2, pages 251–256. SCITEPRESS.

Kotkov, D., Wang, S., and Veijalainen, J. (2016b). A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192.

Lu, Q., Chen, T., Zhang, W., Yang, D., and 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*, volume 1.

Maksai, A., Garcin, F., and Faltings, B. (2015). Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 179–186, New York, NY, USA. ACM.

Ricci, F., Rokach, L., and Shapira, B. (2011). *Recommender Systems Handbook*, chapter Introduction to Recommender Systems Handbook, pages 1–35. Springer US.

Tacchini, E. (2012). *Serendipitous mentorship in music recommender systems*. PhD thesis.

Zhang, Y. C., Séaghdha, D. O., Quercia, D., and Jambor, T. (2012). Auralist: Introducing serendipity into music recommendation. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pages 13–22, New York, NY, USA. ACM.

Zheng, Q., Chan, C.-K., and Ip, H. H. (2015). An unexpectedness-augmented utility model for making serendipitous recommendation. In *Advances in Data Mining: Applications and Theoretical Aspects*, volume 9165, pages 216–230. Springer International Publishing.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, pages 22–32, New York, NY, USA. ACM.

# PV

# A SERENDIPITY-ORIENTED GREEDY ALGORITHM AND A COMPLETE SERENDIPITY METRIC FOR RECOMMENDATION PURPOSES

by

Denis Kotkov, Jari Veijalainen, Shuaiqiang Wang

Springer Computing (Under review)

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

# A Serendipity-Oriented Greedy Algorithm and a Complete Serendipity Metric for Recommendation Purposes

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

Received: date / Accepted: date

**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. In order to evaluate our algorithm and compare it with other proposed ones, we defined and employed a new serendipity metric that captures all three components of serendipity, in contrast to the commonly used one. The evaluations were performed off-line using two datasets from the movie domain, HetRec and 100K ML. We compared our SOG algorithm with topic diversification, popularity baseline, singular value decomposition, serendipitous personalized ranking and Zheng's algorithms relying on the above two datasets. For the recommendation lists generated by the above algorithms, we calculated accuracy, different serendipity metrics, and diversity. SOG generally outperforms other algorithms, but other algorithms might outperform SOG in

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.)

one measured property, like diversity or serendipity, while being simultaneously inferior in other measured dimensions.

**Keywords** Recommender Systems, Learning to Rank, Serendipity, Novelty, Unexpectedness, Algorithms, Evaluation, HetRec, 100K ML

## 1 Introduction

Recommender systems are software tools that suggest items of use to users [29, 19]. 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" [19]. 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 [7] and (b) similar to those the user has assigned high ratings, as these items correspond to users' preferences [32, 19, 21]. 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 [7] 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) [32, 19, 21, 20]. Here the term *user profile* refers to the unique ID and the set of items rated by the target user [19], 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 [33], while others suggest that serendipitous items are relevant and unexpected [24]. Although there is no agreement on the definition of serendipity [21], in this paper, the term *serendipitous items* refers to items relevant, novel and unexpected to a user [19, 21, 20]:

– 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 [19, 21]. 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 [34, 23], 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 this relevance otherwise.
– An item is *novel* to a user if the user has not consumed it yet [19, 21, 20]. 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 [19,21,20,7]. 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 [19,21].
- An item is *unexpected* to a user if the user does not anticipate this item to be recommended to him/her [19,21]. 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 [32,19,21]. Consequently, an item dissimilar to the rated ones is regarded as unexpected [19,21]. The measure of dissimilarity could be based on user ratings or item attributes depending on the application scenario of a recommender system [17].

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 [33,23,21].

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. We also show how the serendipity metric we use in the experiments includes each of the three components of serendipity, unlike the most common serendipity metric [25] we use for comparison.

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) [35], and outperforms this algorithm in terms of accuracy and serendipity. Our algorithm also outperforms the state-of-the-art serendipity-oriented algorithms in terms of serendipity and diversity.

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.
- Our algorithm has two parameters, which adjust the algorithm according to different requirements.

The paper has the following contributions:

- We propose a serendipity-oriented recommendation algorithm.
- We evaluate existing serendipity-oriented recommendation algorithms.
- We use a novel serendipity metric that takes into account each component of 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 and indicates future work.

## 2 Related works

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

### 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” [28].

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 [24, 21], whereas other researchers suggest that serendipitous items are novel and unexpected [33, 21]. However, the most common definition of serendipity includes all three components: relevance, novelty and unexpectedness [20, 19, 20].

### 2.2 Improving Serendipity

There are three categories for serendipity-oriented algorithms [21]: (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 [21]. 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 [33]. The algorithm consists of the three other algorithms: Basic Auralist, which is responsible for relevance scores, Listener Diversity, which is

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

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

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 [21]. 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) [11] by replacing the user similarity measure with relatedness. It is calculated using random walks with restarts on a user similarity graph [26]. 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 [26]. 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) [9]. 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 [34].

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 [21]. For example, TANGENT recommends items using relevance scores and bridging scores, where both kinds of scores are inferred using a bipartite graph [27]. 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 [27]. Another example of an algorithm that belongs to the category of novel algorithms is random walk with restarts enhanced with knowledge infusion [12]. 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 [12].

### 2.3 Definition of Diversity

Diversity is a property of a recommender system reflecting how dissimilar items are to each other in a recommendation list [6,18]. To measure diversity, researchers often calculate an average pairwise dissimilarity of items in a recommendation list [6,18], 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 [35], and by diversifying the recommendation results, we are likely to suggest an item satisfying a current need of a target user [18]. 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 [5], 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 [22].

### 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 [18,6]. 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 [18,6]. 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 [35].

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 [31]. 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 [8].

## 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 $RS_u(n)$ items to user $u$ (at time $t$). Each item $i$ can have a number of features $F_i =$

Table 1: Notations

| Symbol | Description |
|---|---|
| $I = \{i_1, i_2, ..., i_{n_I}\}$ | the set of items |
| $I_u, I_u \subseteq I$ | the set of items rated by user $u$ (user profile) |
| $F = \{f_1, f_2, ..., f_{n_F}\}$ | the set of features |
| $F_i, F_i \subseteq F$ | the set of features of item $i$ |
| $U = \{u_1, u_2, ..., u_{n_U}\}$ | 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 $u$ |
| $r_{u,i}$ | the rating given by user $u$ to item $i$ |
| $\hat{r}_{u,i}$ | the prediction of the rating given by user $u$ to item $i$ |

$\{f_{i,1}, f_{i,2}, ..., f_{i,n_{F,i}}\}$, that describe item $i$. For example, the movie, *Pulp Fiction* can be described with features: $F_{fiction} = \{Tarantino, crime, violent\}$; the song "I Will Survive" by Gloria Gaynor can be described with features: $F_{survive} = \{famous, 70s, disco\}$. The unique rating user $u$ has given to item $i$ before $t$ is represented by $r_{u,i}$, whereas the predicted rating generated by an algorithm is represented by $\hat{r}_{u,i}$.

## 3.1 Description

We propose a SOG algorithm that is based on a TD algorithm [35]. 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 [6]. According to the classification provided in [21], we propose a hybrid reranking algorithm following the post-filtering paradigm and considering unpopularity and dissimilarity.

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 calculated as follows:

$$score_{u,i} = (1 - \Theta_F) \cdot \hat{r}_{u,i} + \Theta_F \cdot c_{u,B,i} \quad , \tag{1}$$

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

**Algorithm 1:** Description of SOG

$$c_{u,B,i} = d_{u,B} + \Theta_S \cdot (\max_{f \in (F_i \backslash F_u)} (\hat{r}_{u,f}) + unexp_{u,i}) \quad , \tag{2}$$

where $\Theta_S$ is a serendipity weight and $\Theta_F$ is a damping factor that is responsible for diversity of the reranked recommendation list $Res$. The predicted rating of feature $f$ for user $u$ is represented by $\hat{r}_{u,f}$, $\hat{r}_{u,f} \in [0,1]$. Feature rating indicates how likely a user is to like an item that has a particular feature. As an item might have several features, we select the rating of a feature that is unfamiliar and most relevant to a user. If an item does not have any features unfamiliar to a user $F_i \backslash F_u = \emptyset$, then $\max_{f \in (F_i \backslash F_u)}(\hat{r}_{u,f}) = 0$. Unexpectedness is based on the number of new features of an item for a user:

$$unexp_{u,i} = \frac{||F_i \backslash F_u||}{||F \backslash F_u||} \quad , \tag{3}$$

where $F$ refers to the set of all features, $F_i$ refers to the features of item $i$, and $F_u$ refers to all features of items rated by user $u$. Suppose selected features correspond to movie genres $F = \{comedy, drama, horror, adventure, crime\}$, the movie *The Shawshank Redemption* could be represented as follows $F_{shawshank} = \{drama, crime\}$, whereas the user might rate comedies and dramas $F_u = \{comedy, drama\}$. For user $u$, the movie *The Shawshank Redemption* has the following unexpectedness: $unexp_{u,shawshank} = \frac{||\{drama,crime\} \backslash \{comedy,drama\}||}{||\{comedy,drama,horror,adventure,crime\} \backslash \{comedy,drama\}||} = \frac{1}{3}$. If the user rates items of all features $F$, then we regard the feature that is the least familiar to the user as novel. If the user has not rated any features $F_u = \emptyset$, then all the features are regarded as novel.

Dissimilarity of an item $i$ to picked items in $B$ (algorithm 1) is calculated as follows:

$$d_{i,B} = \frac{1}{||B||} \sum_{j \in B} 1 - sim_{i,j} \quad , \tag{4}$$

**user-item matrix**

| | i1 | i2 | i3 | i4 |
|---|---|---|---|---|
| u1 | 5 | 4 | | |
| u2 | 2 | 2 | | 5 |
| u3 | 4 | | 4 | |
| u4 | 4 | 3 | | |
| u5 | 2 | | 5 | 5 |

;

**user-feature matrix**

| | f1 | f2 | f3 |
|---|---|---|---|
| u1 | 4.5 | 5 | |
| u2 | 2 | 2 | 5 |
| u3 | 4 | 4 | |
| u4 | 3.5 | 4 | |
| u5 | 2 | 3.5 | 5 |

;

$F_{i1}=\{f1, f2\}$
$F_{i2}=\{f1\}$
$F_{i3}=\{f2\}$
$F_{i4}=\{f3\}$

Fig. 1: An example of user-item and user-feature matrices

where similarity $sim_{i,j}$ can be any kind of similarity measure. In our experiments we used content-based Jaccard similarity:

$$sim_{i,j} = \frac{||F_i \cap F_j||}{||F_i \cup F_j||} \quad . \tag{5}$$

To predict feature ratings $\hat{r}_{u,f}$, we apply an accuracy-oriented algorithm to a user-feature matrix. A rating given by a user to an item $i$ is considered to concern all features $f_k \in F_i$. A user-feature matrix is based on a user-item matrix, where a rating given by a user to a feature corresponds to the mean rating given by this user to items having this feature:

$$r_{u,f} = \frac{1}{||I_{u,f}||} \sum_{i \in I_{u,f}} r_{u,i} \quad , \tag{6}$$

where $I_{u,f}$ is a set of items that have feature $f$ and that were rated by user $u$.

Figure 1 demonstrates an example of user-item and user-feature matrices. Suppose users have rated items $i1$, $i2$, $i3$ and $i4$ on the scale from 1 to 5 (user-item matrix). Each item has a number of features. For example, item $i1$ has features $f1$ and $f2$, whereas item $i2$ only has feature $f1$. A user-feature matrix contains feature ratings based on a user-item matrix (eq. 6). For example, the rating of feature $f1$ for user $u1$ is 4.5, as items $i1$ and $i2$ have feature $f1$ and the user gave a 5 to item $i1$ and a 4 to item $i2$. Further, because $F_{i4}$ only contains one feature, $f3$, all ratings given by users to item $i4$ also occur directly in the user-feature matrix.

## 3.2 Analysis

Our algorithm considers each component of serendipity:

- Ratings $\hat{r}_{u,i}$ and $\hat{r}_{u,f}$ correspond to relevance.
- $unexp_{u,i}$ corresponds to unexpectedness.
- Novelty is handled implicitly. Some items are rated more often than others. Therefore, features of rarely rated (unpopular) items are rated less often than features of items that are rated often (popular). Although it is possible that an unpopular item has a popular feature, suggesting items with unpopular features is likely to increase the chance of suggesting an

unpopular items compared to suggesting items with popular features. SOG suggests items with features novel to users, which is likely to lead to the relative unpopularity of the suggested items, as items with low numbers of ratings often have rarely rated features.

Although SOG is based on TD [35], our algorithm has three key differences with respect to TD:

- SOG considers item scores instead of positions of items in lists, which leads to more accurate scores ($score_{u,i}$).
- SOG employs a serendipity weight that controls how likely the algorithm is to suggest an item with a novel feature.
- SOG predicts features a user will like.

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 has two parameters, damping factor $\Theta_F$ and serendipity weight $\Theta_S$, which adjust the algorithm according to different requirements. The parameters could be different for each user and be adjusted as the user becomes familiar with the system.

## 3.3 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 $d_{i,B}$ (eq. 4). The overall number of actions can be calculated as follows:

$$\begin{aligned}(n-1)\cdot 1 + (n-2)\cdot 2 + (n-3)\cdot 3 + ... + (n-n)\cdot n =\\ = n\cdot(1+2+3+...+n) - (1^2+2^2+3^2+...+n^2) =\\ = n\cdot n\cdot\frac{1+n}{2} - \frac{n(n+1)(2n+1)}{6} = \frac{n^3-2n^2+n}{6} \quad ;\end{aligned} \tag{7}$$

$$\mathcal{O}\left(\frac{n^3-2n^2+n}{6}\right) = \mathcal{O}(n^3) \quad . \tag{8}$$

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 $RS_u(n)$. In our experiments $n = 20$.

## 4 Experiments

To evaluate existing algorithms and test the proposed serendipity metric, we conducted off-line experiments using two datasets: HetRec [13] and 100K MovieLens. The HetRec dataset contains 855,598 ratings given by 2,113 users to 10,197 movies (density; i.e., the average number of movies rated by a user is $405/10197 = 3.97\%$, because each user issued ca. 405 ratings on average). The 100K MovieLens (100K ML) dataset contains 100,000 ratings given by 943 users to 1,682 movies (density 6.3%). The HetRec dataset is based on the MovieLens10M dataset published by Grouplens[3]. Movies in the HetRec dataset are linked with movies on IMDb[4] and Rotten Tomatoes[5] websites.

In our experimental setting, we performed a five-fold cross-validation, where in each fold one-fifth of users (ca. 420 and 190 users in the HetRec and 100K ML datasets, respectively) had their (randomly picked) 20 ratings in the test set and the rest of their data along with the data of other users belong to the training set. When a current fold included users with 20 ratings or less, all the data of these users belonged to the current test set. Each evaluated algorithm ranked test items for a particular user based on the training data and produced a recommendation list consisting of up to 20 items (depending on the number of ratings a user had in the current test set). Accuracy was based on the ratings from the test set (ground truth) and the position of each item in a recommendation list provided to each evaluated user. Serendipity was calculated based on the number of movies that were serendipitous according to a particular serendipity metric that appeared in the top-N positions of each recommendation list. Diversity was based on the average dissimilarity of movies on top-N positions of each recommendation lists. The results reported below are averaged values of the five cross validation rounds over all users in the two data sets, respectively.

This experimental setting was chosen due to the evaluation task. Other settings either let an algorithm rank all the items in the system or a limited number of them assuming that items unknown to a user are irrelevant. This assumption is not suitable for evaluation of serendipity, as serendipitous items are novel by definition [15, 1, 19]. The experiments were conducted using the Lenskit framework [10].

### 4.1 Baselines

We implemented the following baseline algorithms:

- **POP** ranks items according to the number of ratings each item received in descending order.

[3] http://www.grouplens.org
[4] http://www.imdb.com
[5] http://www.rottentomatoes.com

- **SVD** is a singular value decomposition algorithm that ranks items according to generated scores [34]. The objective function of the algorithm is the following:

$$min \quad \sum_{u \in U} \sum_{i \in I_u} (r_{u,i} - p_u q_i^T)^2 + \beta(||p_u||^2 + ||q_i||^2) \quad , \tag{9}$$

  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.
- **SPR** (serendipitous personalized ranking) is an algorithm based on SVD that maximizes the serendipitous area under the ROC (receiver operating characteristic) curve [23]:

$$max \quad \sum_{u \in U} f(u) \quad , \tag{10}$$

$$f(u) = \sum_{i \in I_u^+} \sum_{j \in I_u \backslash I_u^+} z_u \cdot \sigma(0, \hat{r}_{u,i} - \hat{r}_{u,j})(||U_j||)^{\alpha} \quad , \tag{11}$$

  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 $\theta$ (in our experiments $\theta = 3$). Normalization term $z_u$ is calculated as follows: $z_u = \frac{1}{||I_u^+||||I_u \backslash I_u^+||}$. We used hinge loss function to calculate $\sigma(x)$ and set popularity weight $\alpha$ to 0.5, as the algorithm performs the best with these parameters according to [23].
- **Zheng's** is an algorithm based on SVD that considers observed and unobserved ratings and weights the error with unexpectedness [34]:

$$\begin{aligned} min \quad & \sum_{u \in U} \sum_{i \in I_u} (\tilde{r}_{u,i} - p_u q_i^T)^2 \cdot w_{u,i} + \\ & + \beta(||p_u||^2 + ||q_i||^2) \quad , \end{aligned} \tag{12}$$

  where $\tilde{r}_{u,i}$ corresponds to observed and unobserved ratings a user $u$ gave to item $i$. The unobseved ratings equal to 0. The weight $w$ is calculated as follows:

$$w_{ui} = \left(1 - \frac{||U_i||}{max_{j \in I}(||U_j||)}\right) + \frac{\sum_{j \in I_u} diff(i,j)}{||I_u||} \quad , \tag{13}$$

  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:

$$\rho_{i,j} = \frac{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \in S_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \in S_{i,j}} (r_{uj} - \overline{r}_u)^2}} \quad , \tag{14}$$

  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 $u$.

- **TD** is a topic diversification algorithm, where dissimilarity corresponds to eq. 4. Similarly to [35], we set $\Theta_F = 0.9$.
- **SOG** is the proposed serendipity-oriented greedy algorithm ($\Theta_F = 0.9$, $\Theta_S = 0.6$). We set $\Theta_F$ similarly to [35] and $\Theta_S$ slightly higher than 0.5 to emphasize the difference between SOG and TD. To predict feature ratings we used SVD, which received a user-genre matrix. User ratings in the matrix correspond to mean ratings given by a user to items with those genres.
- **SOGBasic** is the SOG algorithm without predicting genre ratings, which means $\hat{r}_{uf} = 0$ in eq. 2.

### 4.2 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 four properties of recommender systems: (a) accuracy, as it is a common property [21]; (b) serendipity, as SPR, Zheng's, SOG and SOGBasic are designed to improve this property [23,34]; (c) diversity, as this is one of the objectives of TD, SOG and SOGBasic [35]; and (d) novelty, as SPR, Zheng's, SOG and SOGBasic are designed to improve this property [23,34].

To measure serendipity, we employed two metrics: traditional serendipity metric and our serendipity metric. The traditional serendipity metric disregards unexpectedness of items to a user, whereas our serendipity metric takes into account each component of serendipity. We provided results for both metrics to demonstrate their difference. Overall, we used five metrics:

- To measure a ranking ability of an algorithm, we use normalized discounted cumulative gain (NDCG), which, in turn, is based on discounted cumulative gain (DCG) [16]:

$$DCG_u@n = rel_u(1) + \sum_{i=2}^{n} \frac{rel_u(i)}{log_2(pos(i))} \quad , \tag{15}$$

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:

$$NDCG_u@n = \frac{DCG_u@n}{IDCG_u@n} \quad , \tag{16}$$

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

– The traditional serendipity metric is based on a primitive recommender system that suggests items known and expected by a user [25]. Evaluated recommendation algorithms are penalized for suggesting items that are irrelevant or generated by a primitive recommender system. Similarly to [34], we used a slight modification of the serendipity metric:

$$SerPop_u@n = \frac{||(RS_u(n) \backslash PM) \cap REL_u||}{n} \quad , \tag{17}$$

where $PM$ is a set of items generated by the primitive recommender system. We selected the 100 most popular items for $PM$ following one of the most common strategies [34,23]. Items relevant to user $u$ are represented by $REL_u = \{i \in TestSet_u | r_{u,i} > \theta\}$, where $TestSet_u$ is a ground truth for user $u$, while $\theta$ is the threshold rating, which in our experiments equals 3.

– Our serendipity metric is based on the traditional one. Although the traditional serendipity metric successfully captures relevance and novelty of recommendations by setting a threshold for ratings and taking into account the number of ratings assigned to an item, the metric disregards unexpectedness. In this paper, we consider an item unexpected to a user if the item has at least one feature unfamiliar to the user e.g., a feature, of which the user has not yet rated an item. We therefore calculate the serendipity metric as follows:

$$Ser_u@n = \frac{||(RS_u(n) \backslash PM) \cap REL_u \cap UNEXP_u||}{n} \quad , \tag{18}$$

where $UNEXP_u$ is a set of items that have at least one feature novel to user $u$. In our experiments, a movie with at least one genre with which the user has not rated any movie is considered unexpected. We use this feature space as an example. A serendipity metric can also be based on other feature spaces, such as movie directors, casts or descriptions.

– To measure diversity, we employed an intra-list dissimilarity metric [34]:

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

where similarity $sim_{i,j}$ is based on Jaccard similarity using item sets based on movie genres (eq. 5).

– Novelty is based on the number of ratings assigned to an item [33]:

$$nov_i = 1 - \frac{||U_i||}{max_{j \in I}(||U_j||)} \quad . \tag{20}$$

## 5 Results

Table 2 demonstrates performance of baselines measured with different evaluation metrics. We evaluated performance of the algorithms only for recommendation lists of lengths 5, 10 and 15, as diversity and serendipity metrics

Table 2: Performance of algorithms

| HetRec dataset | | | | 100K ML dataset | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | NDCG@5 | NDCG@10 | NDCG@15 | Algorithm | NDCG@5 | NDCG@10 | NDCG@15 |
| TD | 0,761 | 0,800 | 0,840 | TD | 0,736 | 0,776 | 0,823 |
| SOGBasic | 0,824 | 0,841 | 0,873 | SOGBasic | 0,800 | 0,821 | 0,859 |
| SOG | 0,825 | 0,842 | 0,873 | SOG | 0,801 | 0,822 | 0,859 |
| POP | 0,824 | 0,848 | 0,878 | POP | 0,804 | 0,833 | 0,869 |
| SPR | 0,854 | 0,873 | 0,894 | SPR | 0,821 | 0,839 | 0,868 |
| Zheng's | 0,857 | 0,874 | 0,898 | Zheng's | 0,836 | 0,859 | 0,887 |
| SVD | 0,871 | 0,894 | 0,916 | SVD | 0,844 | 0,868 | 0,897 |
| Algorithm | SerPop@5 | SerPop@10 | SerPop@15 | Algorithm | SerPop@5 | SerPop@10 | SerPop@15 |
| POP | 0,224 | 0,393 | 0,476 | POP | 0,039 | 0,178 | 0,297 |
| Zheng's | 0,323 | 0,440 | 0,497 | Zheng's | 0,215 | 0,284 | 0,328 |
| TD | 0,457 | 0,482 | 0,493 | TD | 0,318 | 0,324 | 0,328 |
| SOGBasic | 0,493 | 0,494 | 0,501 | SOGBasic | 0,332 | 0,331 | 0,333 |
| SOG | 0,493 | 0,495 | 0,501 | SOG | 0,333 | 0,332 | 0,333 |
| SVD | 0,501 | 0,544 | 0,546 | SVD | 0,338 | 0,353 | 0,357 |
| SPR | 0,431 | 0,550 | 0,563 | SPR | 0,255 | 0,373 | 0,394 |
| Algorithm | Ser@5 | Ser@10 | Ser@15 | Algorithm | Ser@5 | Ser@10 | Ser@15 |
| POP | 0,072 | 0,112 | 0,136 | POP | 0,010 | 0,055 | 0,114 |
| Zheng's | 0,100 | 0,122 | 0,135 | Zheng's | 0,061 | 0,094 | 0,127 |
| SVD | 0,159 | 0,156 | 0,151 | SVD | 0,129 | 0,136 | 0,144 |
| SPR | 0,128 | 0,162 | 0,158 | SPR | 0,086 | 0,150 | 0,165 |
| TD | 0,192 | 0,177 | 0,158 | TD | 0,166 | 0,171 | 0,156 |
| SOGBasic | 0,284 | 0,204 | 0,168 | SOGBasic | 0,239 | 0,193 | 0,166 |
| SOG | 0,305 | 0,216 | 0,174 | SOG | 0,278 | 0,214 | 0,174 |
| Algorithm | Div@5 | Div@10 | Div@15 | Algorithm | Div@5 | Div@10 | Div@15 |
| Zheng's | 0,782 | 0,795 | 0,798 | SPR | 0,788 | 0,792 | 0,796 |
| SVD | 0,787 | 0,795 | 0,799 | Zheng's | 0,783 | 0,794 | 0,799 |
| SPR | 0,797 | 0,797 | 0,796 | SVD | 0,787 | 0,795 | 0,800 |
| POP | 0,794 | 0,803 | 0,803 | POP | 0,813 | 0,810 | 0,808 |
| SOG | 0,944 | 0,891 | 0,850 | SOG | 0,938 | 0,891 | 0,853 |
| SOGBasic | 0,948 | 0,893 | 0,851 | SOGBasic | 0,959 | 0,901 | 0,856 |
| TD | 0,952 | 0,894 | 0,852 | TD | 0,964 | 0,903 | 0,857 |

disregard the order of items in top-N recommendations. The following observations can be observed for both datasets:

1. SOG outperforms TD in terms of accuracy and slightly underperforms it in terms of diversity. For example, the improvement of $NDCG@10$ is 5.3% on the HetRec dataset, whereas on the 100K ML dataset the improvement is 5.9%. The decrease of $Div@10$ is less than 2%.
2. SOG outperforms other algorithms in terms of our serendipity metric and the state-of-the-art serendipity-oriented algorithms in terms of diversity, whereas the highest value of traditional serendipity metric belongs to SPR. TD achieves the highest diversity among the presented algorithms.
3. SOG outperforms SOGBasic in terms of our serendipity metric. For example, the improvement of $Ser@10$ is 5.9% on the HetRec dataset, whereas on the 100K ML dataset the improvement is 10.9%.
4. SOG slightly outperforms SOGBasic in terms of $NDCG@n$ ($< 1\%$) and the traditional serendipity metric $SerPop@n$ ($< 1\%$) and underperforms in terms of $Div@n$ ($< 1\%$ for the HetRec dataset and $1 - 2\%$ for the 100K ML dataset).
5. Popularity baseline outperforms SOGBasic, SOG and TD in terms of $NDCG@n$, but underperforms all the presented algorithms in terms of serendipity.

Observation 1 indicates that our algorithm improves TD in terms of both serendipity and accuracy, having an insignificant decrease in diversity. TD provides slightly more diverse recommendations than SOG, as these algorithms have different objectives. The main objective of TD is to increase diversity [35], whereas SOG is designed not only to diversify recommendations, but also to expose more recommendations with novel genres to a user. SOG therefore picks movies that are less expected and slightly less diverse than TD. Surprisingly, suggesting movies with genres that are novel to a user increases accuracy, which might be caused by diverse user preferences regarding movies. The improvements of accuracy and serendipity seem to overcompensate for the insignificant decrease of diversity.

Observation 2 suggests that our algorithm provides the most serendipitous recommendations among the presented baselines, which is partly due to $\Theta_F = 0.9$ for our algorithm. This parameter also causes the high diversity of TD. We set $\Theta_F$ to the same value as [35] did to emphasize the difference between SOG and TD. SPR achieves the highest traditional serendipity due to its objective function [23]. This algorithm is designed to suggest relevant unpopular items to users.

The prediction of genres a user is likely to find relevant improves serendipity, according to observation 3. Meanwhile, accuracy, traditional serendipity and diversity remain almost the same, as observation 4 suggests. SOG appears to increase the number of relevant, novel and unexpected movies and supplant some relevant and expected movies from recommendation lists with respect to SOGBasic. This is because novel and unexpected movies suggested by SOG are more likely to also be relevant to users than those suggested by SOGBasic.

According to observation 5, our algorithm underperforms the non-personalized baseline in terms of $NDCG@n$. Accuracy of POP is generally relatively high, as users on average tend to give high ratings to popular movies [4]. However, accuracy in this case is unlikely to reflect user satisfaction, as users are often already familiar with popular movies suggested. Despite being relatively accurate, POP suggests familiar and obvious movies, which is supported by both serendipity metrics. The relatively low accuracy of our algorithm was caused by the high damping factor $\Theta_F$.

## 5.1 Serendipity Weight Analysis

Figure 2 indicates performance of SOG and SOGBasic algorithms with the change of serendipity weight $\Theta_S$ from 0 to 1 with the step of 0.1 (without cross-validation) on the HetRec dataset (on the 100K ML dataset the observations are the same). The two following trends can be observed: (1) serendipity declines with the increase of accuracy, as with the growth of $\Theta_S$ our algorithms tend to suggest more movies that users do not usually rate, and (2) diversity declines with the increase of serendipity, as with the growth of $\Theta_S$ our algorithms tend to suggest more movies of genres novel to the user limiting the number of genres recommended.

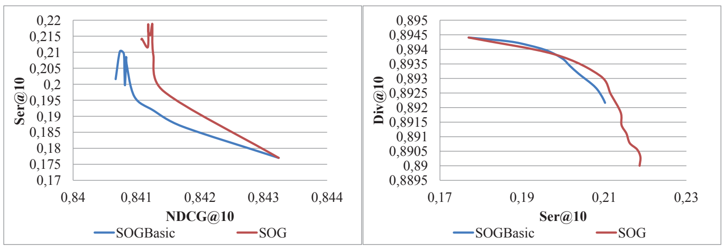


Fig. 2: Performance of SOG and SOGBasic algorithms on the Hetrec dataset

Table 3: Suggestions generated to a random user from the Hetrec dataset (novel genres are in bold)

| User profile | | | |
|---|---|---|---|
| Name | Genres | Rating | *nov* |
| *Major League* | Comedy | 3.5 | 0.882 |
| *The Shawshank Redemption* | Drama | 4.5 | 0.137 |
| *Stargate* | Action Adventure Sci-Fi | 4.5 | 0.572 |
| *Robin Hood: Men in Tights* | Comedy | 3.5 | 0.684 |
| *The Three Musketeers* | Action Adventure Comedy | 3.5 | 0.789 |
| SPR recommendations | | | |
| Name | Genres | Rating | *nov* |
| *V for Vendetta* | **Thriller** | 5 | 0.438 |
| *Enemy of the State* | Action **Thriller** | 3.5 | 0.620 |
| *The Count of Monte Cristo* | Drama | 3.5 | 0.787 |
| *Free Enterprise* | Comedy **Romance** Sci-Fi | 4.5 | 0.989 |
| *First Knight* | Action Drama **Romance** | 3.5 | 0.962 |
| SOG recommendations | | | |
| Name | Genres | Rating | *nov* |
| *V for Vendetta* | **Thriller** | 5 | 0.438 |
| *The Untouchables* | **Thriller Crime** Drama | 4.5 | 0.604 |
| *Monsters. Inc.* | **Animation Children** Comedy **Fantasy** | 4 | 0.331 |
| *Minority Report* | Action **Crime Mystery** Sci-Fi **Thriller** | 3 | 0.260 |
| *Grease* | Comedy **Musical Romance** | 3 | 0.615 |

As SOG predicts genres a user likes, its $Ser@10$ is slightly higher than that of SOGBasic for the same values of $NDCG@10$. SOG tends to suggest more movies of genres not only novel, but also interesting to a user, which slightly hurts diversity, but improves serendipity.

## 5.2 Qualitative Analysis

Table 3 demonstrates the recommendations provided for a randomly selected user, who rated 25 movies in the HetRec dataset. The algorithms received 5 ratings as the training set and regarded 20 ratings as the test set for the user. We provided recommendations generated by two algorithms: (1) SOG due to high $Ser@n$, and (2) SPR due to high $SerPop@n$.

Although SPR suggested less popular movies than SOG, our algorithm outperformed SPR at overcoming the overspecialization problem, as it introduced more novel genres (8 genres) to the user than SPR (2 genres). Our algorithm also provided a more diversified recommendation list than SPR.

The suggestion of the movie *Monsters Inc.* seems to significantly broaden user tastes, as the suggestion is relevant and introduces three new genres to the user. Provided that the movie is serendipitous to the user, it is likely to inspire the user to watch more cartoons in the future.

Analysis of tables 2 and 3 suggests that the traditional serendipity metric captures only novelty and relevance by penalizing algorithms for suggesting popular and irrelevant items, whereas $Ser@n$ takes into account each component of serendipity, which allows assessment of the ability of the algorithm to overcome overspecialization.

## 6 Discussion

A universal algorithm achieving the highest values in all the metrics and suitable for all the tasks a user might need to complete (such as finding good items, finding a credible recommender system and just browsing [30]) does not seem to exist. However, some algorithms might solve particular recommendation tasks better than others.

SOG is flexible by design. It can be adjusted by each user or for each task or by a system for each user. In fact, future recommender systems should not only personalize the content, but also the way of producing recommendations [3]. A recent study shows that different users need different settings [14].

This study has a number of limitations. First, we only evaluated our algorithm on pre-collected datasets. Second, our data comes only from the movie domain. Third, to measure serendipity, we only considered a genre feature space, whereas other feature spaces, such as casts, directors or keywords might be important.

## 7 Conclusions and future research

We proposed a SOG recommendation algorithm. We also provided evaluation results of our algorithm and state-of-the-art algorithms using different serendipity metrics.

SOG is based on the TD algorithm [35] and improves its accuracy and serendipity for the insignificant price of diversity.

Our serendipity-oriented algorithm outperforms the state-of-the-art serendipity-oriented algorithms in terms of serendipity and diversity, and underperforms them in terms of accuracy.

Unlike the traditional serendipity metric, the serendipity metric we employed in this study captures each component of serendipity. The choice of this metric is supported by qualitative analysis.

In our future research, we intend to further investigate serendipity-oriented algorithms. We will also involve real users to validate our results.

## 8 ACKNOWLEDGEMENT

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

### 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. Amatriain, X.: Past, present, and future of recommender systems: An industry perspective. In: Proceedings of the 21st International Conference on Intelligent User Interfaces, pp. 1–1. ACM (2016)
4. Amatriain, X., Basilico, J.: Recommender systems in industry: A netflix case study. In: Recommender Systems Handbook, pp. 385–419. Springer (2015)
5. BellogíN, A., Cantador, I., Castells, P.: A comparative study of heterogeneous item recommendations in social systems. Information Sciences **221**, 142–169 (2013)
6. Castells, P., Hurley, N.J., Vargas, S.: Novelty and diversity in recommender systems. In: Recommender Systems Handbook, pp. 881–918. Springer (2015)
7. Celma Herrada, Ò.: Music recommendation and discovery in the long tail. Ph.D. thesis, Universitat Pompeu Fabra (2009)
8. 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)
9. 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
10. 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)
11. 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
12. 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
13. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems **5**(4), 19:1–19:19 (2015)
14. Harper, F.M., Xu, F., Kaur, H., Condiff, K., Chang, S., Terveen, L.: Putting users in control of their recommendations. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 3–10. ACM (2015)
15. Iaquinta, L., Semeraro, G., de Gemmis, M., Lops, P., Molino, P.: Can a recommender system induce serendipitous encounters? InTech (2010)
16. 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)
17. 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)

18. 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)
19. 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)
20. 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)
21. Kotkov, D., Wang, S., Veijalainen, J.: A survey of serendipity in recommender systems. Knowledge-Based Systems **111**, 180–192 (2016)
22. 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)
23. 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)
24. 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)
25. Murakami, T., Mori, K., Orihara, R.: Metrics for evaluating the serendipity of recommendation lists. In: JSAI. Springer Berlin Heidelberg (2008)
26. 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
27. 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
28. Remer, T.G.: Serendipity and the three princes: From the Peregrinaggio of 1557, p. 20. Norman, U. Oklahoma P (1965)
29. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook, chap. Introduction to Recommender Systems Handbook, pp. 1–35. Springer US (2011)
30. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Recommender systems handbook, pp. 1–34. Springer (2015)
31. 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)
32. Tacchini, E.: Serendipitous mentorship in music recommender systems. Ph.D. thesis (2012)
33. 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)
34. 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)
35. 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)

# PVI

## CROSS-SOCIAL NETWORK COLLABORATIVE RECOMMENDATION

by

Aleksandr Farseev, Denis Kotkov, Alexander Semenov, Jari Veijalainen, Tat-Seng Chua 2015

Proceedings of the 2015 ACM conference on Web science

# Cross-Social Network Collaborative Recommendation

Aleksandr Farseev*, Denis Kotkov**, Alexander Semenov**,
Jari Veijalainen** and Tat-Seng Chua*
farseev@u.nus.edu; deigkotk@student.jyu.fi; {alexander.v.semenov,
jari.veijalainen}@jyu.fi; chuats@nus.edu.sg
* National University of Singapore, School of Computing, COM 1, 13 Computing Drive, Singapore;
** University of Jyvaskyla, Dept. of Comput. Sci. & Inf. Syst., , P.O.Box 35, FI-40014, Finland

## ABSTRACT

Online social networks have become an essential part of our daily life, and an increasing number of users are using multiple online social networks simultaneously. We hypothesize that the integration of data from multiple social networks could boost the performance of recommender systems. In our study, we perform cross-social network collaborative recommendation and show that fusing multi-source data enables us to achieve higher recommendation performance as compared to various single-source baselines.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

## General Terms

Algorithms, Experimentation, Performance

## 1. INTRODUCTION

The fast expanding multi-modal social media data serves as an important resource to perform relevant recommendation and comprehensive user profiling in many application domains. In particular, venue category recommendation (e.g. restaurant, museum, or park) is an important task in tourism and advertisement for suggesting interesting venues near users' current location. At the same time, most internet-active adults prefer to use multiple social services simultaneously to satisfy their different information needs[1], and thus, interests of such users can be better inferred from different perspectives using multiple online social networks (OSNs).

Up to now, only a few studies investigated multi-source data processing, while the usefulness of the multi-source data integration for venue recommendation remains unclear. For example, according to [1, 4], multi-source data integration may help to achieve higher recommendation performance. However, there has not been much research done on multi-source multi-modal recommender systems, while most existing works focus on either multi-source or multi-modal data.

In this paper, we investigate multi-source and multi-modal recommender systems and report the results of an initial experimental study. We seek to address the following research question: *is it possible to improve the recommendation performance by incorporating multi-source multi-modal data?*

To address this question, we employed a subset of NUS-MSS [2] dataset that includes user-generated content posted by the same users in three popular OSNs: Twitter, Foursquare and Instagram. We made the assumption that the social media data from the same user on different social network presents users activities in different perspectives, which are correlated to overall user profile. We therefore built a recommender system that exploits the multi-source multi-modal data and suggests categories of venues visited and unvisited by a particular user. Since location recommendation is especially useful when it is based on the user's current location, we recommend venue categories. It allows an individual to immediately discover the newly suggested places. The task can also be considered as user profiling for content and location-based recommendation. Furthermore, we also address the source integration problem and evaluate our proposed approach against single-source baselines.

## 2. MULTI-SOURCE DATA

To address the research question, we employed a subset of NUS-MSS dataset [2] by considering only those users who performed activities on each of the following three OSNs: Twitter, Foursquare and Instagram, from 10 July till 29 September 2014 in Singapore region. The resulting dataset used for the experimentation contains 3,058,833 tweets, 81,755 check-ins and 87,672 Instagram posts, generated by 4,172 users.

### 2.1 Feature extraction

We model the users as vectors in m-dimensional feature space: $u_{i,F} = (f_{i,1}, f_{i,2}, ..., f_{i,m})$, where $f_k$ is the $k$th feature for user $u_{i,f}$ using features space $F$. Overall, we leverage 4 types of features: Linguistic Inquiry and Word Count (LIWC) from Twitter, Latent Dirichlet Allocation (LDA) from Twitter, Instagram image concepts and Foursquare venue categories[2]. Each user is, thus, represented by 4 feature vectors:

- $u_{i,LIWC} = (liwc_{i,1}, liwc_{i,2}, ..., liwc_{i,70})$, where $liwc_{i,k}$ is a normalized LIWC feature;
- $u_{i,LDA} = (lda_{i,1}, lda_{i,2}, ..., lda_{i,50})$, where $lda_{i,k}$ is the probability that the tweets of user $u_i$ are about topic $k$;

[1]According Pew Research Internet Project's (www.pewinternet.org) Social Media Update 2013

[2]Detailed features description is given in [2]

*WebSci'15* June 28 - July 01, 2015, Oxford, United Kingdom
©2015 ACM. ISBN 978-1-4503-3672-7/15/06...$15.00
DOI: http://dx.doi.org/10.1145/2786451.2786504.

- $u_{i,IN} = (in_{i,1}, in_{i,2}, ..., in_{i,1000})$, where $in_{i,k}$ is a normalized number of pictures posted by user $i$ with a concept $in_k$;
- $u_{i,SQ} = (sq_{i,1}, sq_{i,2}, ..., sq_{i,546})$, where $sq_{i,k}$ is a normalized number of the times user $i$ visited venue category $sq_k$.

## 3. RECOMMENDATION APPROACHES

We recommended venue categories based on each data source independently and based on the fusion of data from multiple sources.

### 3.1 Single-source recommendation

To recommend venue categories, we implemented user-based collaborative filtering. In the case of single-source recommendation, the list of suggested categories is sorted according to the ratings of items. Rating of each item for each user is calculated as follows:

$$\hat{r}(u_{i,F}, j) = \frac{\sum_{k \in U, k \neq i} sq_{k,j} \cdot cos(u_{i,F}, u_{k,F})}{\sum_{k \in U, k \neq i} cos(u_{i,F}, u_{k,F})}, \tag{1}$$

where $\hat{r}(u_{i,F}, j)$ is the rating calculated using feature vector $u_{i,F}$ such as $u_{i,LDA}$, $u_{i,LIWC}$, $u_{i,SQ}$ or $u_{i,IN}$; $u_{i,F}$ is a target user that receives a recommendation list with item $j$; $sq_{k,j}$ is the weight of item $j$ for user $k$; and $cos(u_{i,F}, u_{k,F})$ is the *cosine similarity measure* between users $i$ and $k$.

### 3.2 Multi-source recommendation

We performed different fusion approaches at the different stages of collaborative filtering.

As a simple baseline, we first employed an **early fusion** approach [3] to fuse multi-source data, where features derived from each source were concatenated into a single feature vector:

$$\begin{aligned} u'_i = (&sq_{i,1}, ..., sq_{i,546}, lda_{i,1}, ..., lda_{i,50}, \\ &liwc_{i,1}, ..., liwc_{i,70}, in_{i,1}, ..., in_{i,1000}). \end{aligned} \tag{2}$$

Seeking to boost the recommendation performance, we developed a new **late fusion re-ranking** approach. We linearly combined the outputs from different sources, where the weight of each source is learned based on a stochastic hill climbing with random restart (SHCR) optimization algorithm. Each source is assigned a real-valued weight of between 0 to 1 and the rank of *ith* item in the final recommendation output is computed as follows:

$$Rank_f(item_i) = \frac{1}{n} \sum_{s=1}^{n} \frac{w_s}{Rank_s(item_i)}, \tag{3}$$

where $Rank_s(item_i)$ is the rank of *ith* item in recommendation list for source $s$; $w_s$ corresponds to the weight of the source $s$; $n$ is a total number of sources (in our case, $n = 4$). The venue categories in final recommendation list are sorted in increasing order according to their rank. We optimize the multi-source recommendation performance (measured by $F - measure@10$) in 1000 SHCR iterations that gives a good chance to find reasonable source weights.

## 4. EXPERIMENTAL RESULTS

In our experiments, the recommender system suggests a sorted number of categories to each user. We trained the model using the whole dataset and evaluated based on 736 users who have checked-in in at least 20 categories in the training set and 8 categories in the test set.

To measure the recommendation performance we use $F - measure@K = \frac{2 \cdot P@K \cdot R@K}{P@K + R@K}$, where $P@K$ and $R@K$ are precision and recall at $K$, respectively, and $K$ indicates the number of selected items from the top of the recommendation list.

Figure 1 demonstrates that multi-source multi-modal data fusion helps to improve the recommendation performance. Specifically, the proposed late fusion re-ranking approach outperforms all the baselines starting from $K > 6$.

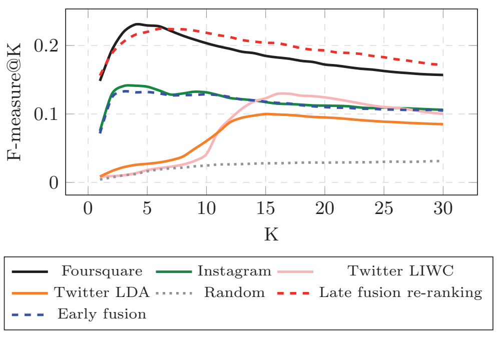


**Figure 1: The evaluation of recommendation performance in terms of F-measure@K**

The lower recommendation performance of multi-source approach for $K < 6$ could be explained by the significant noise level in Twitter data, while for $K > 9$ Twitter features are able to archive better generalization ability of the proposed model.

Another observation is the failure of the early fused model to improve the recommendation performance as compared to single source baselines, where the shape of the performance curve is similar to that of the Instagram approach. The possible reason is the unbalanced sparsity of different feature vectors, since features were derived from different multi-modal sources.

## 5. CONCLUSION

In this study, we investigated the impact of multi-modal data from different social media sources on the recommendation performance. Based on the NUS-MSS dataset [2], we incorporated multi-source multi-modal data and compared its performance with single-source baselines. Our results indicated that the fusion of multi-source multi-modal data is able to boost the recommendation performance significantly.

Our future work includes the development of new efficient source fusion solutions. Also we plan to work on new feature types, data completion techniques and multi-source recommendation models.

## 6. ACKNOWLEDGMENTS

The research in MineSocMed project was partially funded by the Academy of Finland, Grant #268078.

The research is also supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

The Microsoft Windows Azure Cloud and Microsoft MSDN subscription were provided by "MB-Guide"[3] project and as part of Microsoft BizSpark program.

## 7. REFERENCES

[1] F. Abel, E. Herder, G.-J. Houben, N. Henze, and D. Krause. Cross-system user modeling and personalization on the social web. *User Model. User-Adap. Inter.*, 23(2-3):169–209, 2013.

[2] A. Farseev, N. Liqiang, M. Akbari, and T.-S. Chua. Harvesting multiple sources for user profile learning: a big data study. In *ICMR*, 2015.

[3] P. Winoto and T. Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26(3):209–225, 2008.

[4] J. J.-C. Ying, E. H.-C. Lu, W.-N. Kuo, and V. S. Tseng. Urban Point-of-interest Recommendation by Mining User Check-in Behaviors. In *SIGKDD Workshop on Urban Computing*, 2012.

[3]mb-guide.com

# PVII

## CROSS-DOMAIN RECOMMENDATIONS WITH OVERLAPPING ITEMS

by

Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen 2016

Proceedings of the 12th International conference on web information systems and technologies

# Cross-Domain Recommendations with Overlapping Items

Denis Kotkov, Shuaiqiang Wang and Jari Veijalainen
*University of Jyvaskyla, Dept. of Computer Science and Information Systems,*
*P.O.Box 35, FI-40014 Jyvaskyla, Finland*

Keywords: Recommender Systems, Cross-Domain Recommendations, Collaborative Filtering, Content-Based Filtering, Data Collection.

Abstract: In recent years, there has been an increasing interest in cross-domain recommender systems. However, most existing works focus on the situation when only users or users and items overlap in different domains. In this paper, we investigate whether the source domain can boost the recommendation performance in the target domain when *only items overlap*. Due to the lack of publicly available datasets, we collect a dataset from two domains related to music, involving both the users' rating scores and the description of the items. We then conduct experiments using collaborative filtering and content-based filtering approaches for validation purpose. According to our experimental results, the source domain can improve the recommendation performance in the target domain when only items overlap. However, the improvement decreases with the growth of non-overlapping items in different domains.

## 1 INTRODUCTION

Recommender systems use past user behavior to suggest items interesting to users (Ricci et al., 2011). 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.

The majority of recommender systems suggest items based on a single domain. In this paper, the term *domain* refers to "a set of items that share certain characteristics that are exploited by a particular recommender system" (Fernández-Tobías et al., 2012). These characteristics are items' attributes and users' ratings.

However, the single domain recommender systems often suffer from data sparsity and cold start problems. In order to overcome these limitations it is possible to consider data from different domains. Recommender systems that take advantage of multiple domains are called *cross-domain recommender systems* (Fernández-Tobías et al., 2012; Cantador and Cremonesi, 2014).

In this paper, we consider a *cross-domain recommendation task* (Cantador and Cremonesi, 2014), that requires one *target domain* and at least one *source domain*. The former refers to the domain that suggested items are picked from, and similarly the latter refers to the additional domain that contains auxiliary information.

Cross-domain recommender systems can be classified based on domain levels (Cantador and Cremonesi, 2014):

- attribute level - items can be assigned to different domains according to their descriptions. One may contain jazz music, while another may consist of pop audio recordings;
- type level - items may have different types, but share common attributes. Movie and book domains have common genres, such as drama, comedy and horror, while movies and books have different types;
- item level - items from different domains may have completely different attributes and types. Songs and books might not share any common attributes;
- system level - items may belong to different recommender systems, have the same type and share many common attributes. For example, movies from IMDb[1] and MovieLens[2] may belong to different domains.

Depending on whether overlapping occurs in the set of users or items (Cremonesi et al., 2011), there

[1]http://www.imdb.com/
[2]https://movielens.org/

are four situations that enable cross-domain recommendations: a) no overlap between items and users, b) user sets of different domains overlap, c) item sets overlap, and d) item and user sets overlap.

In this work, we investigate whether the source domain improves the recommendation performance in the target domain on system level in the situation when only items overlap. The idea behind the paper is as follows. Traditional cross-domain recommender systems utilize overlapping users to discover additional interests of users, leading to the improvement of the recommendation diversity. When the items overlap, the source domain lets detect more accurate similarities between items, which should positively result in recommendation performance in the target domain.

Due to the lack of publicly available datasets for cross-domain recommender systems with overlapping items (Berkovsky et al., 2008; Kille et al., 2013), we collected data from Vkontakte[3] (VK) – Russian online social network (OSN) and Last.fm[4] (FM) – music recommender service. We then matched VK and FM audio recordings and developed the cross-domain recommender system that suggests VK recordings to VK users based on data from both domains. Each audio recording is represented by its metadata excluding the actual audio file. VK recordings thus represent the target domain, while the source domain consists of FM recordings. VK and FM recordings share titles and artists, but have different user ratings and other attributes.

In order to address the research question and illustrate the potential of additional data, we chose simple but popular recommendation algorithms to conduct experiments for validation: collaborative filtering based on users' ratings and content-based filtering based on the descriptions of the items.

Our results indicate that the source domain can improve the recommendation performance in the target domain. Furthermore, with the growth of non-overlapping items in different domains, the improvement of recommendation performance decreases. This paper thus has the following contributions:

- we initially investigate the cross-domain recommendation problem in the situation when only items overlap;
- we collect a novel dataset to conduct the experiments for addressing the research question.

The paper might be useful in real life scenarios. For example, according to our results, the performance of a recommender system lacking user ratings to achieve an acceptable performance can be improved using ratings collected from another recommender system that suggests items of the same type. However, the performance might decrease if the recommender systems do not have enough overlapping items.

The rest of the paper is organized as follows. Section 2 overviews related works. Section 3 describes the datasets used to conduct experiments. Section 4 is dedicated to recommendation approaches, while section 5 describes conducted experiments. Finally, section 6 draws final concussions.

## 2 RELATED WORKS

Most existing approaches consider additional information about users to boost the recommendation performance. For example, one of the first studies dedicated to cross-domain recommender systems investigated the effectiveness of source domains with overlapping users (Winoto and Tang, 2008). In the experiment, undergraduates from a local university were asked to rate items from different domains, such as movies, songs and books. The authors measured recommendation performance in different domain combinations and concluded that source domains decrease the recommendation performance, but may improve the diversity of recommendations.

In contrast, other studies demonstrated that source domains can boost the recommendation performance in the target domain in situations when users or both users and items overlap. For example, Sang demonstrated the feasibility of utilizing the source domain. The study was conducted on a dataset collected from Twitter[5] and YouTube[6]. The author established relationships between items from different domains using topics (Sang, 2014). Similarly to Sang, Shapira et al. also linked items from different domains, where 95 participants rated movies and allowed the researches to collect data from their Facebook pages. The results suggested that additional domains improve the recommendation performance (Shapira et al., 2013). Another study with positive results was conducted by Abel et al. The dataset contained information related to the same users from 7 different OSNs (Abel et al., 2013). Sahebi et al. demonstrated the usefulness of recommendations based on additional domains to overcome cold start problem (Sahebi and Brusilovsky, 2013).

Most works on cross-domain recommender systems focus on the situation when users or both users

[3]http://vk.com/

[4]http://last.fm/

[5]https://twitter.com/

[6]https://www.youtube.com/

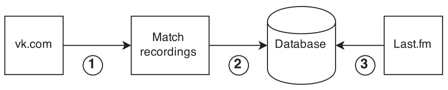


Figure 1: Data collection chart.

and items of several domains overlap (Cantador and Cremonesi, 2014). However, to the best of our knowledge, the efforts on the impact of source domains on the target domain with only overlapping items involving a real cross-domain dataset is very limited.

## 3 DATASETS

Due to the lack of publicly available datasets for cross-domain recommender systems with overlapping items (Berkovsky et al., 2008; Kille et al., 2013) we collected data from VK and FM. The construction of the dataset included three phases (Figure 1): 1) VK recordings collection, 2) duplicates matching, and 3) FM recordings collection.

### 3.1 VK Recordings Collection

The VK interface provides the functionality to add favorite recordings to users' pages. By generating random user ids we collected disclosed VK users' favorite audio recordings using VK API. Our VK dataset consists of $97,737$ ($76,177$ unique) audio recordings added by 864 users.

Each VK user is allowed to share any audio or video recording. The interface of the OSN provides the functionality to add favorite recordings to the users page. VK users are allowed not only to add favorite audio recordings to their pages, but also to rename them. The dataset thus contains a noticeable number of duplicates with different names. To assess this number we randomly selected 100 VK recordings and manually split them into three categories:

- correct names - the name of the recording is correctly written without any grammatical mistakes or redundant symbols;
- misspelled names - the name is guessable, even if the name of the recording is replaced with the combination of artist and recording name or lyrics;
- meaningless names – the name does not contain any information about the recording. For example, "unknown" artist and "The song" recording.

Out of 100 randomly selected recordings we detected 14 misspelled and 2 meaningless names. The example can be seen from table 1.

Table 1: Examples of recordings.

| Artist name | Recording name |
|---|---|
| Correct names | |
| Beyonce | Halo |
| Madonna | Frozen |
| Misspelled | |
| Alice DJ | Alice DJ - Better of Alone.mp3 |
| Reamonn | Oh, tonight you kill me with your smile |
| ● Lady Gaga | Christmas Tree |
| Meaningless | |
| Unknown | classic |
| Unknown | party |

### 3.2 Duplicates Matching

In order to match misspelled recordings, we developed a duplicate matching algorithm that detects duplicates based on recordings' names, mp3 links and durations. The algorithm compares recordings' names based on Levenshtein distance and the number of common words excluding stop words.

We then removed some popular meaningless recordings such as "Unknown", "1" or "01", because they represent different recordings and do not indicate users' preferences. Furthermore, some users assign wrong popular artists' names to the recordings. To restrict the growth of this kind of mistakes, the matching algorithm considers artists of the duplicate recordings to be different. By using the presented matching approach, the number of unique recordings decreased from $76,177$ to $68,699$.

### 3.3 FM Recordings Collection

In order to utilize the source domain we collected FM recordings that correspond to $48,917$ selected VK recordings that were added by at least two users or users that have testing data. Each FM recording contains descriptions such as FM tags added by FM users. FM tags indicate additional information such as genre, language or mood. Overall, we collected $10,962$ overlapping FM recordings and $20,214$ ($2,783$ unique) FM tags.

It is also possible to obtain FM users who like a certain recording (top fans). For each FM recording, we collected FM users who like at least one more FM recording from our dataset according to the distribution of VK users among those recordings. In fact, some unpopular FM recordings are missing top fans. We thus collected $17,062$ FM users, where $7,083$ of them like at least two recordings from our database.

In this work, we constructed three datasets. Each of them includes the collected FM data and different parts of the VK data:

- 0% - the dataset contains only overlapping recordings rated by VK and FM users;
- 50% - the dataset contains overlapping recordings and the half of randomly selected VK recordings that do not correspond to FM recordings;
- 100% - the dataset contains all collected VK and FM recordings.

The statistics of the datasets are presented in table 2. The number of VK users varies in different dataset, due to the lack of ratings after removing non-overlaping VK recordings.

## 4 RECOMMENDATION APPROACHES

In order to emphasize the importance of additional data we implemented simple, but popular collaborative filtering and content-based filtering algorithms.

### 4.1 Item-based Collaborative Filtering

Each recording is represented as a vector in the multidimensional feature space, where each feature is a user's choice. VK recording is represented as follows: $i_j^{vk} = (u_{1,j}^{vk}, u_{2,j}^{vk}, ..., u_{n,j}^{vk})$, where $u_{k,j}^{vk}$ equals to 1 if VK user $k$ picks VK recording $j$ and 0 otherwise. The representation changes if we consider the FM users: $i_j^{vk,fm} = (u_{1,j}^{vk}, u_{2,j}^{vk}, ..., u_{n,j}^{vk}, u_{1,j}^{fm}, u_{2,j}^{fm}, ..., u_{n,j}^{fm})$.

In order to rank items in the suggested list we use sum of similarities of recordings (Ekstrand et al., 2011):

$$score(u_k^{vk}, i_j^{vk}) = \sum_{i_h^{vk} \in I_{u_k^{vk}}} sim(i_j^{vk}, i_h^{vk}), \qquad (1)$$

where $I_{u_k^{vk}}$ is the set of items picked by $u_k^{vk}$ user. We use conditional probability as similarity measure (Ekstrand et al., 2011):

$$p(i_j, i_h) = \frac{Freq(i_j \wedge i_h)}{Freq(i_j) \cdot Freq(i_h)^{\alpha}}, \qquad (2)$$

where $Freq(i_j)$ is the number of users that liked item $i_j$, while $Freq(i_j \wedge i_h)$ is the number of users that liked both items $i_j$ and $i_h$. The parameter $\alpha$ is a demping factor to decrese the similarity for popular items. In our experiments $\alpha = 1$.

It is worth mentioning that item vectors based on FM users contain remarkably more dimensions than vectors based on VK users. In order to alleviate the problem we compared recordings using the following rule:

$$sim(i_j, i_h) = \begin{cases} p(i_j^{vk}, i_h^{vk}), & \exists i_j^{vk} \wedge \exists i_h^{vk} \wedge \\ & (\nexists i_j^{fm} \vee \nexists i_h^{fm}) \\ p(i_j^{fm}, i_h^{fm}), & \exists i_j^{fm} \wedge \exists i_h^{fm} \wedge \\ & (\nexists i_j^{vk} \vee \nexists i_h^{vk}) \\ p(i_j^{vk,fm}, i_h^{vk,fm}), & \exists i_j^{vk} \wedge \exists i_h^{vk} \wedge \\ & \exists i_j^{fm} \wedge \exists i_h^{fm} \end{cases}. \qquad (3)$$

We compare items in each pair using only domains that contain users' ratings for both items.

### 4.2 Content-based Filtering

In a content-based approach similarly to an item-based approach each recording is represented as a vector, but each dimension corresponds to an attribute of the item. In our case, these attributes are VK FM artists and FM tags. It is worth mentioning that FM and VK artists correspond to each other.

An audio recording thus is represented as follows: $i_j^a = (a_{1,j}, a_{2,j}, ..., a_{d,j})$, where $a_{k,j}$ equals to 1 if the recording $i_j$ is performed by the artist $a_k$ and 0 otherwise. The user then can be represented similarly: $u_j^a = (a_{1,j}, a_{2,j}, ..., a_{d,j})$, where $a_{k,j}$ equals to 1 if user $k$ picks the recording perfromed by the artist $a_k$ and 0 otherwise.

The representation chages if we consider FM tags: $i_j^t = (w_{1,j}, w_{2,j}, ..., w_{q,j})$, where $w_{k,j}$ corresponds to the term frequencyinverse document frequency (Lops et al., 2011). The user vector then is denoted as follows: $u_j^t = (t_{1,j}, t_{2,j}, ..., t_{q,j})$, where $t_{k,j}$ is a number of recodings that have tag $t_k$ and are picked by user $u_j$.

The recommender system compares audio recordings' vectors and a user vector using cosine similarity (Ekstrand et al., 2011). First, the suggested list is sorted according to the similarity based on artists. Second, list fragments that consist of items with the same artists' similarity are sorted according to the FM tag similarity.

## 5 EXPERIMENTS

In this section, we conduct experiments to demonstrate whether the source domain improves the recommendation performance in the target domain when only items overlap.

### 5.1 Evaluation Metrics

We used precision@K, recall@K, mean average precision (MAP) and normalized discounted cumulative gain (NDCG) to evaluate our approaches (Zhao,

Table 2: The Statistics of the Datasets.

| | 0% | | 50% | | 100% | |
|---|---|---|---|---|---|---|
| | VK | FM | VK | FM | VK | FM |
| Users | 661 | 7,083 | 850 | 7,083 | 864 | 7,083 |
| Ratings | 14,207 | 40,782 | 62,435 | 40,782 | 96,737 | 40,782 |
| Items | 4,605 | 4,605 | 39,831 | 4,605 | 68,699 | 4,605 |
| Artists | 1,986 | 1,986 | 19,930 | 1,986 | 31,861 | 1,986 |
| Tags | - | 20,167 | - | 20,167 | - | 20,167 |

2013), as these metrics are the most popular in information retrieval. Precision@K, recall@K and mean average precision (MAP) are used to assess quality of recommended lists with binary relevance. Binary relevance requires each item to be relevant or irrelevant for a particular user. As in our case a user can only indicate relevance of a recording by adding it to her page, we regarded added recordings as equally relevant for a user. We regarded the rest recordings as irrelevant. Precision@K and recall@K for a specific user are calculated as follows:

$$P_u@K = \frac{r_u(K)}{K}, \tag{4}$$

$$R_u@K = \frac{r_u(K)}{r_u}, \tag{5}$$

where $r_u(K)$ is the number of items relevant for user $u$ in the first $K$ results, while $r_u$ indicates the number of relevant items in the list recommended to user $u$. Overall precision@K and recall@K are average values.

$$Precision@K = \frac{1}{||U||}\sum_{u \in U} P_u@K, \tag{6}$$

$$Recall@K = \frac{1}{||U||}\sum_{u \in U} R_u@K, \tag{7}$$

where $U$ is a set of evaluated users. MAP then can be calculated in the following way:

$$MAP = \frac{1}{||U||}\sum_{u \in U}\frac{1}{r_u}\left(\sum_{i=1}^{h} r_{u,i} \cdot P_u@i\right), \tag{8}$$

where $r_{u,i} = 1$ if an item at position $i$ in the recommended list is relevant for user $u$ and $r_{u,i} = 0$ otherwise. In our experiments $h$ was set to 30.

We also evaluated our approaches using NDCG (Järvelin and Kekäläinen, 2002). The metric considers positions of items in recommended lists and multiple levels of relevance. We employed NDCG to measure the quality of recommendations with binary relevance. The metric is calculated as follows:

$$NDCG_u@K = Z_n \cdot \sum_{i=1}^{K} \begin{cases} 2^{r_{u,i}} - 1, & i = 1 \\ \frac{2^{r_{u,i}} - 1}{log_2(i)}, & i > 1 \end{cases}, \tag{9}$$

$$NDCG@K = \frac{1}{||U||}\sum_{u \in U} NDCG_u@K, \tag{10}$$

where $Z_n$ is the normalization constant.

## 5.2 Results

Following the datasets sampling strategy in (Ekstrand et al., 2011), we split each of our datasets into training and test datasets. In particular, we selected 40% of the users who rated the most VK recordings, and then chose 30% of their ratings as the testing dataset. We then regarded the rest ratings as the training dataset.

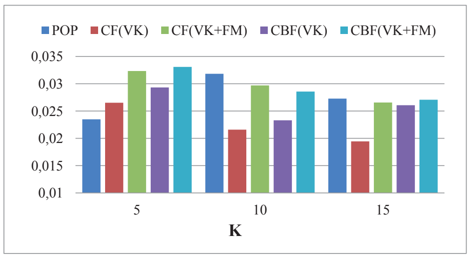
We used offline evaluation to compare results of proposed methods with baselines. The recommender system suggested 30 popular VK recordings to each testing VK user excluding recordings that the user has already added in the training set. In each approach the recommendation list consists of the same items. We chose popular items for evaluation, due to the high probability that users have seen them already.

In this study, we demonstrate the performance improvement resulting from the source domain with three simple but popular algorithms: (1) POP, (2) Collaborative Filtering (CF), and (3) Content-based Filtering (CBF). In particular, POP is a non-personalized recommendation algorithm, which orders items in the suggested list according to their popularity in the VK dataset. For the CF and the CBF algorithms, we obtained two performance results based on only VK and VK+FM data.

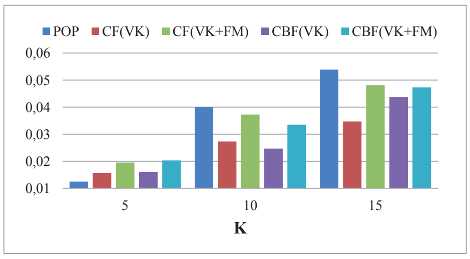
- **POP** - ordering items according to their popularity using the VK dataset.
- **CF(VK)** - item-based collaborative filtering using the VK dataset.
- **CF(VK+FM)** - item-based collaborative filtering using VK and FM datasets.
- **CBF(VK)** - content-based filtering using the VK dataset.
- **CBF(VK+FM)** - content-based filtering using VK and FM datasets.

Figures 2, 3 and 4 demonstrate the experimental results based on three datasets presented in Section 3. From the figures we can observe that:

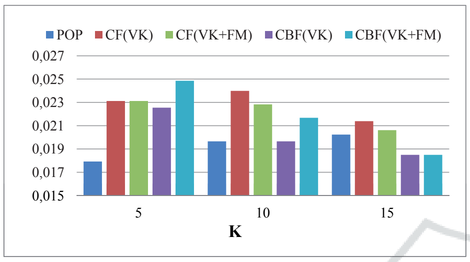
1. The source domain can improve the recommendation performance in the target domain when only items overlap. For the 0% dataset, the CF algorithm achieves 0.0216, 0.0273, 0.0139 and 0.0287



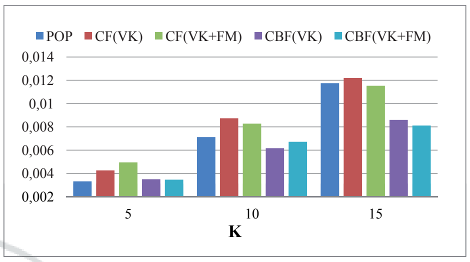
(a) Precision@K (0%)



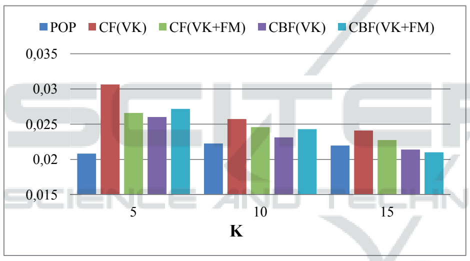
(b) Recall@K (0%)



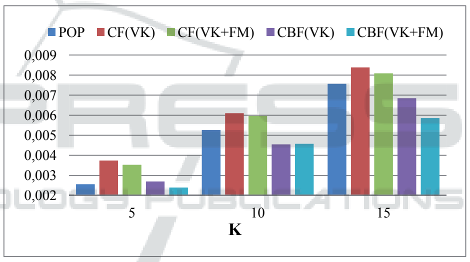
(c) Precision@K (50%)



(d) Recall@K (50%)



(e) Precision@K (100%)



(f) Recall@K (100%)

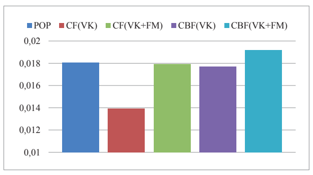
Figure 2: Precision@K and Recall@K for experiments conducted using datasets with different fractions of non-overlapping items.

in terms of precision@10, recal@10, MAP and NDCG@10 based on VK dataset, while these numbers are 0.0297, 0.0372, 0.0179 and 0.0387 based on VK+FM dataset, making the improvement of 37.5%, 36.3%, 28.8% and 34.8%, respectively. Similar improvements can be observed for the CBF algorithm.

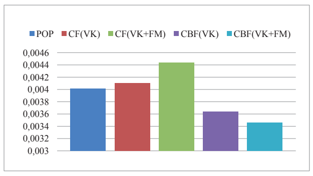
2. The improvement declines with the growth of non-overlapping items in different domains. For example, the improvement of CBF in terms of NDCG@10 decreases as follows: 20.1%, 5.4% and 5.0% using 0%, 50% and 100% datasets, respectively. For the CF algorithm, the declining trend is even sharper. The source domain decreases the performance of the CF algorithm by 11.8% and 7.0% in terms of NDCG@5 and NDCG@10 respectively using 100% dataset. A similar trend can be observed for other numbers of first *K* results and evaluation metrics.

3. CF(VK) and CBF(VK) perform worse than POP in different cases, especially using the dataset that contains only overlapping recordings (0%). CF(VK) algorithm outperforms the popularity baseline with the increase of non-overlapping recordings. CF(VK) achieves 0.0139, while POP outperforms them with the number 0.0180 in terms of MAP using 0% dataset. For 100% dataset the situation is opposite. POP achieves 0.0029, while CF(VK) reaches 0.0031. POP outperforms CBF(VK) algorithm in most cases. For 0% and 100% datasets, CBF(VK) performs 1.9% and 8.4% worse than POP in terms of MAP, respectively.

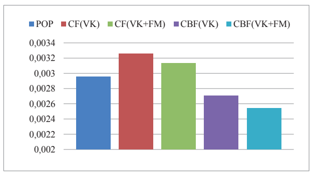
Observation 1 illustrates the global correlation of



(a) MAP (0%)

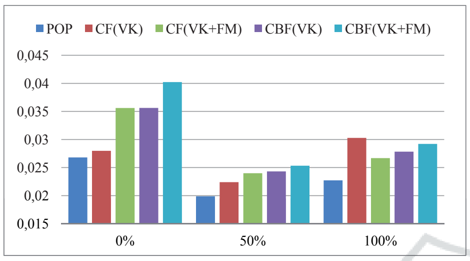


(b) MAP (50%)

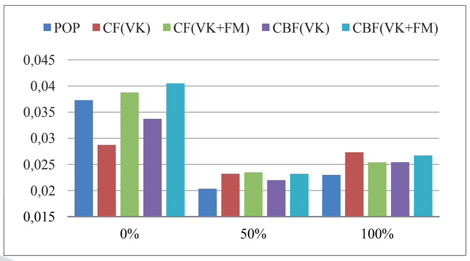


(c) MAP (100%)

Figure 3: MAP (30 recommendations) for experiments conducted using datasets with different fractions of non-overlapping items.



(a) NDCG@5



(b) NDCG@10

Figure 4: NDCG@K for experiments conducted using datasets with different fractions of non-overlapping items.

users' preferences in different domains (Winoto and Tang, 2008; Fernández-Tobías et al., 2012). Although, the data belongs to different domains, users' ratings from the source domain indicate similarities between items that improve the recommendation performance in the target domain.

Observation 2 supports the claim (Fernández-Tobías et al., 2012), that the improvement cased by the source domain rises with the growth of the overlap between target and source domains. The decrease in the recommendation performance of the CF algorithm with the FM data is caused by the different lengths of item vectors in source and target domains, where vectors of FM items contain significantly more dimensions than vectors of VK items.

In observation 3, the non-personalized algorithm POP outperforms both the personalized algorithms in different cases. CF algorithm performs worse than POP due to data sparsity, which is alleviated by adding more VK recordings to the dataset. Low performance of CBF is caused by the poor quality of item descriptions, as in the VK dataset items are described with artists only.

Figures 2, 3 and 4 demonstrate four evaluation metrics that are not always consistent. However, the described observations can still be notices.

## 6 CONCLUSION

In this paper we investigated cross-domain recommendations in the situation when only items overlap on system level. We collected data from VK and FM and built three datasets that contain different fractions of non-overlapping items from source and target domains. We then conducted experiments using collaborative filtering and content-based filtering algorithms to demonstrate the importance of additional data.

According to our results, the source domain can boost the recommendation performance in the target domain when only items overlap resulting from the correlation of users' preferences among different domains (Winoto and Tang, 2008). However, similarly to (Fernández-Tobías et al., 2012) our results indicated that the more items overlap in source and target domains with respect to the whole dataset the higher the improvement.

## ACKNOWLEDGEMENT

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

## REFERENCES

Abel, F., Herder, E., Houben, G.-J., Henze, N., and Krause, D. (2013). Cross-system user modeling and personalization on the social web. *User Modeling and User-Adapted Interaction*, 23(2-3):169–209.

Berkovsky, S., Kuflik, T., and Ricci, F. (2008). Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245–286.

Cantador, I. and Cremonesi, P. (2014). Tutorial on cross-domain recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 401–402, New York, NY, USA. ACM.

Cremonesi, P., Tripodi, A., and Turrin, R. (2011). Cross-domain recommender systems. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 496–503.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173.

Fernández-Tobías, I., Cantador, I., Kaminskas, M., and Ricci, F. (2012). Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Kille, B., Hopfgartner, F., Brodt, T., and Heintz, T. (2013). The plista dataset. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, pages 16–23, New York, NY, USA. ACM.

Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer US.

Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*. Springer US.

Sahebi, S. and Brusilovsky, P. (2013). Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In *User Modeling, Adaptation, and Personalization*, volume 7899 of *Lecture Notes in Computer Science*, pages 289–295. Springer Berlin Heidelberg.

Sang, J. (2014). Cross-network social multimedia computing. In *User-centric Social Multimedia Computing*, Springer Theses, pages 81–99. Springer Berlin Heidelberg.

Shapira, B., Rokach, L., and Freilikhman, S. (2013). Facebook single and cross domain data for recommendation systems. *User Modeling and User-Adapted Interaction*, 23(2-3):211–247.

Winoto, P. and Tang, T. (2008). If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26(3):209–225.

Zhao, Y.-L. (2013). *Community Learning in Location-based Social Networks*. Thesis. Ph.D.

# PVIII

# IMPROVING SERENDIPITY AND ACCURACY IN CROSS-DOMAIN RECOMMENDER SYSTEMS

by

Denis Kotkov, Shuaiqiang Wang, Jari Veijalainen 2017

Web Information Systems and Technologies

# Improving Serendipity and Accuracy in Cross-Domain Recommender Systems

Denis Kotkov[1(✉)], Shuaiqiang Wang[2], and Jari Veijalainen[1]

[1] Faculty of Information Technology, University of Jyvaskyla, P.O.Box 35, FI-40014 Jyvaskyla, Finland
deigkotk@student.jyu.fi,jari.veijalainen@jyu.fi

[2] Manchester Business School, The University of Manchester, Manchester, UK
shuaiqiang.wang@manchester.ac.uk

**Abstract.** Cross-domain recommender systems use information from source domains to improve recommendations in a target domain, where the term *domain* refers to a set of items that share attributes and/or user ratings. Most works on this topic focus on accuracy but disregard other properties of recommender systems. In this paper, we attempt to improve serendipity and accuracy in the target domain with datasets from source domains. Due to the lack of publicly available datasets, we collect datasets from two domains related to music, involving user ratings and item attributes. We then conduct experiments using collaborative filtering and content-based filtering approaches for the purpose of validation. According to our results, the source domain can improve serendipity in the target domain for both approaches. The source domain decreases accuracy for content-based filtering and increases accuracy for collaborative filtering. The improvement of accuracy decreases with the growth of non-overlapping items in different domains.

**Keywords:** Recommender systems · Serendipity · Cross-domain recommendations · Collaborative filtering · Content-based filtering · Data collection

## 1 Introduction

Recommender systems use past user behavior to suggest items interesting to users [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" [12]. Recommender systems use algorithms to generate recommendations.

Traditional recommendation algorithms mainly aim to improve accuracy, which indicates how good an algorithm is at suggesting items a user usually consumes. In this paper, they are referred to as *accuracy-oriented algorithms*.

Shuaiqiang Wang—The research was conducted while the author was working for the University of Jyvaskyla, Finland.

V. Monfort et al. (Eds.): WEBIST 2016, LNBIP 292, pp. 1–15, 2017.
DOI: 10.1007/978-3-319-66468-2_6

Generally speaking, accuracy-oriented algorithms often suggest popular items, as these items are widely consumed by individuals. To improve accuracy, recommendation algorithms also tend to suggest items similar to a user profile (a set of items rated by the user [12]), as these items match previous user tastes. As a result, a user is recommended (1) items that are popular and therefore familiar to the user [6] and (2) items that the user can easily find him/herself, which is referred to as the *overspecialization problem* [21]. In particular, as two main categories of recommendation algorithms, collaborative filtering algorithms often suggest popular items due to the popularity bias in most datasets, while content-based filtering algorithms often suffer from the overspecialization problem due to insufficient information regarding attributes of items.

Typically, the main reason why a user joins a recommender system is to find novel and interesting items the user would not find him/herself [21]. To improve user satisfaction, a recommender system should suggest serendipitous items [12]. In this paper, we follow the definitions of [2,10,12], which indicate that serendipitous items must be relevant, novel and unexpected to a user.

The mentioned problems can be tackled by cross-domain recommender systems, which could predict serendipitous items by enriching the training data from the target domain with additional datasets from other domains. Here the term *domain* refers to "a set of items that share certain characteristics that are exploited by a particular recommender system" [9]. These characteristics are item attributes and user ratings. Recommender systems that take advantage of multiple domains are called *cross-domain recommender systems* [4,9,13].

In this paper, we explore the *cross-domain recommendation task* [4,13], that requires one *target domain* and at least one *source domain*. The former refers to the domain from which suggested items are picked from, and similarly the latter refers to the domain that contains auxiliary information.

In this work, we seek to address the following research question: Can the source domain improve serendipity in the target domain? Due to the lack of publicly available datasets for cross-domain recommender systems [3,11,13], we collected data from Vkontakte[1] (VK) – Russian online social network (OSN) and Last.fm[2] (FM) – music recommender service. We then matched VK and FM audio recordings and developed the cross-domain recommender system that suggests VK recordings to VK users based on data from both domains. Each audio recording is represented by its metadata excluding the actual audio file. VK recordings thus represent the target domain, while the source domain consists of FM recordings. VK and FM recordings share titles and artists, but have different user ratings and other attributes.

We regard items that share certain attributes and belong to different domains as *overlapping*, while those that do not as *non-overlapping*. In our case, VK and FM recordings that have the same titles and artists are overlapping items.

To address the research question and illustrate the potential of additional data, we chose simple but popular recommendation algorithms to conduct exper-

[1] http://vk.com/.
[2] http://last.fm/.

iments for validation: collaborative filtering based on user ratings and content-based filtering based on the descriptions of the items.

Our results indicate that the source domain can improve serendipity in the target domain for both collaborative filtering and content-based filtering algorithms:

– The traditional collaborative filtering algorithms tend to suggest popular items, as most datasets contain rich information regarding these items in terms of user ratings. Combing datasets of different domains decreases the popularity bias.
– Content-based filtering algorithms often suffer from the overspecialization problem due to poor data regarding item attributes. Enriching item attributes alleviates the problem and increases serendipity.

According to our results, the source domain has a negative impact on accuracy for content-based filtering, and a positive impact on accuracy of collaborative filtering. Furthermore, with the growth of non-overlapping items in different domains, the improvement of accuracy for collaborative filtering decreases.
This paper has the following contributions:

– We initially investigate the cross-domain recommendation problem in terms of serendipity.
– We collect a novel dataset to conduct the experiments for addressing the research question.

The rest of the paper is organized as follows. Section 2 overviews related works. Section 3 describes the datasets used to conduct experiments. Section 4 is dedicated to recommendation approaches, while Sect. 5 describes conducted experiments. Finally, Sect. 6 draws final conclusions.

## 2 Related Works

In this section, we survey state-of-the-art efforts regarding serendipity and cross-domain recommendations.

### 2.1 Serendipity in Recommender Systems

According to the dictionary[3], serendipity is "the faculty of making fortunate discoveries by accident". The term was coined by Horace Walpole, who referenced the fairy tale, "The Three Princes of Serendip", to describe his unexpected discovery [16].

Currently, there is no agreement on definition of serendipity in recommender systems. Researchers employ different definitions in their studies. In this paper, we employ the most common definition, which indicates that serendipitous items are relevant, novel and unexpected [2,10,12].

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

Given the importance of serendipity, researchers have proposed different serendipity-oriented recommendation algorithms. For example, Lu et al. presented a serendipitous personalized ranking algorithm [15]. The algorithm is based on matrix factorization with the objective function that incorporates relevance and popularity of items. Another matrix factorization based algorithm is proposed by Zheng, Chan and Ip [24]. The authors proposed the unexpectedness-augmented utility model, which takes into account relevance, popularity and similarity of items to a user profile. In contrast, Zhang et al. provided the recommendation algorithm *Full Auralist* [23]. It consists of three algorithms, each being responsible for relevance, diversity and unexpectedness. To the best of our knowledge, studies that focus on improving serendipity using source domains are of restricted availability.

### 2.2 Cross-Domain Recommendations

Cross-domain recommender systems use multiple domains to generate recommendations, which can be categorized based on domain levels [4,5]:

– Attribute level. Items have the same type and attributes. Two items are assigned to different domains if they have different values of a particular attribute. A pop song and jazz song might belong to different domains.
– Type level. Items have similar types and share some common attributes. Two items are assigned to different domains if they have different subsets of attributes. A photograph and animated picture might belong to different domains. Even though both items have common attributes, such as a title, publisher and tags, other attributes might be different (duration attribute for animated pictures).
– Item level. Items have different types and all or almost all attributes. Two items are assigned to different domains if they have different types. A song and book might belong to different domains, as almost all attributes of the items are different.
– System level. Two items are assigned to different domains if they belong to different systems. For example, movies from IMDb[4] and MovieLens[5] might belong to different domains.

Depending on whether overlapping occurs in the set of users or items [7], there are four situations that enable cross-domain recommendations: (a) no overlap between items and users, (b) user sets of different domains overlap while item sets do not overlap, (c) item sets overlap while user sets do not overlap, and (d) item and user sets overlap.

Most efforts on cross-domain recommendations focus on the situation when users or both users and items overlap [13]. For example, Sang demonstrated the feasibility of utilizing the source domain. The study was conducted on a

[4] http://www.imdb.com/.
[5] https://movielens.org/.

dataset collected from Twitter[6] and YouTube[7]. The author established relationships between items from different domains using topics [19]. Similarly to Sang, Shapira, Rokach and Freilikhman also linked items from different domains, where 95 participants rated movies and allowed the researches to collect data from their Facebook pages [20]. The results suggested that source domains improve the recommendation performance [20]. Another study with positive results was conducted by Abel et al. The dataset contained information related to the same users from 7 different OSNs [1]. Sahebi and Brusilovsky demonstrated the usefulness of recommendations based on source domains to overcome cold start problem [18].

Most works on cross-domain recommendations focus on accuracy. To the best of our knowledge, the efforts on the impact of source domains on the target domain in terms of serendipity involving a real cross-domain dataset are very limited. In this paper, we investigate whether source domains can improve serendipity in the target domain when only items overlap on system level.

## 3 Datasets

Due to the lack of publicly available datasets for cross-domain recommender systems with overlapping items [3,11] we collected data from VK and FM. The construction of the dataset included three phases (Fig. 1): (1) VK recordings collection, (2) duplicates matching, and (3) FM recordings collection.

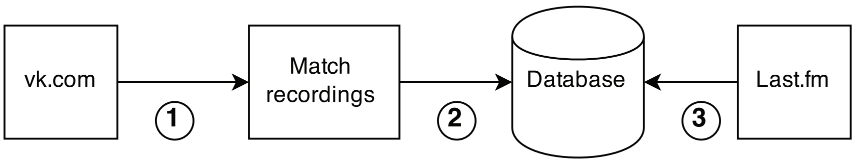


**Fig. 1.** Data collection chart.

### 3.1 VK Recordings Collection

The VK interface provides the functionality to add favored recordings to users' pages. By generating random user IDs we collected accessible VK users' favored audio recordings using VK API. Each audio recording is represented by its metadata excluding the actual audio file. Our VK dataset consists of $97,737$ ($76,177$ unique) audio recordings added by 864 users.

Each VK user is allowed to share any audio or video recording. The interface of the OSN provides the functionality to add favored recordings to the users page. VK users are allowed not only to add favored audio recordings to their

[6] https://twitter.com/.
[7] https://www.youtube.com/.

pages, but also to rename them. The dataset thus contains a noticeable number of duplicates with different names. To assess this number we randomly selected 100 VK recordings and manually split them into three categories:

- Correct names - the name of the recording is correctly written without any grammatical mistakes or redundant symbols.
- Misspelled names - the name is guessable, even if the name of the recording is replaced with the combination of artist and recording name or lyrics.
- Meaningless names - the name does not contain any information about the recording. For example, “unknown” artist and “the song” recording.

Out of 100 randomly selected recordings we detected 14 misspelled and 2 meaningless names. The example can be seen from Table 1.

**Table 1.** Examples of recordings.

| Artist name | Recording name |
|---|---|
| *Correct names* | |
| Beyonce | Halo |
| Madonna | Frozen |
| *Misspelled* | |
| Alice DJ | Alice DJ - Better of Alone.mp3 |
| Reamonn | Oh, tonight you kill me with your smile |
| ● Lady Gaga | Christmas Tree |
| *Meaningless* | |
| Unknown | Classic |
| Unknown | Party |

## 3.2 Duplicates Matching

To match misspelled recordings, we developed a duplicate matching algorithm that detects duplicates based on recordings’ names, mp3 links and durations. The algorithm compares recordings’ names based on the Levenshtein distance and the number of common words excluding stop words.

We then removed some popular meaningless recordings such as “Unknown”, “1” or “01”, because they represent different recordings and do not indicate user preferences. Furthermore, some users assign wrong popular artists’ names to the recordings. To restrict the growth of these kinds of mistakes, the matching algorithm considers artists of the duplicate recordings to be different. By using the presented matching approach, the number of unique recordings decreased from $76,177$ to $68,699$.

### 3.3 FM Recordings Collection

To utilize the source domain we collected FM recordings that correspond to $48,917$ selected VK recordings that were added by at least two users or users that have testing data. Each FM recording contains descriptions such as FM tags added by FM users. FM tags indicate additional information such as genre, language or mood. Overall, we collected $10,962$ overlapping FM recordings and $20,214$ ($2,783$ unique) FM tags.

It is also possible to obtain FM users who like a certain recording (top fans). For each FM recording, we collected FM users who like at least one more FM recording from our dataset according to the distribution of VK users among those recordings. In fact, some unpopular FM recordings are missing top fans. We thus collected $17,062$ FM users, where $7,083$ of them like at least two recordings from our database. FM users liked $4,609$ FM recordings among those collected.

### 3.4 The Statistics of the Datasets

In this work, we constructed three datasets. Each of them includes the collected FM data and different parts of the VK data (percentage indicates the fraction of overlapping items):

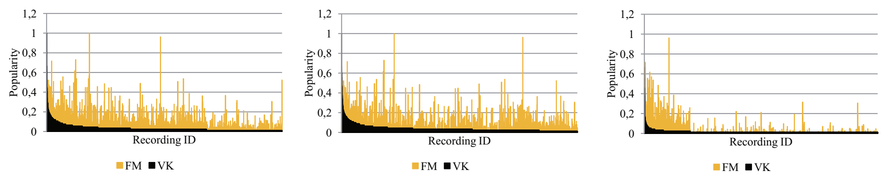
– 100% - the dataset contains only overlapping recordings picked by VK and FM users;
– 50% - the dataset contains equal number of overlapping and non-overlapping recordings;
– 7% - the dataset contains all collected VK and FM recordings. The fraction of overlapping recordings is 6.7%.

The 7% dataset contains all the collected and processed data. We presented results for 50% and 100% datasets to demonstrate how serendipity and accuracy change when a dataset contains different fraction of overlapping items.

**Table 2.** The statistics of the datasets.

| | 100% | | 50% | | 7% | |
|---|---|---|---|---|---|---|
| | VK | FM | VK | FM | VK | FM |
| Users | 665 | $7,083$ | 795 | $7,083$ | 864 | $7,083$ |
| Ratings | 14,526 | $40,782$ | 33,680 | $40,782$ | 96,737 | $40,782$ |
| Items | 4,609 | $4,609$ | 9,218 | $4,609$ | 68,699 | $4,609$ |
| Artists | 1,986 | $1,986$ | 4,595 | $1,986$ | 31,861 | $1,986$ |
| Tags | - | $20,167$ | - | $20,167$ | - | $20,167$ |

The statistics of the datasets are presented in Table 2. The number of VK users varies in different datasets, due to the lack of ratings after removing non-overlapping VK recordings.



(a) 100% dataset (b) 50% dataset (c) 7% dataset

**Fig. 2.** Popularity distributions of VK and FM datasets.

According to Fig. 2, each recording has different popularity among VK and FM users. The FM dataset contains rich information in terms of user ratings regarding recordings unpopular in the VK dataset. In the figure, popularity is based on the number of users who picked a particular item:

$$Popularity_i = \frac{Freq(i)}{Freq_{max}}, \tag{1}$$

where $Freq(i)$ is the number of users who picked recording $i$, while $Freq_{max}$ corresponds to the maximum number of users picked the same recording in a dataset.

## 4 Recommendation Approaches

In this section, we implement and observe simple but popular collaborative filtering and content-based filtering algorithms to demonstrate the impact of the data from source domains.

### 4.1 Item-Based Collaborative Filtering

We chose item-based collaborative filtering as the first experimental algorithm. It is a representative recommendation algorithm that has been widely used in industry due to its scalability [8]. In item-based collaborative filtering, each audio recording (item) is represented as a vector in a multidimensional feature space, where each feature is a user's choice (rating). VK recording is represented as follows: $i^{vk} = (u^{vk}_{1,i}, u^{vk}_{2,i}, ..., u^{vk}_{n,i})$, and each element $u^{vk}_{k,i} \in \{0, 1\}$ for $k = 1, ..., ||U||$, where $U$ is a set of users, while $u^{vk}_{k,i}$ equals to 1 if VK user $k$ picks VK recording $i^{vk}$ and 0 otherwise. To integrate the source domain (FM) with our target domain (VK), we included FM users as follows: $i^{vkfm} = (u^{vk}_{1,i}, u^{vk}_{2,i}, ..., u^{vk}_{n,i}, u^{fm}_{1,i}, u^{fm}_{2,i}, ..., u^{fm}_{n,i})$.

To generate recommendations, item-based collaborative filtering first detects recordings that are most similar to recordings picked by the target user. The algorithm then ranks recordings based on the obtained similarities.

To measure similarity, we used conditional probability, which is a common similarity measure for situations in which users only indicate items they like without specifying how much they like these items (unary data) [8]. Conditional probability is calculated as follows:

$$p(i,j) = \frac{Freq(i \wedge j)}{Freq(i) \cdot Freq(j)^{\alpha}}, \tag{2}$$

where $Freq(i)$ is the number of users that picked item $i$, while $Freq(i \wedge j)$ is the number of users that picked both items $i$ and $j$. The parameter $\alpha$ is a damping factor to decrease the similarity for popular items. In our experiments $\alpha = 1$.

Item vectors based on FM users contain remarkably more dimensions than vectors based on VK users. To alleviate the problem, we compared recordings using the following rule:

$$sim(i,j) = \begin{cases} p(i^{vk}, j^{vk}), & \exists i^{vk} \wedge \exists j^{vk} \wedge \\ & (\nexists i^{fm} \vee \nexists j^{fm}) \\ p(i^{fm}, j^{fm}), & \exists i^{fm} \wedge \exists j^{fm} \wedge \\ & (\nexists i^{vk} \vee \nexists j^{vk}) \\ p(i^{vkfm}, j^{vkfm}), & \exists i^{vk} \wedge \exists j^{vk} \wedge \\ & \exists i^{fm} \wedge \exists j^{fm} \end{cases}. \tag{3}$$

We compared items in each pair using domains that contain user ratings for both items. To rank items in the suggested list, we used sum of similarities of recordings [8]:

$$score(u,i) = \sum\nolimits_{j \in I_u} sim(i,j), \tag{4}$$

where $I_u$ is the set of items picked by user $u$ (user profile).

## 4.2 Content-Based Filtering

We chose content-based filtering algorithm, as this algorithm uses item attributes instead of user ratings to generate recommendations. In our case, these attributes are VK - FM artists and FM tags. Each FM artist corresponds to a particular VK artist.

To represent items, we used a common weighting scheme, term frequency-inverse document frequency (TF-IDF). TF-IDF weight consists of two parts:

$$tfidf_{attr,i} = tf_{attr,i} \cdot idf_{attr}, \tag{5}$$

where $tf_{attr,i}$ corresponds to the frequency of attribute $attr$ for item $i$ (term frequency), while $idf_{attr}$ corresponds to the inverse frequency of attribute $attr$ (inverse document frequency). The term frequency is based on the number of times an attribute appears among attributes of an item with respect to the number of item attributes:

$$tf_{attr,i} = \frac{n_{attr,i}}{n_i}, \tag{6}$$

where $n_i$ is the number of attributes of item $i$, while $n_{attr,i}$ is the number of times attribute $attr$ appears among attributes of item $i$. In our case, $n_{attr,i} = 1$ for each item, while $n_i$ varies depending on the item. The term frequency increases with the decrease of the number of item attributes. The inverse document frequency is based on the number of items with an attribute in the dataset:

$$idf_{attr} = ln \frac{||I||}{||I_{attr}||}, \tag{7}$$

where $I$ is a set of all the items, while $I_{attr}$ is a set of items that have attribute $attr$. The inverse document frequency is high for rare attributes and low for popular ones. TF-IDF weighting scheme assigns high weights to rare attributes that appear in items with low number of attributes.

An audio recording is represented as follows: $i^a = (a_{1,i},\ a_{2,i}, ..., a_{d,i})$, where $a_{k,i}$ corresponds to the TF-IDF weight of artist $a_k$ [14]. The user is represented as follows: $u^a = (a_{1,u}, a_{2,u}, ..., a_{d,u})$, where $a_{k,u}$ corresponds to the number of recordings picked by user $u$ performed by artist $a_k$.

To integrate FM data, we considered FM tags as follows: $i^{at} = (a_{1,i},\ a_{2,i}, ..., a_{d,i},\ t_{1,i},\ t_{2,i}, ..., t_{q,i})$, where $t_{k,i}$ corresponds to the TF-IDF weight of tag $t_k$ [14]. The user vector then is denoted as follows: $u^{at} = (a_{1,u},\ a_{2,u}, ..., a_{d,u},\ t_{1,u}, t_{2,u}, ..., t_{q,u})$, where $t_{k,u}$ is the number of recordings picked by user $u$ having tag $t_k$.

The recommender system compares audio recordings' vectors and a user vector using cosine similarity [8]:

$$cos(u, i) = \frac{u \cdot i}{||u||||i||}, \tag{8}$$

where $u$ and $i$ are user and item vectors. To suggest recordings, content-based filtering ranks recordings according to $cos(u, i)$. In our experiments, we used $cos(u^a, i^a)$ for VK data and $cos(u^{at}, i^{at})$ for VK and FM data.

## 5 Experiments

In this section, we detail experiments conducted to demonstrate whether the source domain improves serendipity and accuracy in the target domain when only items overlap.

### 5.1 Evaluation Metrics

To assess the performance of algorithms we used two metrics: (1) $Precision@K$ to measure accuracy and (2) a traditional serendipity metric $Ser@K$.

$Precision@K$ is a commonly used metric to assess quality of recommended lists with binary relevance. In our datasets, recordings added by a user to his/her page are relevant, while the rest of the recordings are irrelevant to the user. $Precision@K$ reflects the fraction of relevant recordings retrieved by a recommender system in the first $K$ results. The metric is calculated as follows:

$$Precision@K = \frac{1}{||U||} \sum_{u \in U} \frac{||RS_u(K) \cap REL_u||}{K}, \tag{9}$$

where $U$ is a set of users, while $RS_u(K)$ is a set of top-K suggestions for user $u$. Recordings from the test set (ground truth) for user $u$ are represented by $REL_u$.

The traditional serendipity metric is based on (1) a primitive recommender system, which suggests items known and expected by a user, and (2) a set of items similar to a user profile. Evaluated recommendation algorithms are penalized for suggesting items that are irrelevant, generated by a primitive recommender system or included in the set of items similar to a user profile. Similarly to [2], we used a slight modification of the serendipity metric:

$$Ser@K = \frac{1}{||U||} \sum_{u \in U} \frac{||(RS_u(K) \backslash PM \backslash E_u) \cap REL_u||}{K}, \tag{10}$$

where $PM$ is a set of suggestions generated by the primitive recommender system, while $E_u$ is a set of recordings similar to recordings picked by user $u$. We selected the 10 most popular recordings for $PM$ following one of the most common strategies [15,24]. Set of items similar to a user profile $E_u$ represents all the recordings that have common artists with recordings user $u$ picked. User $u$ can easily find recordings from set $E_u$ by artist name, we therefore regard these recordings as obvious.

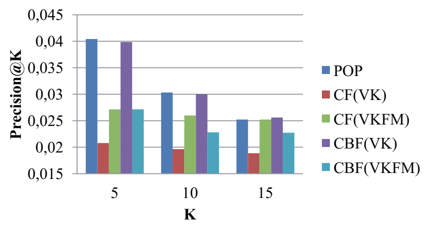
## 5.2 Results

Following the datasets sampling strategy in [8], we split each of our datasets into training and test datasets and applied 3-fold cross-validation. We selected 40% of the users who picked the most VK recordings, and chose 30% of their ratings as the testing dataset. We then regarded the rest of the ratings as the training dataset.

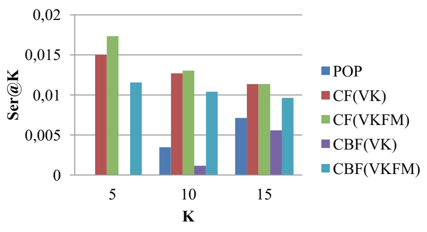
To compare the results of various baselines, we used offline evaluation. The recommender system suggested 30 popular VK recordings to each testing VK user excluding recordings that the user has already added in the training set. In each approach the recommendation list consists of the same items. We chose popular items for evaluation, as the users are likely to be familiar with those items.

In this study, we demonstrate serendipity and accuracy improvements resulting from the source domain with three simple but popular algorithms: (1) POP, (2) Collaborative Filtering (CF), and (3) Content-Based Filtering (CBF). It is important to note that POP is a non-personalized recommendation algorithm, which orders items in the suggested list according to their popularity in the VK dataset. For the CF and the CBF algorithms, we obtained two performance results based on (1) data collected from VK and (2) data collected from both VK and FM.

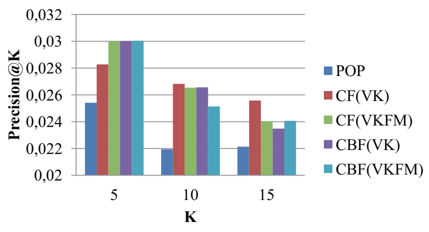
- **POP** - ordering items according to their popularity using the VK dataset.
- **CF(VK)** - item-based collaborative filtering using the VK dataset.
- **CF(VKFM)** - item-based collaborative filtering using VK and FM datasets.
- **CBF(VK)** - content-based filtering using the VK dataset.



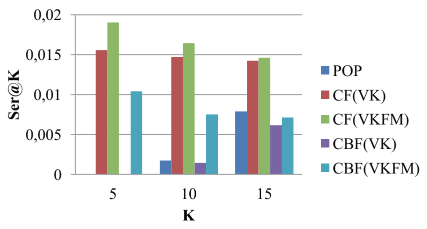
(a) 100% dataset



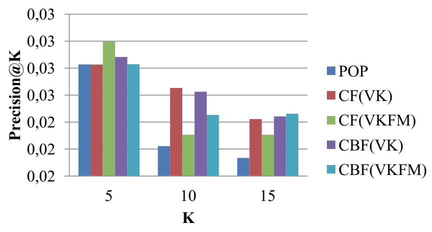
(b) 100% dataset



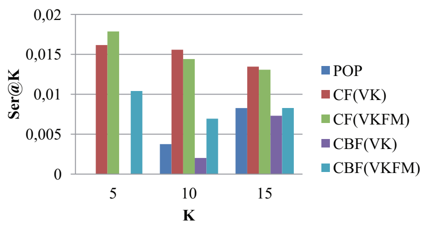
(c) 50% dataset



(d) 50% dataset



(e) 7% dataset



(f) 7% dataset

**Fig. 3.** *Precision*@*K* and *Ser*@*K* for experiments conducted using datasets with different fractions of non-overlapping items.

– **CBF(VKFM)** - content-based filtering using VK and FM datasets.

Figure 3 demonstrates the experimental results based on three datasets presented in Sect. 3. From the figure we can observe that:

1. The source domain can improve serendipity in the target domain. On all datasets, CBF based on VK and FM data outperforms CBF based on only VK data in terms of serendipity. For collaborative filtering the situation is very similar, except the decrease of serendipity for recommendation lists of length 10 and 15 on the 7% dataset. For the 50% dataset, the CF algorithm achieves 0.0156, 0.0147 and 0.0142 in terms of *Ser*@5, *Ser*@10 and *Ser*@15 based on VK data, while these numbers are 0.0190, 0.0164 and 0.0146 based on VK and FM data, making the improvement of 22.2%, 11.7% and 2.7%, respectively.
2. For collaborative filtering, the source domain can improve accuracy in the target domain when only items overlap. For the 100% dataset, the CF algorithm achieves 0.0208, 0.0196 and 0.0189 in terms of *Precision*@5, *Precision*@10 and *Precision*@15 based on VK data, while these numbers are 0.0271, 0.0260

and 0.0252 based on VK and FM data, making the improvement of 30.6%, 32.4% and 33.7%, respectively.
3. The improvement of accuracy declines with the growth of non-overlapping items for collaborative filtering. The improvement of CF in terms of $Precision$@5 decreases as follows: 30.6%, 6.1% and 6.0% using 100%, 50% and 7% datasets, respectively.
4. The source domain decreases accuracy of content-based filtering. For the 100% dataset, CBF based on VK and FM data decreases $Precision$@5, $Precision$@10 and $Precision$@15 by 31.9%, 24.0% and 11.2%, respectively.
5. Despite being accurate, popularity baseline has a very low serendipity. POP outperforms other algorithms in terms of accuracy on the 100% dataset. Meanwhile, the algorithm fails to suggest any serendipitous items in top-5 recommendations on each dataset.

According to observations 1 and 2, CF(VKFM) outperforms CF(VK) in terms of both serendipity and accuracy. The improvement of accuracy illustrates the global correlation of user preferences in different domains [9,22]. Although, the data belongs to different domains, user ratings from the source domain indicate similarities between items that improve the recommendation performance in the target domain. The improvement of serendipity is caused by the growth of accuracy and by different popularity distributions in VK and FM datasets.

Observation 3 supports the claim [9], that the improvement caused by the source domain rises with the growth of the overlap between target and source domains. The decrease of accuracy for the CF algorithm with the FM data is caused by the different lengths of item vectors in source and target domains, where vectors of FM items contain significantly more dimensions than vectors of VK items.

Observations 1 and 4 indicate that the FM data positively contributes to serendipity and negatively affects accuracy of the content-based filtering algorithm. As users tend to add recording of the same artist, CBF(VK) significantly outperforms CBF(VKFM). However, most recordings suggested by CBF(VK) are obvious to a user, as the user can find these recordings him/herself. As a result, the serendipity of CBF(VK) is very low. FM tags help recommend similar recordings of artists novel to the user. Recordings that share the same FM tags do not necessarily share the same artists, which results in the decrease of accuracy and increase of serendipity.

Observation 5 indicates that POP has very low serendipity, despite being accurate. Popular recommendations are likely to be accurate, as users tend to add familiar recordings. However, popular recordings are widely recognized by users and therefore regarded as obvious.

## 6 Conclusion

In this paper, we first initially investigated the cross-domain recommendation problem in terms of serendipity. We collected data from VK and FM and built three datasets that contain different fractions of non-overlapping items from source and target domains. We then conducted extensive experiments with collaborative filtering and content-based filtering algorithms to demonstrate the impact of source domains on performance gains of the target domain.

According to our results, the source domain can improve serendipity in the target domain when only items overlap on system level for both collaborative filtering and content-based filtering algorithms. The integration of the source domain resulted in the decrease of accuracy for content-based filtering and the increase of accuracy for collaborative filtering. Similarly to [9] our results indicated that the more items overlap in source and target domains with respect to the whole dataset the higher the improvement of accuracy for collaborative filtering.

**Acknowledgement.** The research at the University of Jyväskylä was performed in the MineSocMed project, partially supported by the Academy of Finland, grant #268078. The communication of this research was supported by Daria Wadsworth.

## References

1. Abel, F., Herder, E., Houben, G.J., Henze, N., Krause, D.: Cross-system user modeling and personalization on the social web. User Model. User-Adap. Inter. **23**, 169–209 (2013)
2. Adamopoulos, P., Tuzhilin, A.: On unexpectedness in recommender systems: or how to better expect the unexpected. ACM Trans. Intell. Syst. Technol. **5**, 1–32 (2014)
3. Berkovsky, S., Kuflik, T., Ricci, F.: Mediation of user models for enhanced personalization in recommender systems. User Model. User-Adap. Inter. **18**, 245–286 (2008)
4. Cantador, I., Cremonesi, P.: Tutorial on cross-domain recommender systems. In: Proceedings of the 8th ACM Conference on Recommender Systems, New York, NY, USA, pp. 401–402 (2014)
5. Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 919–959. Springer, Boston (2015). doi:10.1007/978-1-4899-7637-6_27
6. Celma Herrada, Ò.: Music recommendation and discovery in the long tail. Ph.D. thesis, Universitat Pompeu Fabra (2009)
7. Cremonesi, P., Tripodi, A., Turrin, R.: Cross-domain recommender systems. In: 11th IEEE International Conference on Data Mining Workshops, pp. 496–503 (2011)
8. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum. Comput. Interact. **4**, 81–173 (2011)
9. Fernández-Tobías, I., Cantador, I., Kaminskas, M., Ricci, F.: Cross-domain recommender systems: a survey of the state of the art. In: Proceedings of the 2nd Spanish Conference on Information Retrieval, pp. 187–198 (2012)
10. Iaquinta, L., Semeraro, G., de Gemmis, M., Lops, P., Molino, P.: Can a recommender system induce serendipitous encounters? IN-TECH (2010)
11. Kille, B., Hopfgartner, F., Brodt, T., Heintz, T.: The plista dataset. In: Proceedings of the 2013 International News Recommender Systems Workshop and Challenge, pp. 16–23. ACM, New York (2013)

12. 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. SCITEPRESS (2016)
13. Kotkov, D., Wang, S., Veijalainen, J.: Cross-domain recommendations with overlapping items. In: Proceedings of the 12th International Conference on Web Information Systems and Technologies, vol. 2, pp. 131–138. SCITEPRESS (2016)
14. Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: state of the art and trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) Recommender Systems Handbook, pp. 73–105. Springer, Boston (2011). doi:10.1007/978-0-387-85820-3_3
15. Lu, Q., Chen, T., Zhang, W., Yang, D., Yu, Y.: Serendipitous personalized ranking for top-n recommendation. In: Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, pp. 258–265. IEEE Computer Society, Washington, DC (2012)
16. Remer, T.G.: Serendipity and the Three Princes: From the Peregrinaggio of 1557. University of Oklahoma Press, Norman (1965)
17. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston (2011). doi:10.1007/978-0-387-85820-3_1
18. Sahebi, S., Brusilovsky, P.: Cross-domain collaborative recommendation in a cold-start context: the impact of user profile size on the quality of recommendation. In: Carberry, S., Weibelzahl, S., Micarelli, A., Semeraro, G. (eds.) UMAP 2013. LNCS, vol. 7899, pp. 289–295. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38844-6_25
19. Sang, J.: Cross-network social multimedia computing. User-centric Social Multimedia Computing. ST, pp. 81–99. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44671-3_5
20. Shapira, B., Rokach, L., Freilikhman, S.: Facebook single and cross domain data for recommendation systems. User Model. User-Adap. Inter. **23**, 211–247 (2013)
21. Tacchini, E.: Serendipitous mentorship in music recommender systems. Ph.D. thesis, Università degli Studi di Milano (2012)
22. Winoto, P., Tang, T.: If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? A study of cross-domain recommendations. New Gener. Comput. **26**, 209–225 (2008)
23. 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 (2012)
24. Zheng, Q., Chan, C.-K., Ip, H.H.S.: An unexpectedness-augmented utility model for making serendipitous recommendation. In: Perner, P. (ed.) ICDM 2015. LNCS, vol. 9165, pp. 216–230. Springer, Cham (2015). doi:10.1007/978-3-319-20910-4_16