Neural Serendipity Recommendation: Exploring the Balance between Accuracy and Novelty with Sparse Explicit Feedback

YUANBO XU, YONGJIAN YANG, EN WANG, and JIAYU HAN, Jilin University
FUZHEN ZHUANG, Key Lab of Intelligent Information Processing of Chinese Academy of Sciences and Beijing Advanced Innovation Center for Imaging Theory and Technology
ZHIWEN YU, Northwestern Polytechnical University
HUI XIONG, Rutgers, The State University of New Jersey

Recommender systems have been playing an important role in providing personalized information to users. However, there is always a trade-off between accuracy and novelty in recommender systems. Usually, many users are suffering from redundant or inaccurate recommendation results. To this end, in this article, we put efforts into exploring the hidden knowledge of observed ratings to alleviate this recommendation dilemma. Specifically, we utilize some basic concepts to define a concept, *Serendipity*, which is characterized by high-satisfaction and low-initial-interest. Based on this concept, we propose a two-phase recommendation problem which aims to strike a balance between accuracy and novelty achieved by serendipity prediction and personalized recommendation. Along this line, a Neural Serendipity Recommendation (NSR) method is first developed by combining Muti-Layer Percetron and Matrix Factorization for serendipity prediction. Then, a weighted candidate filtering method is designed for personalized recommendation. Finally, extensive experiments on real-world data demonstrate that NSR can achieve a superior serendipity by a 12% improvement in average while maintaining stable accuracy compared with state-of-the-art methods.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**; *Factorization methods*;

Additional Key Words and Phrases: Serendipity, recommender system, muti-layer percetron, matrix factorization

This work is supported by the National Natural Science Foundations of China under Grant No. 61772230, No. 61972450, and No. 61773361; Natural Science Foundation of China for Young Scholars No. 61702215; China Postdoctoral Science Foundation funded projects No. 2017M611322, No. 2018T110247, and No. BX20190140; and Changchun Science and Technology Development Project No.18DY005.

Authors addresses: Y. Xu, Y. Yang, E. Wang (corresponding author), and J. Han, Jilin University, Qianjin Street No. 2699, Changchun, Jilin, 130012, China; emails: yuanbox@jlu.edu.cn, {yyj, wangen}@jlu.edu.cn, jyhan15@mails.jlu.edu.cn; F. Zhuang, Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Beijing, 100190, China, Beijing Advanced Innovation Center for Imaging Theory and Technology, Capital Normal University, China; email: zhuangfuzhen@ict.ac.cn; Z. Yu, Northwestern Polytechnical University, the School of Computer Science, Xi'an, Shanxi, 710072, China; email: zhiwenyu@nwpu.edu.cn; H. Xiong, Rutgers, The State University of New Jersey, School of Business, Newark, NJ; email: hxiong@rutgers.edu.

1556-4681/2020/06-ART50 $15.00
https://doi.org/10.1145/3396607

ACM Reference format:
Yuanbo Xu, Yongjian Yang, En Wang, Jiayu Han, Fuzhen Zhuang, Zhiwen Yu, and Hui Xiong. 2020. Neural Serendipity Recommendation: Exploring the Balance between Accuracy and Novelty with Sparse Explicit Feedback. *ACM Trans. Knowl. Discov. Data* 14, 4, Article 50 (June 2020), 25 pages.
https://doi.org/10.1145/3396607

1 INTRODUCTION

In the past decade, recommender systems (RSs) have played an important role in improving users' experience on e-commerce, apps of mobile phones, and the like [C. He et al. 2016; Liu et al. 2016; Zhu et al. 2014]. RSs try to utilize the known information, such as users' profile, context, and user-item ratings to generate users' preferences accurately [Bobadilla et al. 2013; Gomez-Uribe and Hunt 2016; Yu et al. 2016a]. In other words, RSs can filter the vastly growing explosion of information on the Internet and provide insights for finding specific, focused products and services among millions of items [C. He et al. 2016].

However, how to evaluate a recommendation is still a critical and challenging problem for researchers [Lee and Hosanagar 2016]. Some metrics have been proposed to evaluate the effectiveness of RSs, such as accuracy [Kaminskas and Bridge 2016], diversity [Cheng et al. 2017] and novelty [Bobadilla et al. 2013]. Accuracy is to evaluate whether RSs can accurately match users' preferences by extracting from their historical interactions. Diversity usually denotes the differences among recommended items, which can enhance the range of recommendation. And novelty is always applied to some special RSs to tackle the cold-start problem. However, many existing RSs based on accuracy are not capable of giving impressive recommendations and those based on diversity or novelty may frustrate users due to their chaotic recommendations. To the best of our knowledge, there is no well-defined metric to balance both accuracy and novelty to evaluate a RS.

Let us consider an example in Figure 1. The table in Figure 1 is a known rating matrix and "null" means system does not get the information. *a*-*b* means item *a* and *b* are in the same category. Now, we have four candidates (i5, i6, i7, and i8) to recommend for u2. If we recommend i5 that u2 may be interested in (i5 is in same category of i3), like many existing interest-based methods, a high accuracy could be achieved. Existing models do not care if u2 will give a high or low rate on i5, they just recommend i5 to u2, because u2 has bought i3. However, u2 is not satisfied with i3 (rated only 1 score), which means recommending i5 from the same category will create a risk of low satisfaction, and degrade u2's experience. In order to maximize diversity, we may recommend all four items for u2. But in this way, RS will lose its ability to filter overload information. Otherwise, if recommending a novel item based on some Collaborate Filtering (CF)-based or context-based methods, like i6, or a brand-new item like i7, we also need to consider the accuracy to evaluate which is better. All these considerations arouse a new dilemma—how to trade-off the accuracy and novelty for evaluating RSs.

As can be seen in Figure 1, for i4, there are two different situations: (1) u2 may scan i4 without buying it, or (2) u4 is just a brand-new item to u2. Both situations can not indicate that u2 is interested in i4. However, the users u1 and u3 bought i4 and gave relatively high scores, so i8 in the same category may have the potential to satisfy u2. Moreover, according to u2's purchasing historical records, it is novel to recommend i8 to u2. Therefore, i8 may be the best choice in this recommendation dilemma, which has a relatively low interest but high potential satisfaction for u2. This recommendation not only increases the category u2 may consume, but also creates more value than i5. So if a man has bought a tire and a nursing bottle, why don't we recommend a babyseat rather than another nursing bottle or a phone for him? Here comes a good idea, if we can recommend the items with relatively low interest but high potential satisfaction for users, the RS should be more

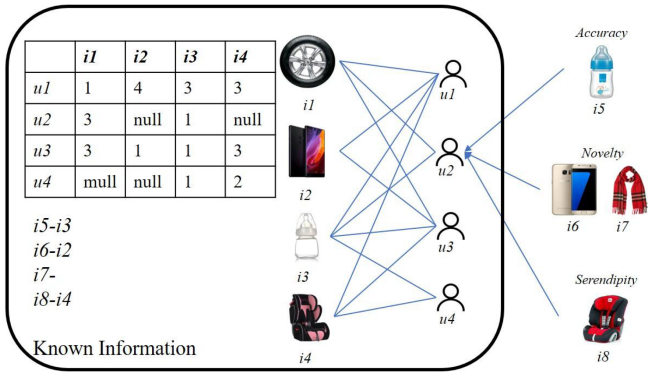


Fig. 1. An example to illustrate recommendation dilemma between accuracy and novelty.

Table 1. The Differences Among Different Concepts

Concepts	Interest	Satisfaction	Number of Category	Created Value
Accuracy	High	Unknown	Small	Low
Novelty	Low	Unknown	Large	High
Diversity	Unknown	Unknown	Large	Low
Serendipity	Low	**High**	Large	**High**

valuable than only with interest or satisfaction, like traditional RS does. To sum up, the differences among different concepts (Accuracy, Novelty, Diversity, and Serendipity) are shown in Table 1:

In this article, to address the recommendation dilemma above, we introduce a well-defined concept: Serendipity, which is characterized by high-satisfaction and low-interest. Particularly, based on this concept, we define a two-phase problem including Serendipity Prediction and Personalized Recommendation. Along this line, a Neural Serendipity Recommend (NSR) method is developed. Specifically, we combine Multi-Layer Perceptron (MLP) and Matrix Factorization (MF) to predict serendipity, where MF is employed to ensure the accuracy of recommendation while MLP is to capture the novelty. Furthermore, we propose a candidate filtering method, which can maximize the knowledge acquirement from predicted serendipity, alleviate the data sparsity problem and make a personalized recommendation. Finally, extensive experiments are conducted on Movielens, Yelp, and BookCrossing to demonstrate the effectiveness of NSR in comparison with state-of-the-art baselines. Our unique contribution is that NSR is a machine-optimizable measure for serendipity, so that it becomes possible to develop machine learning algorithms to directly optimize the serendipity of recommendation results.

Our main contributions are summarized as follows:

—We investigate the recommendation dilemma and the trade-off between accuracy and novelty. In order to tackle this dilemma, a well-defined concept: Serendipity and a two-phase recommendation problem (Serendipity Prediction and Personalized Recommendation) are proposed. We also define a machine-optimizable measure for serendipity.
—We propose NSR, a Neural Serendipity Recommendation method to tackle the recommendation problem. In NSR, MLP and MF are combined to predict serendipity, then a candidate filtering method is proposed to make a personalized recommendation.

— We introduce a novel serendipity metric and conduct extensive experiments on one standard dataset (Movielens) and two large-scale real-world datasets (Yelp and BookCrossing). The effects of different parameters are also discussed in detail. The results demonstrate that NSR can improve the serendipity performance (nearly 12% in average) with a stable accuracy as compared to the state-of-the-art baselines.

The rest of this article is organized as follows: Section 2 shows the related works of this article. Section 3 identifies the basic definitions of our problems. In Section 4, we introduce the details of the NSR approach. Experiments on different datasets to evaluate the proposed method as well as an analysis of the experimental results are reported in Section 5. Finally, we briefly conclude the article in Section 6.

2 RELATED WORK

In this section, we introduce recent researches from three aspects: RS, Metrics for RS, and Neural Network (NN).

2.1 Recommender System

In this part, we briefly introduce two related recommendation methods: CF-based methods and MF-based methods [Bu et al. 2016; Li et al. 2019; Xiong et al. 2018; Xu et al. 2017; Yan et al. 2017; Yang et al. 2015; Yu et al. 2016b].

CF-based method is the most popular method applied by many e-commerce companies. CF-based method utilizes the relationships between either users or items to make recommendations, which is called user-based or item-based method [Sarwar et al. 2001; Sun et al. 2015; Wang et al. 2017; Ying et al. 2018; Zhang et al. 2016; Zhuang et al. 2017]. The relationship can be measured through Euclidean distance or Pearson Correlation of the vectors in the user-item matrix. Some researchers focus on how to improve the performance of basic CF-based methods. [Hwang et al. 2016] utilizes users' interests as heuristic information and increases the weight of users' interesting items in unrated items, which gains a better recommendation. Also, some works show that item-based CF is better than user-based CF in some real-world scenarios [Sarwar et al. 2001]. Knowledge-based method (K-based method) is a heuristic method which utilizes users' profile, context, or multi-view data to make efficient and accurate recommendations [Adomavicius and Tuzhilin 2015]. [Lee and Hosanagar 2016] takes a deep consideration into which attribute in these multi-view data source should be used to gain the best recommendation performance. Some RSs use users' context as basic information and transfer the recommendation problem into a regression problem [Adomavicius and Tuzhilin 2015].

MF-based method addresses the relationship between users and items only from a user-item matrix [X. He et al. 2016; Rendle et al. 2009]. These methods utilize supervised MF to map users and items into a lower-dimension space and represent users and items respectively. Finally, MF-based method multiplies user vectors and item vectors to get the ratings for unrated items, as SVD and SVD++ [Zhang et al. 2005]. e-ALS [X. He et al. 2016] is a fast MF method to deal with the cold start and efficiency problem in traditional MF-problem. BPR [Rendle et al. 2009] employs a pair-wise loss function, tries to maximize the distance between observed items and unobserved items in order to make an accurate recommendation. MF-based method is very effective to cope with the linear-like predictions. However, in complicated real-world RS, the problem is always non-linear, which harms the performance of MF-based method exceedingly. [X. He et al. 2016d] proposes a fast MF method to tackle non-linear problem. Furthermore, [He et al. 2017] explores the non-linear problem and builds a general framework to cover MF and CF. Although there are many improved recommendation methods, they are usually not designed for diversity and novelty

specifically [Bu et al. 2016; Kotkov et al. 2016b; Zhou et al. 2017], while our proposed method is designed to balance serendipity and accuracy, which is an improvement in different aspect from other recommendation methods.

2.2 Metrics for Recommender System

Basically, there are three metrics to evaluate recommender system: accuracy, diversity, and novelty.

Accuracy is always a significant metric of recommendation system. The most common metrics applied in RSs are Precision (P), Recall (R), and Normalized Discounted cumulative gain (NDCG) [Bobadilla et al. 2013]. P denotes the proportion of correct items in recommendation list. R denotes the proportion of items in all correct items. Usually, R and P cannot reinforce each other, so we utilize F-measure to make the trade-off. Besides, NDCG is an important metric for a rank sequence of recommendation list [Rendle et al. 2009]. However, if a RS makes the recommendation too accurate, it may fall into an "overfitting," which means it only recommends "safe" items and loses the ability to provide users the various choices of unobserved items [Kaminskas and Bridge 2016].

Diversity is another important metric. It means to recommend users with different types of items [Cheng et al. 2017; Gogna and Majumdar 2017; Ziegler et al. 2005]. And it can be viewed from average diversity and single diversity [Cheng et al. 2017]. Average diversity means utilizing the long-tail effect in a recommendation list while single diversity means maximizing the distance between each pair items in a recommendation list. Some researchers use different methods to gain better diversity with stable accuracy. [Cheng et al. 2017] treats diversified recommendation as a supervised learning task, defines two coupled optimization problems, and proposes a diversified collaborative filtering algorithm to solve both problems. [Gogna and Majumdar 2017] utilizes a matrix completion to balance accuracy and diversity. These improved methods surely level up the representation of RS in diversity, but they usually do not consider deep relationships between diversity, accuracy and users' interests.

Recently, serendipity and novelty have attracted researchers' attention as a concept to evaluate RS [Kaminskas and Bridge 2016; Kotkov et al. 2016a, 2016c, 2016b; Yang et al. 2018; Zhou et al. 2017; Ziegler et al. 2014]. Basically, serendipity and novelty consist of three aspects: originality, interest, and unexpectedness [Zhou et al. 2017]. Originality means that the item should be brand-new to users, different from the items users have consumed. Interest ensures the accuracy because users always buy the items they are interested in. Finally, unexpectedness enables the user-never-known item to be recommended, which is able to stimulate the purchase desire of the users. Recent years saw an increasing attention on novelty and some novelty-oriented methods have been proposed to tackle the problem. [Zhou et al. 2017] proposes a theory-based algorithm to capture the information in context and make a novelty recommendation. And [Kotkov et al. 2016c] explores the balance between accuracy and novelty. Moreover, TRECS [Ziegler et al. 2014] is a hybrid algorithm to increase the novelty of RS and overperforms the baselines. The key issue is that serendipity is difficult to define and hard to get the ground truth to validate the serendipity, while our work just focuses on this aspect.

2.3 Neural Networks

The concept of NN has been proposed for several decades [Specht 1990]. NN is a kind of data processing architecture inspired by bioinformatics. Yet NN's development was slow as a result of the limit of computing ability. However, in recent five years, deep learning becomes the major tendency of NN. Deep learning learns multiple levels of representations and abstractions from data, which can solve both supervised and unsupervised learning tasks [Deng et al. 2014; Goodfellow et al. 2014; Xu et al. 2019]. Consequently, we treat deep learning as an extension of NN.

Recently, different frameworks of NN have been developed: MLP [He et al. 2017] is a feed-forward NN with several hidden layers between an input layer and an output layer [Deng et al. 2014]. It's the most common and effective NN framework which can be applied to tackle non-linear problems. Autoencoder is an unsupervised model which reconstructs its input data in the output layer. There are plenty of variants of autoencoders such as denoising autoencoder, marginalized denoising auto-encoder, sparse auto-encoder, contractive auto-encoder and variational auto-encoder [Chen et al. 2012]. Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) [Goodfellow et al. 2016] are the most popular NN frameworks. CNN is a feed-forward NN with pooling operations and several convolutional layers, while RNN is a recycling framework for sequential data. Meanwhile some more delighted NN framework has been proposed, such as GAN [Goodfellow et al. 2014] and GNN [Wu et al. 2019] Some researches also utilize NN to cover some basic recommendation methods, like NCF [He et al. 2017], and it can achieve the state-of-the-art performance.

Designers of RSs always focus on how to improve users' satisfactions with the recommendation experience. How to construct users' preference is a significant task, as the recommendation dilemma shown in Figure 1. From a long-term perspective, it shows that users' preferences are linear, which ensures the accuracy. However, many existing methods ignore that users' preferences will be shifted because of their experience. If they get a bad consuming experience, they will give a low rating and lose the interest in certain kind of items. Different ratings and different factors that affect users' preference make the novelty problem a complex non-linear one. Based on this fact, we propose NSR, which employs MF to capture the accuracy of users' preference, and NN to model the novelty part to solve the problem.

3 SERENDIPITY PROBLEM DEFINITION

Some basic definitions about serendipity problem in RS are introduced in this section. First, we introduce a well-defined concept: Serendipity. Then a two-phrase recommend problem based on Serendipity is also proposed. At last, we take insight into this problem and give some institutions.

3.1 Basic Definition

We identify a recommendation problem by treating serendipity as a character combining users' interests and satisfactions. Then we utilize machine learning techniques to train the model, select the candidate set and make recommendations.

In RS, let U, I be the sets of users and items respectively, $|U| = m, |I| = n$. User-item matrix $R \in \mathbb{R}^{m \times n}$ consists of m users, n items. I_u^r is a set of rated items of user u whose size is much smaller than n, $|I_u^r| << n$. And r_{ui} stands for the user u's rating of item i.

In our method, users' preferences on items can be modeled by *Interests* and *Satisfactions*.

Interests are the incentive extent (r^{in}) whether users want to purchase an item.

Satisfactions are the experience extent (r^{sa}) after users' purchasing.

In user-item matrix R, we treat every rated item $i \in I^r$ as users' Interest item, $I^{in}, r^{in} = 1$, but only parts with relative high rating level should be subjected to Satisfaction Item, $I^{sa}, r^{sa} = 1$. For items in unrated items I^{ur}, we need to predict the value of r^{in}, r^{sa} and use thresholds θ to screen the item set. All the notations are shown in Table 2.

3.2 Problem Definition

Existing methods which focus on high-interest items always lead to a narrow and bland recommendation of items. Consequently, as a more important metric than interest, satisfaction is often neglected. Based on this, we utilize Interest and Satisfaction to define Serendipity, as follows:

Table 2. Notations in this Article

Notation	Description
U	User set in our RS
I	Item set in our RS
m, n	user's quantity and item's quantity
k	latent embedding dimension
r_{ui}	User u's ratings on item i
r^{in}/r^{sa}	Interest/satisfaction extent of items
r^{se}	Serendipity label of items
R	User-item matrix with ratings r_{ui}
R^{in}	An Interest matrix whose entry is r_{in}
R^{sa}	A Satisfaction matrix whose entry is r_{sa}
R^c	An enhanced matrix whose entry is r_{sa}
I_u^r	A set of rated items for user u
I_u^{ur}	A set of unrated items for user u
I^{can}	A set of candidate items which will be evaluated for user u
I_u^{rec}	A set of recommended items for user u
α, β, γ	parameters of MLP, WMC and ECS
θ^{in}, θ^{sa}	thresholds of Interests and Satisfactions
τ	parameter of embedding layer
η	hyper parameter

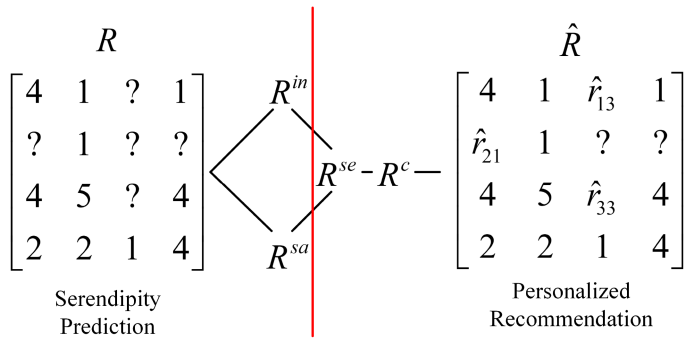


Fig. 2. Two-phrase Serendipity Recommendation Problems.

Serendipity denotes high Satisfaction (r^{sa}) but low Interest (r^{in}) in I^{ur}. And we want to utilize this concept to predict Serendipity (r^{se}), pick candidate item set in I^{ur}, thereby increasing the expression of original ratings in R and enhancing the quality of recommendation.

Basically, our recommendation task is a two-phase case, which is shown in Figure 2: named, Serendipity Prediction and Personalized Recommendation.

Problem 1 (Serendipity Prediction). *Given a user-item matrix R, with rating items I^r and unrated I^{ur}, how to predict r^{in}, r^{sa} and decide r^{se} in I^{ur} with the only constraint of ratings r in I^r becomes a problem.*

Generally, we transfer R into two matrices: Interest Matrix R^{in}, whose entries are $r^{in} \in (1, null)$ and Satisfaction Matrix R^{sa}, whose entries are $r^{sa} \in (0, 1, null)$. It's intuitive that we can use MF to do Matrix Completion, respectively. However, utilizing MF respectively neglects the relationships between Interests and Satisfactions. For example, if a user was not satisfied with a specific item, he would give

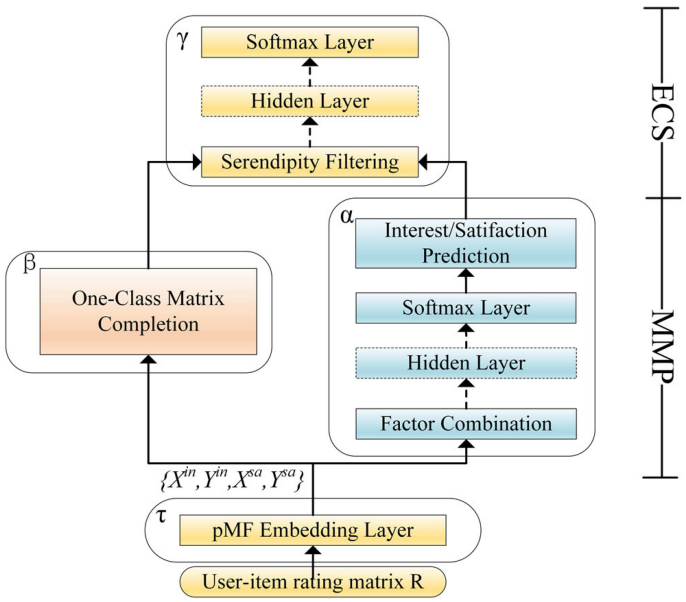


Fig. 3. Framework of NSR.

a low rating, and lost the interest in similar items. Otherwise, a user's Interest in similar items will increase if he is satisfied with the thing he bought. Interest and Satisfaction reinforce mutually, which makes Serendipity Prediction a complex problem that cannot be solved directly by MF. NN has a strong ability to tackle with the complex non-linear problem. So we make an attempt to combine Multi-Layer Percetrion and MF to face this complex recommendation problem.

Problem 2 (Personalized Recommendation). *Given a user-rated item matrix R, and prediction of r^{in}, r^{sa}, how to screen serendipity items and pick candidate item set I^{can} from unrated items I^{ur} and give each user a recommendation list I_u^{rec} becomes a problem.*

In Personalized Recommendation, we have got all the $r^{in} \in R^{in}$, $r^{sa} \in R^{sa}$ for all unrated items in I^{ur}. There are two key problems we need to solve in this phase: the first problem is how to use these predictions to filter the serendipity items R^{se} in I^{ur}. It's difficult to find a proper threshold to screen the unrated item. If the chosen threshold is too large or small, the system will lose the ability to find the serendipity items because all the unrated items will be treated as un-serendipity or serendipity in the same way. Then it will decay to an item-based MF RS. The second one is how to apply personalized recommendation in different scenarios. Sometimes we need to make an accurate recommendation for a preference-stable user but a more serendipitous recommendation for a risky-spirit user.

4 THE NSR ALGORITHM

NSR utilizes *MMP*, which combines *MLP* and *MF* to predict Serendipity, and ECS, an *Enhanced Candidate Selection* method to tackle Personalized Recommendation problem. The whole framework of NSR is shown in Figure 3.

4.1 Data Pre-processing

In data pre-processing, first, user-item matrix R is transferred into two similar dimension matrices: Interest Matrix R^{in} and Satisfaction Matrix R^{sa}. Items in I^r can be treated as Interest item, $r^{in} = 1$,

and items in I^{ur} are unobserved, $r^{in} = null$. So Interest Matrix R^{in} is defined as follows:

$$R^{in} = \{r^{in} | r^{in}_{ui} = 1/null, i \in I^r / I^{ur}\}, R^{in} \in \mathbb{R}^{m \times n} \tag{1}$$

Only relative high-rating items in I^r should be treated as Satisfaction Item. We use a threshold θ^{sa} to control the proportion of Satisfaction Item in I^r. For each user u, θ^{sa} is personalized because of their different ranking habits, $\theta^{sa}_u = \sum^{i \in I^r_u} r_{ui} / |I^r_u|$. In accordance with Interest Matrix, Satisfaction Matrix R^{sa} is defined as follows:

$$\begin{aligned} R^{sa} = \{r^{sa} | r^{sa}_{ui} = 1, r_{ui} > \theta^{sa}_u; r^{sa}_{ui} = 0, r_{ui} < \theta^{sa}_u; \\ r^{sa}_{ui} = null, i \in I^{ur}\}, R^{sa} \in \mathbb{R}^{m \times n} \end{aligned} \tag{2}$$

NSR employs MF to embed R^{in} and R^{sa} into a k-dimensional latent space. We introduce parameter τ^{in} and τ^{sa} as parameters to represent the multiple possible factorization results, like parameterized Matrix Factorization (pMF). Given a user-item matrix R, and two sequences of parameters, $\tau^{in} = \{\tau^{in}_1, \tau^{in}_2, ..., \tau^{in}_k\}$ and $\tau^{sa} = \{\tau^{sa}_1, \tau^{sa}_2, ..., \tau^{sa}_k\}$, users and items in user-item matrix R can be mapped into lower rank matrices R^{in}, R^{sa}:

$$R^{in} \approx \left(X^{in}\right)^T \Sigma^{in} Y^{in}; X^{in} \in \mathbb{R}^{k \times m}, Y^{in} \in \mathbb{R}^{k \times n} \tag{3}$$

$$R^{sa} \approx (X^{sa})^T \Sigma^{sa} Y^{sa}; X^{sa} \in \mathbb{R}^{k \times m}, Y^{sa} \in \mathbb{R}^{k \times n} \tag{4}$$

where Σ^{in}, Σ^{sa} are k-diagonal matrices whose entries are τ^{in} and τ^{sa}.

4.2 MMP for Serendipity Prediction

MMP combines MF and MLP to address Serendipity Prediction. Obviously, Serendipity Prediction problem can be tackled as one-class matrix completion problem, which means we need to mark all the items in I^{ur} whether they are serendipity or not. And as we have introduced in the former section, we just need to predict Interest and Satisfaction for each item, and pick the ones with high Satisfaction and low Interest as Serendipity items.

This problem consists of an accuracy part and a novelty part. It is obvious that users' Interests and Satisfactions maintain a consistency. People's preferences are represented by their former interactions with a RS. Therefore, we employ MF to tackle the accuracy part of serendipity problem. However, Interests and Satisfactions affect each other and users' preferences may be shifted by their interactions. This shift makes the problem more complicated and appropriately non-linear. And we employ MLP to cope with this novelty part of serendipity problem.

4.2.1 Weighted Matrix Completion for Accuracy. We use Weighted Matrix Completion to tackle the accuracy part of Serendipity Prediction. It is a typical one-class matrix completion problem because Interest and Satisfaction for each item is (0, 1). The reason why we use MF-based model for accuracy is that: first, the accuracy prediction is a linear problem which can be solved by MF models or NN-based models. Second, there are some existing models which utilize the MF model to address the accuracy problem and achieve a good performance. Finally, consider the cost of NN-based models and the limited improvement on accuracy, we employ MF model to tackle the accuracy prediction. So we define loss functions as follows:

$$L^{WMC}\left(r^{in}_{ui}, \hat{r}^{in}_{ui}\right) = \sum_{u \in U} \sum_{i \in I^r_u} w^{in}_{ui} \left(r^{in}_{ui} - \hat{r}^{in}_{ui}\right)^2 \tag{5}$$

$$L^{WMC}\left(r^{sa}_{ui}, \hat{r}^{sa}_{ui}\right) = \sum_{u \in U} \sum_{i \in I^r_u} w^{sa}_{ui} \left(r^{sa}_{ui} - \hat{r}^{sa}_{ui}\right)^2 \tag{6}$$

Note that the weights decide the importance of different examples, so we set the weights as follows:

$$w_{ui}^{in} = \frac{|I^r - I_u^r|}{|I^r|} + \frac{|U_i|}{|U|}; r_{ui}^{in} = 1 \tag{7}$$

$$w_{ui}^{sa} = \begin{cases} \frac{|I^r - I_u^r|}{|I^r|} + \frac{|U_i|}{|U|} + \frac{|I_u - I^{sa}|}{|I_u|}; r_{ui}^{sa} = 1 \\ \frac{|I^r - I_u^r|}{|I^r|} + \frac{|U_i|}{|U|} + \frac{|I^{sa}|}{|I_u|}; r_{ui}^{sa} = 0 \end{cases} \tag{8}$$

The first two parts are similar for w^{in} and w^{sa}, which consider the item popularity and the user popularity. The more popularity an item has, the less importance it gets. It's obvious that if a user who ranks three items only, every item's effort is much more important, and vice versa. The third part of Equation (8) is special for Satisfaction, which means that if a user has more satisfaction items, the importance of satisfaction items is less than un-satisfaction ones. It's a trade-off between an item's local and global popularity.

With the latent representations X^{in}, Y^{in}, X^{sa}, Y^{sa} of users and items, we are able to rewrite the loss function as follows:

$$L^{WMC}(X^{in}, Y^{in}) = \sum_{u \in U} \sum_{i \in I_u^r} \frac{w_{ui}^{in}}{2} \left(r_{ui}^{in} - \left(x_{ui}^{in}\right)^T y_{ui}^{in}\right)^2 + \frac{\lambda^{in}(\delta^{in})}{2} \tag{9}$$

$$L^{WMC}(X^{sa}, Y^{sa}) = \sum_{u \in U} \sum_{i \in I_u^r} \frac{w_{ui}^{sa}}{2} \left(r_{ui}^{sa} - \left(x_{ui}^{sa}\right)^T y_{ui}^{sa}\right)^2 + \frac{\lambda^{sa}(\delta^{sa})}{2} \tag{10}$$

Where λ^{in} and λ^{sa} are regularization parameters, δ^{in} and δ^{sa} denote the Frobenius 2-norm to avoid over-fitting. We apply a weighted alternating least squares (wALS) method to solve this low-rank approximation problem. Taking Interest as an example, iterative formulas of X and Y are as follows:

$$X_{u.}^{in} = R_{u.}^{in} W_{u.}^{in} Y^{in} \left(\left(Y^{in}\right)^T W_{u.}^{in} Y^{in} + \lambda^{in} \sum_i w_{ui}^{in} I\right)^{-1} \tag{11}$$

$$Y_{.i}^{in} = R_{.i}^{in} W_{.i}^{in} X^{in} \left(\left(X^{in}\right)^T W_{i.}^{in} X^{in} + \lambda^{in} \sum_u w_{ui}^{in} I\right)^{-1} \tag{12}$$

Where $W_{u.}^{in}$, $W_{.i}^{in}$ are diagonal matrices whose entries are $w_{u.}^{in}$ and $w_{.i}^{in}$ on the diagonal and I is a $k \times k$ identity diagonal matrix. For Satisfaction, iterative formulas are the same as Interest. We use β to represent other parameters in WMC. Therefore, we make the Serendipity Prediction with a weighted matrix completion introduced above, as shown in Algorithm 1:

With this method, we get Prediction Matrix $\hat{R}^{in}/\hat{R}^{sa} \in \mathbb{R}^{(m \times n)}$, whose entries are between $[0, 1]$.

4.2.2 Multi-Layer Perceptron for Novelty. We use Multi-Layer Perceptron to capture the novelty part of serendipity prediction.

As we discussed before, embedding layer can be seen as performing the latent factor model of Interests and Satisfactions. In order to model the complicated interactions between users and items, Interests and Satisfactions, we use one-hot encoding, o_u and o_i as input control vectors. We express the predictions of Interests and Satisfactions as follows:

$$\left\{\tilde{r}_{ui}^{in}, \tilde{r}_{ui}^{sa}\right\} = h(X^{in} o_u, X^{sa} o_u, Y^{in} o_i, Y^{sa} o_i \,|\tau, \alpha) \tag{13}$$

ALGORITHM 1: Weighted Matrix Completion for Accuracy

Input: User latent matrices X^{in}, X^{sa}/Item latent matrices Y^{in}, Y^{sa}, User-item matrix R, parameters τ^{in}, τ^{sa}
Output: Prediction Matrix $\hat{R}^{in}/\hat{R}^{sa}$, WMC Embedding parameter $\hat{\tau}^{in}/\hat{\tau}^{sa}$, WMC parameter β
Calculate W^{sa} with Equation (8)
repeat
  Update $X^{in}_{u.}, \forall u$ with Equation (11),
  Update $Y^{in}_{.i}, \forall i$ with Equation (12),
  Update τ^{in}, β
until *convergence*;
repeat
  Update $X^{sa}_{u.}, \forall u$ with Equation (11)
  Update $Y^{sa}_{i.}, \forall i$ with Equation (12)
  Update τ^{sa}, β
until *convergence*;
Calculate $\hat{r}^{in}_{ui}=(X^{in}_{u.})^T(Y^{in}_{i.})$,
Calculate $\hat{r}^{sa}_{ui}=(X^{sa}_{u.})^T(Y^{sa}_{i.})$,
Output $\hat{R}^{in}/\hat{R}^{sa}$, $\hat{\tau}^{in}/\hat{\tau}^{sa}$, β

where $X \in \mathbb{R}^{k\times m}$, $Y \in \mathbb{R}^{k\times n}$. τ denotes the parameter of embedding layer and α denotes all the parameters in MLP. Different τ controls the different embedding results.

Traditional methods often employ element-wise products to predict ratings, which is better to measure the accuracy problem. But in our problem, it is more non-linear-like because different users, different items, and different ratings affect each other. In order to model complex interactions between users, items, Interests and Satisfactions, we use direct vector concatenation instead of element-wise products. Then we feed $[X, Y] = [X^{in}o_u, X^{sa}o_u, Y^{in}o_i, Y^{sa}o_i]$ into multiple hidden layers of neural structure $\mathbb{H}$. The p-th hidden layer is denoted as h^p, which is a non-linear function of former hidden layer h^{p-1}. $\mathbb{H}$ employs $ReLU(x) = max(0, x)$ as activation function:

$$h^p(X, Y) = ReLU(W^p h^{p-1}(X, Y) + b^p) \tag{14}$$

W^p and b^p are the parameters of the p-th hidden layer. Above all, we get the formulation of MLP:

$$\left\{\tilde{r}^{in}_{ui}, \tilde{r}^{sa}_{ui}\right\} = h^{out}(h^P(h^{p-1}(...h^2(h^1(X, Y))))) \tag{15}$$

P is the number of hidden layer of MLP. Predicting Interests and Satisfactions of unrated items is a typical one-class problem. So on the top of hidden layers, we use sigmoid activation function as the output layer h^{out}. With this MLP structure, we finally achieve the prediction of novelty part.

In order to learn the parameter τ, α of MLP, we employ a log loss function, which is cross-entropy-style but special for an output of Interests and Satisfactions. We construct this supervised loss function as follows:

$$\begin{aligned} L^{MLP} &= \log p(\mathbb{H}|\tau, \alpha) = L\left(r^{in}_{ui}, \tilde{r}^{in}_{ui}\right) + L\left(r^{sa}_{ui}, \tilde{r}^{sa}_{ui}\right) \\ &= -\sum_{(o_u, o_i)\in R} r^{in}_{ui} \log \tilde{r}^{in}_{ui} + \left(1 - r^{in}_{ui}\right) \log\left(1 - \tilde{r}^{in}_{ui}\right) \\ &\quad - \sum_{(o_u, o_i)\in R} r^{sa}_{ui} \log \tilde{r}^{sa}_{ui} + \left(1 - r^{sa}_{ui}\right) \log\left(1 - \tilde{r}^{sa}_{ui}\right) \end{aligned} \tag{16}$$

Note that our loss function consists of two different parts: Interest Loss and Satisfaction Loss. The reason is that for Interest, we need to sample negative user-item pair from unrated items because there is no negative examples in R^{in}. However, for Satisfaction, we don't need to do negative

sampling because $r^{sa} \in (1, 0, null)$. We uniformly sample negative pairs for Interest to with the consideration of training time and data scale.

The process of MLP for serendipity prediction is shown in Algorithm 2:

ALGORITHM 2: MLP for Novelty

Input: User latent matrices X^{in}, X^{sa}/Item latent matrices Y^{in}, Y^{sa}, User-item matrix R, parameters τ^{in}, τ^{sa}
Output: MLP embedding parameter $\tilde{\tau}^{in}/\tilde{\tau}^{sa}$, MLP parameter α
Combine $X^{in}, X^{sa}, Y^{in}, Y^{sa}$ into (X, Y),
Feed (X, Y) into Neural Networks,
repeat
  Compute L^{MLP},
  Compute the gradient,
  Update α, τ,
until *convergence*;
$\tilde{\tau}^{in}/\tilde{\tau}^{sa} = \tau$
Output$\tilde{\tau}^{in}/\tilde{\tau}^{sa}$, α

After constructing MLP, we make predictions as well as we did in Algorithm 1.

Specifically, the combination of interest and satisfaction is helpful to improve the novelty in recommendations. First, in this MLP, we explore users' latent preference about the items with two metrics: interest and satisfaction. If a user wants to buy a brand-new item, which is not similar to the items he has bought before, a RS should explore the relationship between this new item and the items he has bought. This relationship is so complicated that traditional models (linear prediction or MF models) can not tackle it. However, with our proposed model with MLP, we can explore the latent relationship in this situation, which can improve the novelty of recommendations with our lose accuracy.

Finally, we get prediction matrices $\tilde{R}^{in}/\tilde{R}^{sa}$, whose entries are also between $[0, 1]$.

4.3 ECS for Personalized Recommendation

After the first phase, we got accuracy prediction $\hat{R}^{in}/\hat{R}^{sa}$ and the novelty prediction $\tilde{R}^{in}/\tilde{R}^{sa}$. ECS is applied to decide the candidate set in unrated items, and make a personalized recommendation.

4.3.1 Candidate Decision. First, we combine the accuracy prediction and novelty prediction, and use two thresholds θ^{in}, θ^{sa}. We can change the thresholds to personalize the system, as shown in Section 5 to filter the unrated items. The accuracy part is to tackle with the consistency of users' preference while the novelty part captures the complex relationships between users and items, Interests, and Satisfactions. So we utilize $\omega \in (0, 1)$ to control the weight of both parts:

$$\bar{r}_{ui}^{in} = \omega \hat{r}_{ui}^{in} + (1 - \omega)\tilde{r}_{ui}^{in} \tag{17}$$

$$\bar{r}_{ui}^{sa} = \omega \hat{r}_{sa}^{in} + (1 - \omega)\tilde{r}_{sa}^{in} \tag{18}$$

We need to deeply understand which item should be considered as a candidate item in unrated items. First, as many interest-based methods suggested, an item of high-Interest is perfect to be a candidate and it ensures the basic accuracy of the recommendation. However, we cannot ignore serendipity items with low-Interest but high-Satisfaction, which have the potential to enhance users' experience and create more value. It's the reason why many methods failed with high accuracy but low novelty.

In our proposed method, we choose three types of items as candidate items: high-interest & high-satisfaction, high-interest & low-satisfaction, and low-interest & high-satisfaction (serendipity items). In NSR, we select the candidate set using the label $\bar{r}$ with two thresholds θ^{in}, θ^{sa}, as follows:

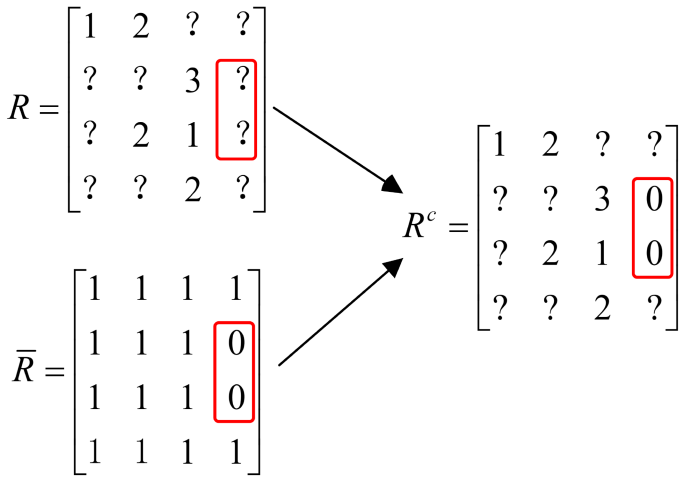


Fig. 4. Candidate decision.

$$\bar{r} = \begin{cases} 1; \bar{r}_{ui}^{in} > \theta^{in}, \bar{r}_{ui}^{sa} > \theta^{sa} \\ 1; \bar{r}_{ui}^{in} > \theta^{in}, \bar{r}_{ui}^{sa} < \theta^{sa} \\ 1; \bar{r}_{ui}^{in} < \theta^{in}, \bar{r}_{ui}^{sa} > \theta^{sa} \\ 0; \bar{r}_{ui}^{in} < \theta^{in}, \bar{r}_{ui}^{sa} < \theta^{sa} \end{cases} \tag{19}$$

We build candidate mark matrix $\bar{R}$ with entries $\bar{r}$, noting that every item in I^r is marked by 1.

Furthermore, we treat the unrated items whose mark $\bar{r} = 1$ as the candidate set I^{can}. Instead of picking candidate item, we use a negative injection [He et al. 2018; Xiao et al. 2017] to increase the data density of original matrix R, which is able to alleviate the data sparsity problem and enhance the ability of representing information. With all the considerations above, we build an enhanced user-item matrix R^c. As shown in Figure 4, The sparsity of the original user-item matrix R is (16-6)/16 = 62.5%. But after utilizing negative injection, we can gain more information about unknow items and filter the candidate set. the sparsity of R^c is (16-8)/16 = 50%, which relieves the sparsity problem significantly.

4.3.2 Personalized Recommendation. Enhanced user-item matrix R^c can be applied to various recommender methods, such as CF, user-based MF, or Bayes Personalized Ranking (BPR) to improve the recommendation quality. That is why our method is applicable to many scenarios and methods. However, noting that compared with basic user-item matrix R, it increases the number of negative examples. So we employ General Matrix Factorization (GMF) [He et al. 2017] as our method to make recommendations. We have made some comparisons in our experiment between different recommend methods. Basically, GMF is a NN which utilizes the positive and negative examples to make a prediction. We set the GMF with tower-structure 64-32-16-8, softmax output layer and some proper hyper-parameters.

In GMF framework, we also need to do some MF for R^c as we did in Section 4.1. $R^c \approx P_u Q_i$, then we employ direct vector concatenation instead of element-wise products to combine P_u, Q_i, feeding to GMF as input. Finally, we get a recommendation list I_u^{rec} consist of items with Top-k outputs of GMF.

$$I_u^{rec} = GMF(\bar{R}|\gamma) \tag{20}$$

Where γ is the parameter of GMF.

Table 3. Datasets Statics

	ML100k	ML1M	Yelp	Book-Crossing
m(users)	943	6,040	43,873	278,858
n(items)	1682	3,900	11,537	271,379
r(ratings)	100,000	1,000,209	229,907	11,497,80
sparsity	6.79%	4.24%	1.30%	0.0015%

To make the recommendation personalized, we could use three strategies: (1) Change the threshold θ. θ is employed to control the proportion of Interests and Satisfactions Prediction. If we want to build an Interest-Focused RS, we can set $\theta^{sa} = 1$, which cancels the ability of satisfaction screen, and vice versa. (2) Change the parameter ω. ω is employed to control the proportion of accuracy and novelty parts of this problem. If we set $\omega = 1$, NSR is approximately an MF-based method while it is a NNs-based method if equals $\omega = 0$. (3) Rerank the recommendation lists. NSR can get every item's Interest and Satisfaction in I^{ur}. So we can rerank the results by different metrics, such as Interests First, or Satisfaction First. All the personalized recommendation methods are evaluated in next Section.

Above all, NSR is generalized as follows:

$$I^{rec} = NSR(R|\alpha, \beta, \gamma, \tau, \eta) \tag{21}$$

Where $\alpha, \beta, \gamma, \tau$ denote to the parameters of WMC, MLP, ECS, and embedding layer, and η denotes the hyper parameters of NSR. And the procedure of NSR is shown in Algorithm 3.

ALGORITHM 3: NSR procedure

Input: User-Item Matrix R
Output: Personalized Recommender List I_u^{rec}
  Initialize $\alpha, \beta, \gamma, \tau$,
  Embed R into X, Y
 Phrase 1: Serendipity Prediction
  Employ WMC to calculate $\hat{R}$ with Algorithm 1, update β, τ
  Employ MLP to calculate $\tilde{R}$ with Algorithm 2, update α, τ
 Phrase 2: Personalized Recommendation
  Employ ECS to decide I^{can} and bulid $\bar{R}$, R^c update γ
  Employ GMF to calculate I_u^{rec}
 Output I_u^{rec}

5 EXPERIMENTAL RESULTS

In this section, we conduct plenty of experiments on three different datasets. Because NSR is a combination of MF and NN, different baselines of MF and NN recommender methods are employed to make a comparison. Moreover, the effectiveness of parameters in NSR is also evaluated in detail.

5.1 Experimental Data

To validate NSR, we conducted experiments on three datasets: Movielens, a popular and standard dataset for evaluating RSs, and two real-world datasets: Yelp and Book-Crossing. Ratings range from 1 to 5. To verify the experiments, we use five-cross validation to divide the datasets, with 80% as training set and 20% as test set. All the statistics of these three datasets are shown in Table 3:

5.2 Baselines

We choose several state-of-the-art baselines: Item-Based RS (IBRS) [Sarwar et al. 2001], Interest-Oriented RS (IORS) [Hwang et al. 2016], BPR [Rendle et al. 2009], element-ALS [X. He et al. 2016c], NeuCF [He et al. 2017], and two novelty-based methods: TRECS [Ziegler et al. 2014] and Serendipity-ranking (SRE) [Kotkov et al. 2016a].

IBRS: Item-Based RS is a basic algorithm of a RS, which calculates the distance between items to get similarities. Then IBRS makes a recommendation through these similarities.

IORS: Interest-Oriented RS utilizes users' interests and employs MF to predict users' preference, making a personalized Interest-Oriented recommendation.

BPR: Bayesian personalized ranking is an improved algorithm which employs a pair-wise loss function, trying to maximize the distance between observed items and unobserved items.

e-ALS: element-ALS is a popular algorithm which can be applied to the RS with implicit feedbacks (clicks, browsing). Also, it achieves a stable performance in some RSs with explicit feedbacks.

NeuCF: NeuCF borrows the idea from Matrix Decomposition to build a NN. It generalizes a framework of NNs to cover basic collaborative filtering.

TRECS: TRECS is a hybrid algorithm to increase the novelty of RS, and it's a heuristic context-based method which utilizes users' prior experience, such as items' themes or categories.

SRE: Serendipity-ranking is a representation of re-ranking methods to improve novelty. It employs an item-based method to make a recommendation and sets some novelty-based metrics to rerank the recommender results.

5.3 Metrics

As our proposed method NSR is to recommend items with accuracy and novelty, we evaluate these perspectives to demonstrate the effectiveness. Furthermore, we explore the effects of parameters on both accuracy and serendipity and utilize F-measure to do some trade-offs between them.

5.3.1 Accuracy. We utilize two metrics to measure the accuracy of a RS for coverage and ranking utilities. ACC is used to measure the coverage of recommendation list, which is a combination of Precision $P = \frac{|I_u^{rec} \cap I_u^{true}|}{|I_u^{rec}|}$ and Recall $R = \frac{|I_u^{rec} \cap I_u^{true}|}{|I_u^{true}|}$, where I_u^{true} denotes the ground truth of users' rating item in I^{ur}, shown as follows:

$$ACC@k = \frac{2 \times (R@k) \times (P@k)}{(R@k) + (P@k)} \tag{22}$$

Where k denotes for the length of the recommendation list. Meanwhile, we utilize NDCG [Järvelin and Kekäläinen 2002] to evaluate the rank precision, shown as follows:

$$nDCG@k = \frac{DCG@k}{iDCG@k} \tag{23}$$

Where *DCG* is the rank score of the recommendation list, while *iDCG* is the ideal rank score.

5.3.2 Serendipity. We propose a novel Serendipity metric $S - p$. As we described before, we treat items with high Satisfaction but low Interest as Serendipity. If the Interest of an item is higher than Satisfaction, we think it harms the utility of Serendipity. So we calculate the pair-wise distance between Interest and Satisfaction of each item, which is shown as follows:

$$S - v@k = \frac{\sum_{i \in I^{rec}} (r_{ui}^{sa} - r_{ui}^{in})}{|I^{rec}|} \tag{24}$$

where r_{ui}^{in} and r_{ui}^{sa} is the predictions of Interests and Satisfactions and $|I^{rec}| = k$. Without losing generality, we also employ a proposed novelty metric [Kotkov et al. 2016b] to evaluate the

Table 4. Experiment Default Settings

Settings	Values
data partitions	80%T, 20%V
embedding-space k	20
threshold θ	$\theta^{in} = \theta^{sa} = 0.5$
NN structure	P = 4, tower-style 64-32-16-8
learning rate	0.01
recommend list	Top 5
evaluation metric	F with S-p and NDCG
linear parameter ω	0.5

serendipity.

$$S - p@k = \frac{\sum_{i \in I^{rec}} rel_u^i}{|I^{rec}|} \tag{25}$$

Note that rel_u^i is the novelty score consisting of relevance, unexpectedness, and novelty.

5.3.3 F-metric. Basically, Accuracy and Serendipity are confrontations. Therefore, we employ the F-metric to represent the trade-off between the accuracy A and serendipity S, shown as follows:

$$F@k = \frac{2 \times (A@k) \times (S@k)}{(A@k) + (S@k)} \tag{26}$$

Where A can be ACC or $nDCG$, and S can be $S - v$ and $S - p$.

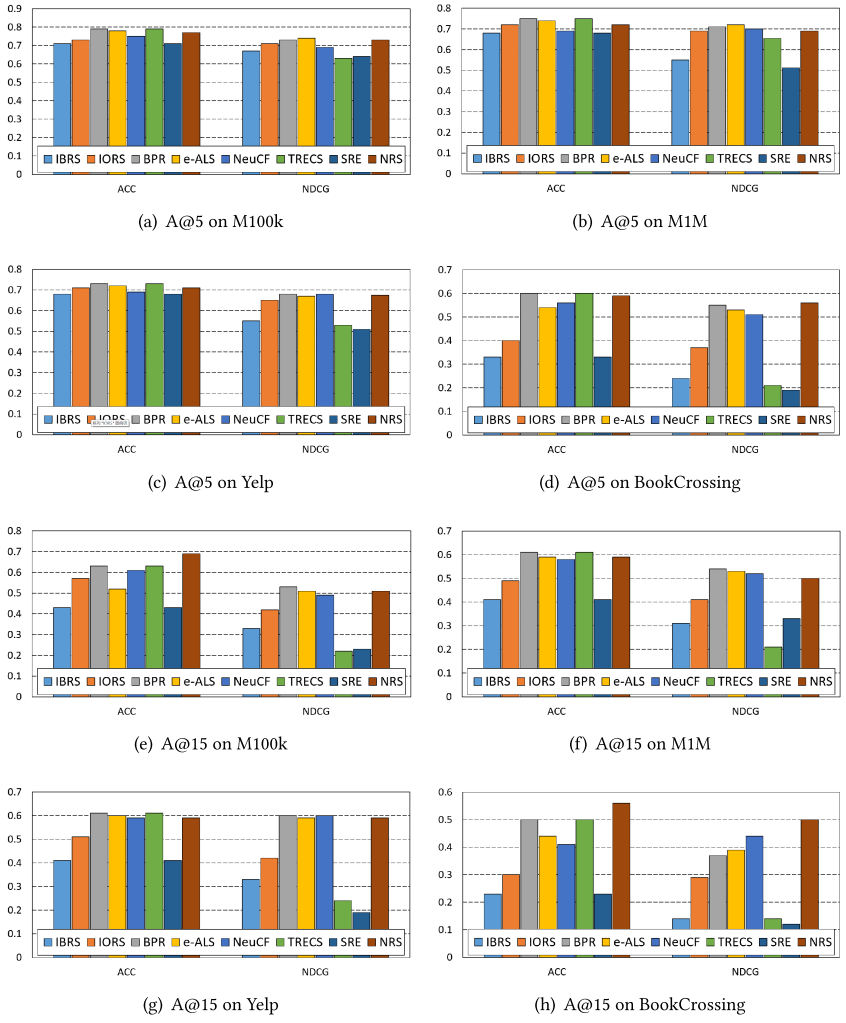
To make a fair comparison with the baselines, we select users with same rated-items number in unrated items as our recommendation targets. We use five-cross-validation, and divide the dataset into three different sizes of the training set (T) and the test set (V): 80%T, 20%V; 50%T, 50%V; 30%T, 70%V to validate the efficiency on different scales of datasets. Because our method is a hybrid method which employed MF and NN, so our baselines include different types of recommendation methods. If the baselines are MF-based, such as IBRS, IORS, BPR, and e-ALS, we can set a same latent space dimension k and same other hyper-parameters. If the baselines are NN-based like NeuCF, we can build a similar structure of networks (in our study, it is a tower structure 64, 32, 16, 8) and same optimizations (Adam) and hyper-parameters, like learning rate and attenuation rate. If we compare NSR with other reranking methods (TRECS, SRE), we set same parameters, reranking metrics, and thresholds. And we make our recommendation list as Top 5 and Top 15. Default settings are shown in Table 4.

5.4 Experiment Settings

5.5 Results and Discussion

5.5.1 Accuracy, Serendipity, and F. First, we evaluate our method with accuracy, serendipity, and F, with different categories of baselines. The results are shown in Figures 5–7.

Figure 5 shows the results of accuracy with different methods on different datasets. Generally, all the methods achieve better results in small and dense datasets than in large-scale and sparse datasets. As we introduced before, many existing methods focus on how to improve the accuracy of recommendation, and the performance of different methods on Top 5 recommendation in M100k is at the same level (shown in Figure 5(a)). However, when we add the number to Top 15 as shown in Figure 5(e), the performance of all methods is much worse than they did in Top 5. The similar situation occurs in other three datasets. Noting that TRECS and SRE are reranking methods to add



(a) A@5 on M100k (b) A@5 on M1M

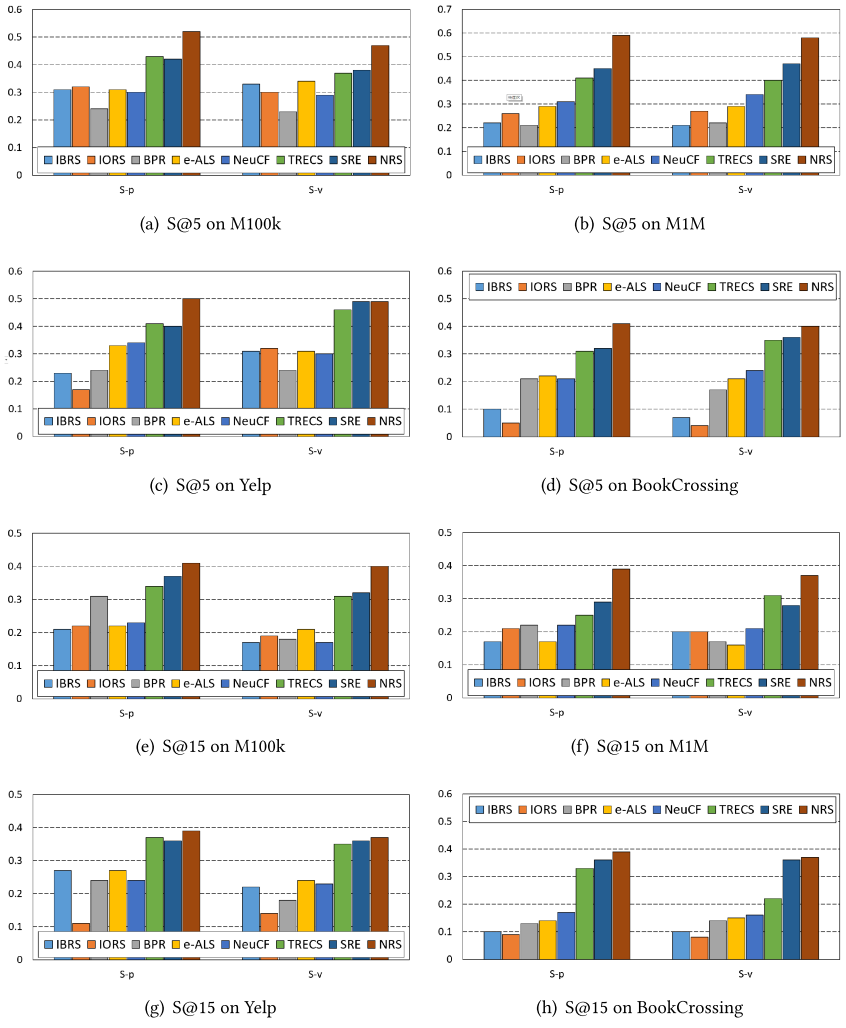
(c) A@5 on Yelp (d) A@5 on BookCrossing

(e) A@15 on M100k (f) A@15 on M1M

(g) A@15 on Yelp (h) A@15 on BookCrossing

Fig. 5. Performance of accuracy: ACC and NDCG.

the serendipity greatly, their NDCGs are quite low compared with other baselines. Basically, BPR, eALS, NeuCF, and our method NSR achieves better performance than other baselines. As dataset becomes more large-scale and sparse, obviously, our method performs better than all the other baselines. The reason is that NSR uses negative 0 injections instead of picking candidates, which builds an enhanced user-item matrix and alleviates the data sparsity situation. Figure 6 shows the serendipity of different baselines in different datasets. Cross-wisely, The performance in different datasets shows a similar tendency as they did before with accuracy. However, our method performs nearly 15% better than any other baselines (followed by TRECS and SRE, two reranking methods for novelty). In addition, we find that our proposed metric S-p contains consistency with S-v, which shows the efficiency and authority of our metric. Moreover, it is very obvious that the methods proposed for accuracy such as BPR, eALS, and IORS do not perform well for serendipity. These methods put too much attention to the accuracy part, ignoring the novelty part that users' ratings may affect their interests and actions, which is also the core issue of our method could address.



(a) S@5 on M100k (b) S@5 on M1M

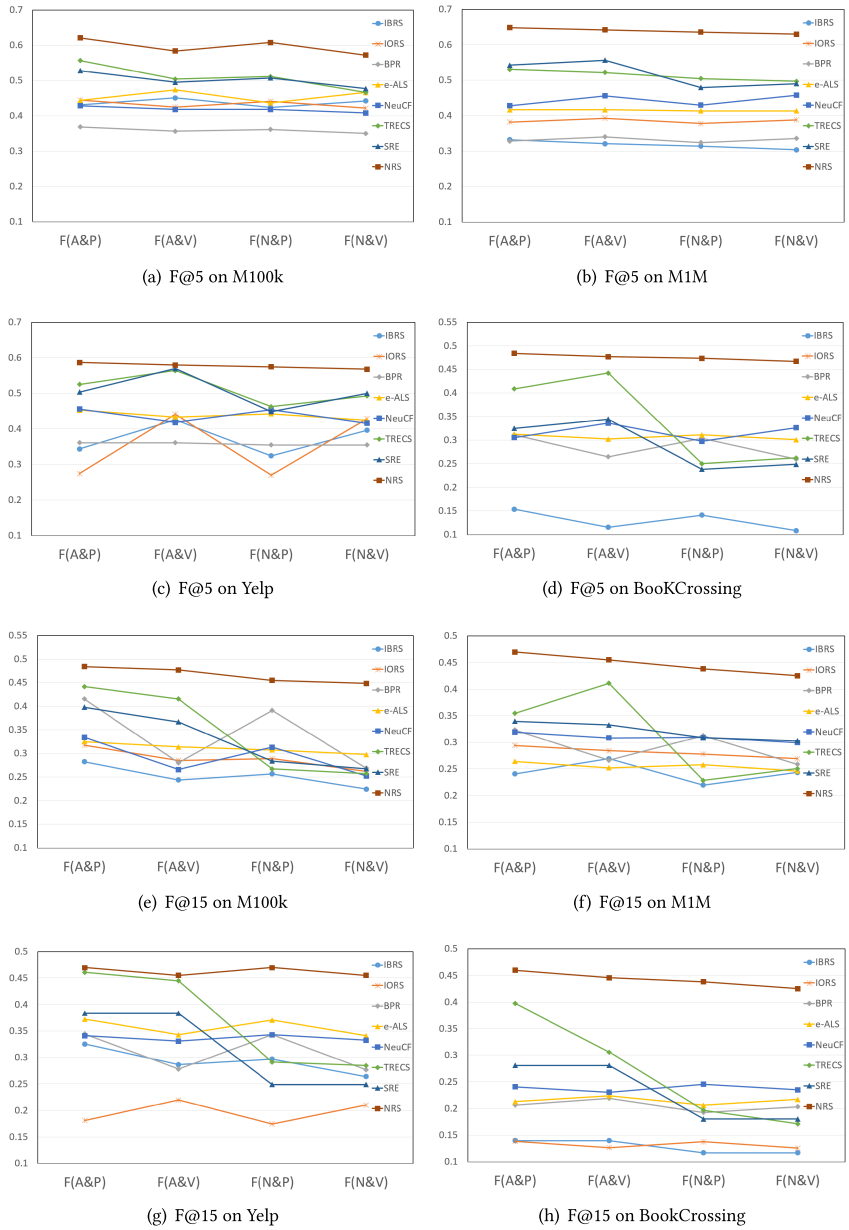
(c) S@5 on Yelp (d) S@5 on BookCrossing

(e) S@15 on M100k (f) S@15 on M1M

(g) S@15 on Yelp (h) S@15 on BookCrossing

Fig. 6. Performance of Serendipity: S-p and S-v.

And with datasets becoming much sparser and more large-scale, our method performs better than all the other baselines. Note that in Yelp, the basic method IBRS achieves a better serendipity than some advanced methods. The reason is that IBRS scarifies the accuracy and the results are random to some extent. Totally, the results show that our method is able to capture the serendipity efficiently. To make the trade-off between accuracy and serendipity obvious, we compare different F-measures cross serendipity, S-p (P), S-v (V), and accuracy ACC (A), NDCG (N), as shown in Figure 7. We notice that our method is better than any other baselines (almost 12%). The reason is simple: all existing baselines focus on either accuracy or novelty, which make a bad balance between them, and achieve an unstable performance. NSR can ensure the accuracy and achieve a better serendipity, so F measure of NSR is much better. Through different metrics of RSs and different scales of datasets, the results demonstrate that our proposed method NSR can facilitate a superior performance compared with baselines.



(a) F@5 on M100k (b) F@5 on M1M

(c) F@5 on Yelp (d) F@5 on BooKCrossing

(e) F@15 on M100k (f) F@15 on M1M

(g) F@15 on Yelp (h) F@15 on BookCrossing

Fig. 7. Performance of different F-measure: trade-off between Accuracy and Serendipity.

5.5.2 Deep Evaluation of NSR. In this section, we evaluate NSR in detail. Different data-partitions, parameters, and settings are applied respectively to test the efficiency and robust of NSR.

First, as we know, the data in real-world is not only sparse but also large-scale. A well-designed RS should have the ability to achieve a stable performance with poor training data quality and large quantity. So we evaluate NSR with different data-partitions, 80%T, 20%V; 50%T, 50%V; 30%T,

Table 5. Performance of F-measure with Different Data-partitions with Top@5

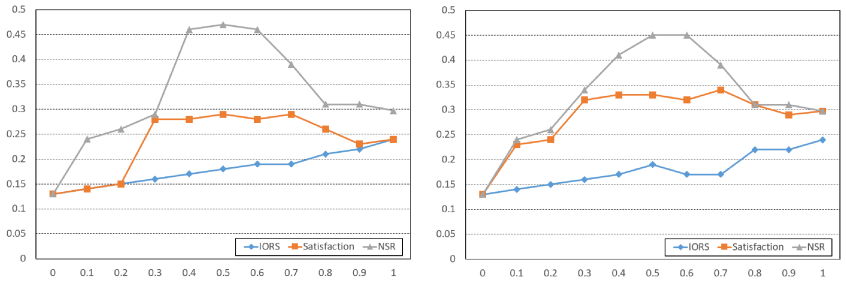
Dataset	Partition	Top@5			NSR
		IBRS	BPR	NeuCF	
M100K	80%T, 20%V	0.42±0.021	0.36±0.042	0.42±0.041	**0.60±0.021**
	50%T, 50%V	0.20±0.053	0.27±0.084	0.26±0.112	**0.42±0.043**
	30%T, 70%V	0.10±0.033	0.12±0.027	0.13±0.023	**0.27±0.016**
M1M	80%T, 20%V	0.31±0.013	0.32±0.024	0.43±0.010	**0.64±0.010**
	50%T, 50%V	0.27±0.022	0.26±0.021	0.34±0.016	**0.41±0.011**
	30%T, 70%V	0.20±0.122	0.22±0.082	0.18±0.092	**0.27±0.062**
Yelp	80%T, 20%V	0.32±0.012	0.35±0.032	0.45±0.102	**0.57±0.047**
	50%T, 50%V	0.24±0.102	0.33±0.072	0.31±0.053	**0.40±0.041**
	30%T, 70%V	0.23±0.100	0.18±0.066	0.18±0.053	**0.27±0.084**
Book Crossing	80%T, 20%V	0.14±0.012	0.30±0.008	0.29±0.012	**0.47±0.020**
	50%T, 50%V	0.12±0.002	0.11±0.012	0.15±0.009	**0.24±0.012**
	30%T, 70%V	0.13±0.010	0.13±0.011	0.13±0.014	**0.17±0.020**

Table 6. Performance of F-measure with Different Data-partitions with Top@15

Dataset	Partition	Top@5			NSR
		IBRS	BPR	NeuCF	
M100K	80%T, 20%V	0.25±0.033	0.39±0.023	0.31±0.013	**0.45±0.022**
	50%T, 50%V	0.23±0.063	0.33±0.092	0.32±0.085	**0.37±0.073**
	30%T, 70%V	0.14±0.033	0.15±0.043	0.16±0.52	**0.19±0.083**
M1M	80%T, 20%V	0.22±0.023	0.31±0.031	0.31±0.022	**0.43±0.019**
	50%T, 50%V	0.26±0.012	0.27±0.032	0.31±0.029	**0.34±0.012**
	30%T, 70%V	0.22±0.112	0.23±0.082	0.21±0.061	**0.23±0.042**
Yelp	80%T, 20%V	0.29±0.055	0.34±0.012	0.34±0.042	**0.46±0.036**
	50%T, 50%V	0.22±0.100	0.24±0.022	0.26±0.072	**0.29±0.052**
	30%T, 70%V	0.22±0.073	0.21±0.062	0.21±0.063	**0.29±0.082**
Book Crossing	80%T, 20%V	0.11±0.003	0.19±0.007	0.24±0.023	**0.43±0.009**
	50%T, 50%V	0.17±0.003	0.14±0.011	0.15±0.007	**0.19±0.010**
	30%T, 70%V	0.13±0.012	0.14±0.022	0.12±0.021	**0.17±0.009**

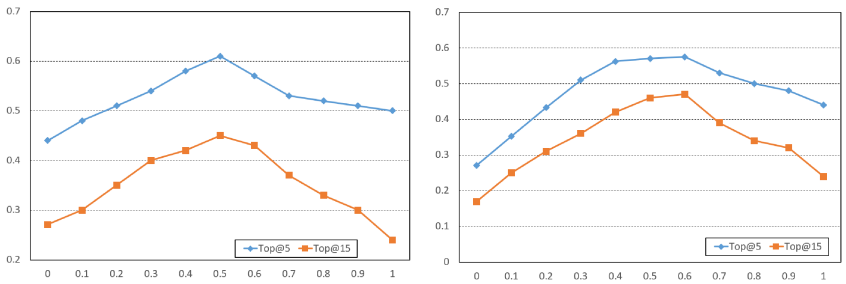
70%V on Yelp dataset for Top 5 recommendation. We choose NDCG & S-p as our metric and pick some typical baselines (IBRS, BPR, NeuCF). The results are shown in Table 5

As shown in Tables 5 and 6, all these methods perform best with the most training data (80%T, 20%V) and worst with the least training data (30%T, 70%V). With a plat comparison across Top@5 and Top@15, as applying NDCG and S-p to build F-measure in this experiment, we find that when the data becomes sparse and large-scale, the accuracy of RS decreases greatly (especially for the baselines) but the serendipity is hold at a stable level (obvious in Table 6). However, for our proposed method NSR, we consider accuracy and novelty jointly, and use an enhanced matrix as our data input, which alleviates the data sparse problem to some extent. NSR can make a balance between accuracy and novelty, which can ensure the accuracy with a relatively high satisfaction. More data we mine from the users interaction, a much better performance are achieved [Bobadilla et al. 2013]. This character keeps our method nearly 10% better than baselines on different data-partitions.



(a) F@5 on Yelp (b) F@15 on Yelp

Fig. 8. Effect of Interests and Satisfactions threshold θ on F-measure.



(a) F on M100k (b) F on Yelp

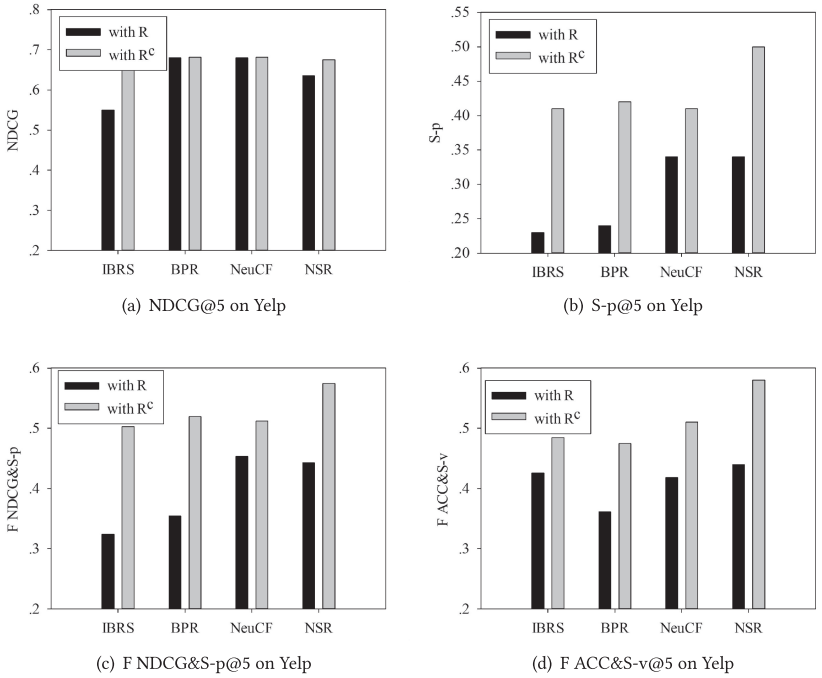
Fig. 9. Effect of trade-off parameter ω on F-measure.

Second, we evaluate our method with different parameters. The core parameters in NSR are the thresholds θ^{in}, θ^{sa}. And we choose two baselines which use either θ^{in} (IORS) or θ^{sa}(Satisfaction). To make it general, we set $\theta^{in} = \theta^{sa}$ in NSR, and make a comparison with F-measure with NDCG and S-p in Yelp for Top 5 and Top 15 recommendation. The threshold changes from 0.0 to 1.0 with 0.1 each step. The results are shown in Figure 8.

In Figure 8(a), we see that at the near end and the far end of the threshold value, the performances of the three methods are at the same level. However, when it comes to 0.4–0.8, our method performs much better than the baselines. The reason is that our method uses two thresholds, and build an enhanced matrix with negative injections, which greatly increases the robustness and keeps the superior performance across the baselines. Note that IORS performs the worst because it totally ignores the serendipity in a RS. And when we set the threshold too large, RS will lose the ability to screen the candidate set, so the results will be closer to the basic non-heuristic recommender methods. In Figure 8(b), the performance becomes more sensitive at first. And only the baseline utilizing satisfaction keeps the pace with NSR. The reason is that when we recommend more items, serendipity will increase with the loss of accuracy, that's why IORS performs the worst.

As we introduced before, our method is a two-phase method which measures accuracy part and novelty part of the recommendation. Then we decide to evaluate the importance of both parts by changing the parameter ω. And we set ω from 0 (completely novelty method) to 1 (completely accuracy method) with each step of 0.1. The results are shown in Figure 9.

Noting that when we set $\omega = 0$, our method becomes a linear-model which is very similar to IORS which focuses on accuracy. And the performance is unbearable. With the trade-off between linear and non-linear increases, NSR gets its best performance. However, when ω becomes too large, novelty part overcome the accuracy part then it hurts the performance of NSR badly. The similar



(a) NDCG@5 on Yelp (b) S-p@5 on Yelp

(c) F NDCG&S-p@5 on Yelp (d) F ACC&S-v@5 on Yelp

Fig. 10. Effect of enhanced matrix R^c for different methods.

situation happens in both small dataset (Figure 9(a) for M100k) and large dataset (Figure 9(b)). The results prove that for serendipity, the combination our proposed method NSR employed can get better performance than using MF or MLP, respectively.

Finally, we evaluate our methods for different combinations with different recommendation methods. As our method is building an enhanced user-item matrix R^c, it can be applied in various different recommender methods. So we choose some typical baselines and make a comparison with our proposed method NSR. We apply our method in the first phase to build R^c with fixed $\omega = 0.5$, then feed R^c to the phase 2 of NSR, and different baselines instead of original matrix R. The results are shown in Figure 10.

As shown in Figure 10(a), with original input R, our proposed method performs no better than baselines in NDCG. And the enhanced matrix R^c improves the accuracy performance of baselines only to a small extent. However, when we check the serendipity, we found that R^c made a significant improvement in $S - p$, maybe several times better than using original matrix R. Also, we make some comparisons in F-measure. The results show that our method can be applied in different recommender methods, which proves the general applicability. However, the chosen recommender method GMF can get the best performance based on R^c among baselines.

6 CONCLUSION

Many researchers have focused on improving the users' experience through RS by extracting users' preferences. But they always failed because of ignoring the conflict of accuracy and novelty. In this study, we provide insights into a recommendation dilemma in real-world scenarios: the trade-off between accuracy and novelty. To resolve this dilemma, we define a novel concept: Serendipity, which is characterized by low-interest but high-satisfaction, to model the users' preferences. Along

this line, we propose a novel NSR method, combining MF and NN. Specifically, MF is to ensure the accuracy while NN is to explore the novelty. Furthermore, based on predicted serendipity and a candidate filtering method, NSR can make personalized recommendations. Finally, extensive experiments are conducted to demonstrate that our proposed method can make an excellent trade-off performance compared with the state-of-art baselines, especially with sparse and large-scale datasets.

However, noting that we only utilize user-item ratings as the input of our proposed method, NSR has the potential to be more effective and applicable. In the future, the work could be explored for how to utilize the multi-source, multi-view data, such as pictures, photos, texts and audio to make the serendipity recommendation for users. In addition, the advanced NN structures, such as CNN and GAN [Goodfellow et al. 2014] ought to be applied in order to improve the efficiency and accuracy.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. 2015. Context-aware recommender systems. In *Recommender Systems Handbook*. Springer, 191–226.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based Systems* 46 (2013), 109–132.

Jiajun Bu, Xin Shen, Bin Xu, Chun Chen, Xiaofei He, and Deng Cai. 2016. Improving collaborative recommendation via user-item subgroups. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2363–2375.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*. http://icml.cc/2012/papers/416.pdf.

Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to recommend accurate and diverse items. In *Proceedings of the 26th International Conference on World Wide Web*. 183–192.

Li Deng and Dong Yu. 2014. Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing* 7, 3–4 (2014), 197–387.

Anupriya Gogna and Angshul Majumdar. 2017. Balancing accuracy and diversity in recommendations using matrix completion framework. *Knowledge-Based Systems* 125 (2017), 83–95.

Carlos A. Gomez-Uribe and Neil Hunt. 2016. The Netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems* 6, 4 (2016), 13.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems*. 2672–2680.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

Chen He, Denis Parra, and Katrien Verbert. 2016. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications* 56 (2016), 9–27.

Xiangnan He, Hanwang Zhang, Min Yen Kan, and Tat Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 549–558.

Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 355–364.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

Won Seok Hwang, Juan Parc, Sang Wook Kim, Jongwuk Lee, and Dongwon Lee. 2016. "Told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *Proceedings of theIEEE International Conference on Data Engineering*. 349–360.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

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 (2016), 2.

Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2016a. Challenges of serendipity in recommender systems. In *Proceedings of the 12th International Conference on web Information Systems and Technologies. Volume 2, ISBN 978-989-758-186-1*. SCITEPRESS.

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016b. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016c. Improving serendipity and accuracy in cross-domain recommender systems. In *Proceedings of the International Conference on Web Information Systems and Technologies*. Springer, 105–119.

Dokyun Lee and Kartik Hosanagar. 2016. When do recommender systems work the best?: The moderating effects of product attributes and consumer reviews on recommender performance. In *Proceedings of the 25th International Conference on World Wide Web*. 85–97.

Zhaoyi Li, Fei Xiong, Ximeng Wang, Hongshu Chen, and Xi Xiong. 2019. Topological influence-aware recommendation on social networks. *Complexity* 2019 (2019) 12.

Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1015–1024.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 452–461.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the International Conference on World Wide Web*. 285–295.

Donald F. Specht. 1990. Probabilistic neural networks. *Neural Networks* 3, 1 (1990), 109–118.

Jing Sun, Yun Xiong, Yangyong Zhu, Junming Liu, Chu Guan, and Hui Xiong. 2015. Multi-source information fusion for personalized restaurant recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 983–986.

Hao Wang, Yanmei Fu, Qinyong Wang, Hongzhi Yin, Changying Du, and Hui Xiong. 2017. A location-sentiment-aware recommender system for both home-town and out-of-town users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1143.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *Arxiv Preprint Arxiv:1901.00596* (2019).

Lin Xiao, Zhang Min, Zhang Yongfeng, Liu Yiqun, and Ma Shaoping. 2017. Learning and transferring social and item visibilities for personalized recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 337–346.

Fei Xiong, Ximeng Wang, Shirui Pan, Hong Yang, Haishuai Wang, and Chengqi Zhang. 2018. Social recommendation with evolutionary opinion dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* PP, 99 (2018), 1–13.

Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. 2017. R a P are: A generic strategy for cold-start rating prediction problem. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1296–1309.

Yuanbo Xu, Yongjian Yang, Jiayu Han, En Wang, Fuzhen Zhuang, Jingyuan Yang, and Hui Xiong. 2019. NeuO: Exploiting the sentimental bias between ratings and reviews with neural networks. *Neural Networks* 111 (2019), 77–88. DOI: https://doi.org/10.1016/j.neunet.2018.12.011

Surong Yan, Kwei-Jay Lin, Xiaolin Zheng, Wenyu Zhang, and Xiaoqing Feng. 2017. An approach for building efficient and accurate social recommender systems using individual relationship networks. *IEEE Transactions on Knowledge and Data Engineering* 29, 10 (2017), 2086–2099.

Jingyuan Yang, Chuanren Liu, Mingfei Teng, Hui Xiong, March Liao, and Vivian Zhu. 2015. Exploiting temporal and social factors for B2B marketing campaign recommendations. In *Proceedings of the IEEE International Conference on Data Mining*. 499–508.

Yongjian Yang, Yuanbo Xu, En Wang, Jiayu Han, and Zhiwen Yu. 2018. Improving existing collaborative filtering recommendations via serendipity-based algorithm. *IEEE Transactions on Multimedia* 20, 7 (2018), 1888–1900. DOI: https://doi.org/10.1109/TMM.2017.2779043

Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 3926–3932. DOI: https://doi.org/10.24963/ijcai.2018/546

Zhiwen Yu, Miao Tian, Zhu Wang, Bin Guo, and Tao Mei. 2016a. Shop-type recommendation leveraging the data from social media and location-based services. *ACM Transactions on Knowledge Discovery from Data* 11, 1 (2016), 1.

Zhiwen Yu, Huang Xu, Zhe Yang, and Bin Guo. 2016b. Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints. *IEEE Transactions on Human-Machine Systems* 46, 1 (2016), 151–158.

Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 325–334.

Sheng Zhang, Weihong Wang, James Ford, Fillia Makedon, and Justin Pearlman. 2005. Using singular value decomposition approximation for collaborative filtering. In *Proceedings of the 7th IEEE International Conference on E-Commerce Technology*. IEEE, 257–264.

Xiaosong Zhou, Zhan Xu, Xu Sun, and Qingfeng Wang. 2017. A new information theory-based serendipitous algorithm design. In *Proceedings of the International Conference on Human Interface and the Management of Information*. Springer, 314–327.

Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen. 2014. Mobile app recommendations with security and privacy awareness. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 951–960.

Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. 2017. Representation learning via Dual-Autoencoder for recommendation. *Neural Networks* 90 (2017), 83–89. DOI: https://doi.org/10.1016/j.neunet.2017.03.009

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*. ACM, 22–32.

Cai-Nicolas Ziegler, Thomas Hornung, Martin Przyjaciel-Zablocki, Sven Gauß, and Georg Lausen. 2014. Music recommenders based on hybrid techniques and serendipity. *Web Intelligence and Agent Systems: An International Journal* 12, 3 (2014), 235–248.

Received September 2018; revised October 2019; accepted April 2020