This is the final draft of the following article:

Gaurav Pandey, Denis Kotkov, and Alexander Semenov. 2018. Recommending Serendipitous Items using Transfer Learning. In The 27th ACM International Conference on Information and Knowledge Management (CIKM'18), October 22–26, 2018, Torino, Italy. ACM, New York, NY, USA, 4 pages. doi: 10.1145/3269206.3269268

Recommending Serendipitous Items using Transfer Learning

Gaurav Pandey
University of Jyvaskyla
gaurav.g.pandey@jyu.fi

Denis Kotkov
University of Jyvaskyla
kotkov.denis.ig@gmail.com

Alexander Semenov
University of Jyvaskyla
alexander.v.semenov@jyu.fi

ABSTRACT

Most recommender algorithms are designed to suggest relevant items, but suggesting these items does not always result in user satisfaction. Therefore, the efforts in recommender systems recently shifted towards serendipity, but generating serendipitous recommendations is difficult due to the lack of training data. To the best of our knowledge, there are many large datasets containing relevance scores (relevance oriented) and only one publicly available dataset containing a relatively small number of serendipity scores (serendipity oriented). This limits the learning capabilities of serendipity oriented algorithms. Therefore, in the absence of any known deep learning algorithms for recommending serendipitous items and the lack of large serendipity oriented datasets, we introduce *SerRec* our novel transfer learning method to recommend serendipitous items. *SerRec* uses transfer learning to firstly train a deep neural network for relevance scores using a large dataset and then tunes it for serendipity scores using a smaller dataset. Our method shows benefits of transfer learning for recommending serendipitous items as well as performance gains over the state-of-the-art serendipity oriented algorithms.

KEYWORDS

Recommender System; Serendipity; Deep Learning; Transfer Learning

ACM Reference Format:
Gaurav Pandey, Denis Kotkov, and Alexander Semenov. 2018. Recommending Serendipitous Items using Transfer Learning. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3269206.3269268

1 INTRODUCTION

Relevance oriented recommender algorithms often suggest items that users are either already familiar with or would easily find themselves leading to low satisfaction [8]. To overcome this problem, recommender algorithms should suggest serendipitous (i.e. relevant, novel and unexpected) items, as these items are more likely to broaden user preferences than relevant non-serendipitous ones [7]. While there has been a lot of work in the area of relevance oriented recommendations including deep learning [1–3, 10], the efforts for serendipitous recommendations are still very limited. To the best of our knowledge, deep learning or transfer learning methods have not yet been explored for recommending serendipitous items.

Although there is abundant availability of big training datasets with relevance scores, that can be used for relevance oriented algorithms, there are not many training datasets available with serendipity scores. To the best of our knowledge, Serepdipity-2018[1] is the only such publicly available dataset, having comparatively a rather small set of serendipity scores. One of the reasons for the lack of large serendipity related datasets is the high level of difficulty to collect user feedback regarding serendipity. This requires a lot of effort from users as they need to answer many questions, and not just provide scores for the items [7]. The unavailability of large datasets with serendipity scores poses limitations on the training of serendipity oriented recommender models.

Therefore, in the absence of large datasets and deep learning algorithms focused on serendipity, we introduce *SerRec*: a transfer learning method that trains a deep neural network for relevance scores using a large dataset and then tunes it for serendipity scores using a smaller dataset. This allows us to use the available training data with relevance scores to benefit in the process of recommending serendipitous items.

Our method utilizes the neural collaborative filtering (NCF) framework proposed by He et al [3]. They introduced an ensemble of deep neural networks, originally proposed to learn a relevance oriented recommender model. To benefit from transfer learning, we firstly train the deep neural network ensemble layers using an available large training dataset with relevance scores. Thereafter, we tune the last layer of the network using a small dataset with serendipity scores.

For our experiments, we have used Serendipity-2018, the only publicly available dataset that has user feedback on serendipitous items [7], that to the best of our knowledge has not been used so far in studies. This dataset consists of a large collection of relevance scores along with a smaller collection of serendipity scores. Experimenting with Serendipity-2018 is also novel in itself, because serendipity oriented algorithms till now have been evaluated on datasets where serendipity was measured using artificial serendipity metrics based on assumptions regarding serendipity that might not correspond to reality [7]. Our experimental results show the benefits of transfer learning to train a serendipity oriented recommender model and shows improvements over the state-of-the-art serendipity oriented recommender models.

To summarize, this paper has the following key contributions:

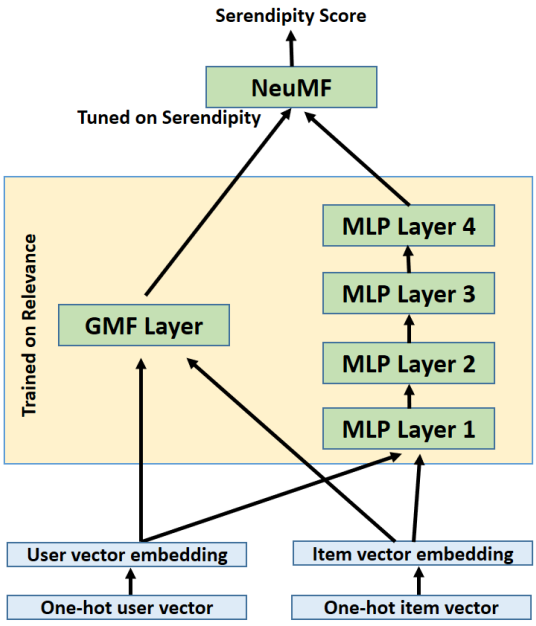
- We propose the novel deep transfer learning method *SerRec* for serendipitous recommendations.
- Our method utilizes the neural collaborative filtering framework to utilize a large relevance score dataset along with a

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269268

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

Figure 1: *SerRec* Architecture for Transfer Learning for Serendipity using NCF framework [3]



smaller serendipity score dataset to enable high performance serendipitous items recommendations.

- We evaluate *SerRec* and compare it with the-state-of-the-art serendipity oriented algorithms on the first publicly available serendipity oriented dataset Serendipity-2018.

2 OBJECTIVES

In this study, we aim to address mainly the following questions:

- How can we do transfer learning for serendipitous recommendations using the relevance score training data?
- Does transfer learning help the serendipity oriented recommender model?
- How does our model compare to the state-of-the-art serendipity oriented models?

3 *SERREC* FRAMEWORK

In this section, we describe the *SerRec* methodology, its setup, utilized datasets, baselines and the metric employed for comparisons.

3.1 Methodology

Consider a set of users as $U = \{u_0, u_1, ..., u_N\}$, and a set of items as $I = \{i_0, i_1, ..., i_M\}$. We denote user-item interaction matrix as $R = \{r_{jk}\}$, where $r_{jk} = 1$ if user u_j rated item i_k, $j \in \{0, ..., N\}, k \in \{0, ..., M\}, r_{jk} \in \{0, 1\}$. We denote user-item serendipity matrix as $S = \{s_{jk}\}$, where $s_{jk} = 1$ if user u_j considers item i_k, where $j \in \{0, ..., N\}, k \in \{0, ..., M\}, s_{jk} \in \{0, 1\}$ as serendipitous.

Our goal is to learn a function $\hat{s}_{jk} = f(j, k \mid \Theta)$, where $\hat{s}_{jk}$ is predicted serendipity score, and Θ is the vector of model parameters. Here, j and k are the indexes of user and item matrices U and I, respectively. In order to find optimal parameters Θ for function $f(j, k \mid \Theta)$ we need to minimize loss function $\mathcal{L}(S, f(j, k \mid \Theta))$ between actual and predicted serendipity scores.

As per He et al. [3], we represent function $f(j, k \mid \Theta))$ as neural network, depicted in Figure 1. Input data for the neural network is one-hot encoded user and item vectors: $\boldsymbol{u}$ and $\boldsymbol{i}$, $|\boldsymbol{u}| = N + 1, |\boldsymbol{i}| = M + 1$. At first, we create vector embeddings $\boldsymbol{p}$ and $\boldsymbol{q}$ from vectors $\boldsymbol{u}$ and $\boldsymbol{i}$ respectively. These embeddings are created by learning weights of two matrices: W_U and W_I; multiplication of $\boldsymbol{u}$ or $\boldsymbol{i}$ to W_U or W_I respectively would return $\boldsymbol{p}$ and $\boldsymbol{q}$ [3]. The sizes of these embeddings $\boldsymbol{p}$ and $\boldsymbol{q}$ are n and m respectively, where $n << N + 1$ and $m << M + 1$.

Left-hand side of the neural network at Figure 1 is the layer performing Generalized Matrix Factorization (GMF), i.e.:

$$O_{GMF} = (\boldsymbol{p} \odot \boldsymbol{q}) \tag{1}$$

Moreover on the right hand side, Multi Layer Perceptron (MLP) layers 1, 2, 3 and 4 contain dense layers:

$$O_{MLP_k} = RelU(W_k \cdot O_{MLP_{k-1}} + B_k), k \in \{1, 2, 3, 4\}, \tag{2}$$

where $O_{MLP_0} = [\boldsymbol{p}, \boldsymbol{q}]$, i.e. vector built by concatenation of user and item embeddings. Final layer of the network Neural Matrix Factorization (NeuMF in Figure 1) implements the sigmoid activation function:

$$O_{NeuMF} = \sigma(W_O \cdot [O_{GMF}, O_{MLP_4}] + B_O), \tag{3}$$

where W_O and B_O are weights and biases of the output layer.

We adopt cross entropy as the loss function:

$$\mathcal{L}(S, \hat{S}) = -\sum_{(j,k) \in \widetilde{S}} (s_{jk} \log \hat{s}_{jk} - (1 - s_{jk}) \log(1 - \hat{s}_{jk})), \tag{4}$$

where $\widetilde{S}$ denotes observed part of serendipity matrix S, and $\hat{S} = f(j, k \mid \Theta)$ is the predicted serendipity.

Since serendipity scores dataset is small, we try to utilize transfer learning by training a deep neural network using relevance scores. First, we train the entire neural network framework on relevance data R, and optimize the loss $\mathcal{L}(R, \hat{R})$. After training the entire neural network on relevance, we fix all weights except final one (W_O, and B_O), and train it on serendipity matrix S.

3.2 *SerRec* Setup

For our experiments we have used the Neural Collaborative Filtering [3] implementation[2] as discussed in section 3.1. As shown in Figure 1, we used the training dataset with relevance scores to train the GMF the MLP layers, where we used four layers in MLP. Thereafter, the NeuMF layer is tuned using the serendipity tuning dataset and final network is tested on the serendipity test dataset. These datasets are explained in detail in Section 3.3.

For initial training on GMF and MLP, we used the user and item vectors embeddings $\mathbf{p}$ and $\mathbf{q}$, created using relevance training dataset $\widetilde{R} \subset R$. We used a learning rate of 0.001 for GMF and 0.01 for MLP, and trained both of them for 20 epochs while keeping a batch size of 254. Also for both GMF and MLP, we used Adam optimization algorithm [5]. Moreover, the available implementations for MLP and GMF convert the available ratings into implicit feedback (rated

[2]https://github.com/hexiangnan/neural_collaborative_filtering

or unrated) and for this chooses randomly four negative samples (unrated) for each positive (rated) item for a user. We have used these default settings.

Then using these trained GMF and MLP layers and keeping the weights fixed in them, we tuned the NeuMF layer using the serendipity tuning dataset. For this we used $\widetilde{S} \subset S$, that contains negative and positive samples on serendipity. We tuned it till convergence using a learning rate of 0.001. We used this trained and tuned network to predict the serendipity scores of the test dataset.

3.3 Datasets

To evaluate our algorithm and baselines, we employed Serendipity-2018 dataset [7]. To the best of our knowledge, this is the only publicly available dataset, which contains user feedback regarding serendipity. This dataset contains 5-star scores (relevance scores) users gave to movies in MovieLens[3] and binary scores (serendipity scores) indicating whether particular movies are serendipitous to particular users. The dataset contains ten million relevance scores and 2,150 serendipity scores.

The dataset contains different kinds of serendipity. In this study, we target six kinds of serendipity that are missing the unexpectedness variation, which hurts user satisfaction: strict serendipity (find), strict serendipity (implicit), strict serendipity (recommend), motivational serendipity (find), motivational serendipity (implicit) and motivational serendipity (recommend) [7]. We pre-process this dataset and regard a movie serendipitous (positive sample) if it is serendipitous according to at least one of these variations of serendipity, and otherwise regard it as non-serendipitous (negative example). The dataset contains 277 serendipitous user movie pairs out of total 2,150.

For the Serendipity-2018 dataset, serendipity scores were obtained in a survey taken by 481 users. In the survey, the authors selected movies that were likely to be serendipitous to users, such as unpopular movies that were given high scores [7]. To extend serendipity scores for our study, we randomly selected five relevance scores per user and assigned negative (non-serendipitous) serendipity scores to them. We regarded these movies non- serendipitous, as they were unlikely to be serendipitous to users. In the best case scenario, the chance of a movie to be serendipitous is 13% ($\frac{277}{2150} = 0.129$). In our case, the chance of a movie to be serendipitous is much lower, since we did not control for popularity or score. After attaching serendipity scores to randomly selected relevance scores, the number of serendipity scores exceeded 4,555 with 277 scores indicating serendipitous movies and 4,278 indicating non-serendipitous ones.

We split the dataset into three datasets: training, tuning and test. Tuning and test datasets contain both relevance and serendipity scores, while the training dataset only contains relevance scores. The tuning datasets contains 75% of serendipity scores, while the test dataset contains the remaining 25% of serendipity scores.

3.4 Baselines

We implemented the following baselines for comparison with our transfer learning method *SerRec*:

- **POP**: We implemented popularity baseline that arranges items according to the number of relevance scores received by them in the training dataset, in the descending order.
- **UNPOP**: Unpopularity or inverse popularity baseline orders items according to the number of relevance scores in the ascending order.
- **Random**: This baseline orders the items randomly.
- **SVD**: Singular value decomposition [6] orders items according to the predicted scores. SVD decomposes the user-item matrix into two matrices using gradient decent. The gradient decent algorithm minimizes the objective function, which is the error between actual and predicted scores. Based on tuning, we picked the parameters: feature number=200, learning rate=10^{-5} and regularization term=0.1.
- **SPR**: Serendipitous Personalized Ranking is a serendipity-oriented variation of SVD with the modified objective function [9]. SPR is a learning to rank algorithm, which maximizes the difference between scores of relevant and irrelevant items for each user and weights this distance based on popularity of the irrelevant item. Based on tuning, we picked the parameters: Bayesian loss function, $\alpha = 0.4$, feature number=200, learning rate=10^{-5} and regularization term=0.1.
- **UAUM**: Unexpectedness-Augmented Utility Model is also a serendipity-oriented variation of SVD [12]. UAUM minimizes the objective function, which is the error weighted with the unexpectedness term. In our implementation, we excluded unobserved scores due to the size of our dataset. Based on tuning, we picked the parameters: feature number=200, learning rate=10^{-5} and regularization term=0.1.
- ***SerRec***$_{NoTL}$: To see if transfer learning indeed helps in recommending serendipitous items, we trained the non transfer learning version of our method, where we trained all the layers in Figure 1 using only serendipity scores from tuning dataset.

We used the following procedure to evaluate our baseline algorithms: (1) we trained the baselines on relevance scores of the training dataset, (2) we tuned the parameters of the baselines on serendipity scores of the tuning dataset, (3) we trained the baselines with the tuned parameters on relevance scores of training and tuning datasets combined and (4) we evaluated the trained baselines on serendipity scores of the test dataset. We did not train the baselines directly on the serendipity scores, as Serendipity-2018 does not contain enough serendipity scores for training the algorithms.

3.5 Metric

To compare the serendipitous item recommendation performance of *SerRec* against the baselines on the serendipity test set, we employed the standard retrieval metric NDCG@1-10 (normalized discounted cumulative gain) [4]. While NDCG computation typically utilizes the relevance scores of items in a ranking, we used the available serendipity scores.

4 RESULTS

Table 1 compares the serendipitous recommendation performance of *SerRec* against the baselines using NDCG@1-10. The following observations can be made from the results:

[3]https://movielens.org/

Table 1: Serendipity Ranking Performance Comparison of *SerRec*

NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
POP	0.0303	0.0606	0.0606	0.0909	0.1146	0.1948	0.2518	0.2994	0.3550	0.4078
Random	0.0909	0.1969	0.3262	0.3895	0.4248	0.4529	0.5036	0.5339	0.5639	0.5674
SVD	0.2121	0.3484	0.4035	0.4499	0.4981	0.5298	0.5643	0.6086	0.6218	0.6281
UAUM	0.3333	0.4696	0.4765	0.5117	0.5511	0.5900	0.5900	0.6336	0.6623	0.6749
UNPOP	0.3636	0.4393	0.5017	0.5651	0.6478	0.6633	0.6849	0.6849	0.7040	0.7040
SPR	0.3636	0.5000	0.5678	0.6230	0.6731	0.6944	0.7268	0.7369	0.7369	0.7369
$SerRec_{NoTL}$	0.2727	0.4091	0.5420	0.6490	**0.6960**	0.7135	0.7135	0.7236	0.7236	0.7236
SerRec	**0.4848**	**0.5455**	**0.6269**	**0.6505**	0.6907	**0.7186**	**0.7363**	**0.7452**	**0.7614**	**0.7614**

(1) We see that *SerRec* outperforms all the baselines algorithms at all the NDCG metrics, only except at NDCG@5 where $SerRec_{NoTL}$ is the best performing algorithm.
(2) For some metrics $SerRec_{NoTL}$ is the second best algorithm and SPR the third best, while for other metrics it is the other way round. They are followed by UNPOP, UAUM, SVD, Random and POP in this particular order.
(3) We also see the benefits of transfer learning since *SerRec* outperforms $SerRec_{NoTL}$ for most of the metrics.
(4) We observe that popularity of the items is working against the serendipity because UNPOP shows decent performance whereas POP is the worst (even worse than Random).
(5) As expected, serendipity oriented algorithms SPR and UAUM outperform the relevance oriented SVD.

5 DISCUSSION

Our results mostly corresponded to our expectations and the literature on serendipity in recommender systems, i.e.: a) transfer learning improves serendipity (observation 3), b) serendipity oriented recommendation algorithms outperform relevance oriented ones (observation 5) [9, 12] and c) popularity baseline has the lowest serendipity [8]. Our unexpected finding was that the non-personalized algorithm UNPOP outperforms some personalized algorithms (observation 2), which emphasizes the importance of popularity factor for suggesting serendipitous items. This might suggest that popularity is the most important factor for suggesting serendipitous items and that the traditional artificial serendipity metrics [11] reflect the real world scenario. However, answering these questions is beyond research conducted in this paper.

The limitations are mostly caused by the lack of publicly available datasets containing the necessary data. The dataset contains a relatively small number of serendipitous scores. To increase the number of these scores, we marked some items as non-serendipitous for some users. Although, as we explained in Section 3.3, the chance of the mistake is rather small, some items could have been serendipitous to users, while being marked as non-serendipitous ones.

The performance of our approach can be improved with a different configuration of parameters. We used arbitrary learning rate for training GMF and MLP layers, just considering that the loss function should converge (see Section 3.2). We trained them for a limited number of epochs, used the default number of four negative samples per positive sample and also used the default number of four layers in MLP. It is highly probable that further tuning of such parameters would result in further performance gains for *SerRec*.

6 CONCLUSION

This paper presents *SerRec*, a novel approach to use deep neural networks and transfer learning to generate serendipitous recommendations. We employed the Neural Collaborative filtering [3] framework, that we train using a large dataset with relevance scores and then tune using a smaller serendipity oriented dataset. Our approach shows the benefit of transfer learning and improvements over the state-of-the-art serendipity oriented baselines.

In future work, we would like to explore further tuning of the hyper-parameters (number of layers, epochs, etc) of the utilized deep neural networks, to achieve additional performance gains.

ACKNOWLEDGMENTS

The research was supported by KAUTE Foundation and the European Office of Aerospace Research and Development (Grant No FA9550-17-1-0030)

REFERENCES

[1] Oren Barkan and Noam Koenigstein. 2016. Item2Vec: Neural Item Embedding for Collaborative Filtering.. In *MLSP*. 1–6.
[2] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *SIGKDD*. 1809–1818.
[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[4] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *SIGIR*. ACM, New York, NY, USA, 41–48.
[5] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[6] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 77–118.
[7] 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*. ACM.
[8] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.
[9] 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.
[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[11] 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*.
[12] 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.