NAIRS: A Neural Attentive Interpretable Recommendation System

Shuai Yu[†], Yongbo Wang[†], Min Yang[†], Baocheng Li[†], Qiang Qu[†], Jialie Shen[‡]

[†] SIAT, Chinese Academy of Sciences
[‡] Newcastle University, United Kingdom
{shuai.yu,yb.wang,min.yang,bc1.li,qiang}@siat.ac.cn,jialie@gmail.com

ABSTRACT

In this paper, we develop a neural attentive interpretable recommendation system, named NAIRS. A self-attention network, as a key component of the system, is designed to assign attention weights to interacted items of a user. This attention mechanism can distinguish the importance of the various interacted items in contributing to a user profile. Based on the user profiles obtained by the self-attention network, NAIRS offers personalized high-quality recommendation. Moreover, it develops visual cues to interpret recommendations. This demo application with the implementation of NAIRS enables users to interact with a recommendation system, and it persistently collects training data to improve the system. The demonstration and experimental results show the effectiveness of NAIRS.

KEYWORDS

Collaborative filtering, Self-attention network, Interpretable recommendation, Item-based recommendation

1 INTRODUCTION

With the huge volumes of online information, attention has been continuously paid to recommender systems [11, 12]. Item-based collaborative filtering (CF) is one of the most successful techniques in practice due to its simplicity, accuracy, and scalability [7, 8, 10]. It profiles a user with the historically interacted items and recommends similar items in terms of user profiles.

Most of the existing item-based CF methods utilize statistical measures (e.g., cosine similarity) to estimate item similarities. However, the assumption of equal weights is often applied for the items in the measurement [7]. In other words, different items in the historical list are equally treated, which is not true for many of the real-world recommendation applications. On the other hand, interpretable recommendations are of increasing interest, which explain the underlying reasons for the potential user interest on the recommended items. Traditional methods often generate explanations from the textual data such as the content and reviews associated with the items [2–4, 13]. Yet, generating reasons of recommendation remains unsolvable when the texts are unavailable.

Inspired by the recent successes of attention-based neural networks [1] in computer vision and natural language processing, this paper proposes a neural attentive interpretable recommendation system (NAIRS) to alleviate the aforementioned limitations. The key to the design of NAIRS is an self-attention network that computes the attention weights of the historical items in a user profile according to their intent importance associated with the user's preferences. With the learned attention weights, NAIRS provides a high-quality personalized recommendation to users according to their historical preferences. Meanwhile, it interprets the reasons of recommendation by visualizing the learned attention weights for the user's historical list. The function of personalized and interpretable recommendation assists users and manufacturer in identifying results of interest and exploring alternative choices more efficiently. In addition, NAIRS enables users to search for the users who have the similar results and search for the items which are similar to the chosen item. The two functions help the users to discover more potentially interesting items. Furthermore, NAIRS actively records users' interactive behaviors in the system, such as their input queries, liked items, and clicked results. The logged information is then utilized to improve the quality of user profiling for better recommendation.

2 CORE ALGORITHM

We denote a user-item interaction matrix as $\mathbf{R} \in \mathbb{R}^{M \times N}$, where M and N are the number of users and items, respectively. We use $\mathcal{R} = \{(i,j)|\mathbf{R}_{ij} = 1\}$ to denote the set of user-item pairs and use $\mathcal{R}_u^+$ to denote the set of items that user u has interacted with. As described in [7], each item has two embedding vectors $\mathbf{p}$ and $\mathbf{q}$ to distinguish its role of history item and prediction target. The FISM [7] is one of the most widely used collaborative filtering method, which achieves the state-of-the-art performance among the item-based methods. In its standard setting, the prediction of a user u to an item i can be calculated as below:

$$\hat{r}_{ui} = b_u + b_i + (\frac{1}{|\mathcal{R}_u^+|^\alpha} \sum_{j \in \mathcal{R}_u^+} \mathbf{p}_j^T) \cdot \mathbf{q}_i, \quad (1)$$

where b_u and b_i denote the user and item biases, respectively.

Despite the effectiveness of FISM, we argue that its performance is hindered by assigning equal weight to each interacted item. To address this limitation, we propose a neural attentive network to assign different weights to the items according to their intent importance. Mathematically, the prediction of user u to item i can

CIKM'18, Italy

DOI: 10.475/123_4

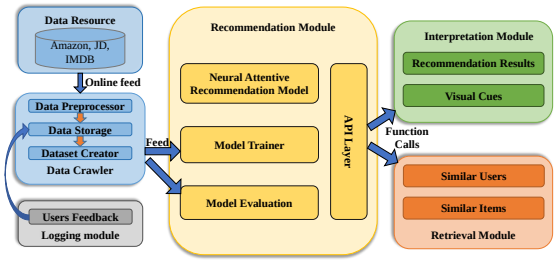


Figure 1: Architecture overview of NAIRS.

be calculated as

$$\hat{r}_{ui} = b_u + b_i + (\sum_{j \in \mathcal{R}_u^+} \alpha_{uj} \mathbf{p}_j^T) \cdot \mathbf{q}_i, \tag{2}$$

where α_{uj} is the attention weight of item j in contributing to user u's representation. Specifically, we exploit self-attention to learn the representation of user u, each of the historical items learns to align to each other. The weight α_{uj} of each historical item j is computed by

$$\alpha_{uj} = \frac{exp(e(\mathbf{p}_j))}{[\sum_{k \in \mathcal{R}_u^+} exp(e(\mathbf{p}_k))]^\beta}, \tag{3}$$

$$e(\mathbf{p}_j) = \mathbf{V}^T g(\mathbf{W} \cdot \mathbf{p}_j + b), \tag{4}$$

where $e(\mathbf{p}_j)$ is an alignment model which scores the contribution of item j to the representation of user u. To form a proper probability distribution over the items, we normalize the scores across the items using *softmax* function and get attention score α_{uj}. β is a smoothing hyper parameter that will be discussed later in this section. V and W are the weight matrices, and $g(\cdot)$ is the activation function. In this paper, we choose *tanh* as activation function for its better performance compared with *relu* and *sigmoid*.

In practice, the standard attention network fails to learn from users' historical data and perform accurate recommendation. By analyzing the attention weights outputted by the model, we reveal that the performance of the model is largely hindered by the *softmax* function, due to the large variances on the lengths of user histories. The attention weights of the items from long history list are largely decreased. To address this problem, we introduce a new symbol β to smooth the denominator of the original attention formula. β can be set in a range of $[0, 1]$. If $\beta = 1$, then Eq. (3) degenerates into the original *softmax*. One typically chooses the value of β between zero and one. This smooth setting leads to much better performance than standard *softmax* function.

Following the strategy in the previous work [5], we treat the observations as positive instances and randomly sample the unobserved items as negative instances. Cross entropy is adopted as the objective function, which minimizes the regularized log loss:

$$L = -\frac{1}{N}\left[\sum_{(u,i) \in \mathcal{R}_u^+} r_{ui} \log \sigma(\hat{r}_{ui}) + \sum_{(u,i) \in \mathcal{R}_u^-} (1 - r_{ui}) \log(1 - \sigma(\hat{r}_{ui}))\right] + \lambda \|\theta\|^2 \tag{5}$$

where N denotes the number of the training instances, θ denotes the parameters of the model.

3 SYSTEM ARCHITECTURE

The proposed NAIRS, overviewed in Figure 1, consists of five main modules. (1) The *Data crawler* module collects user interactive information from various websites such as Amazon, Jindong, and IMDB. (2) The *Recommendation* module produces recommendation results and interpretable partial scores of user-item pairs. (3) The *Interpretation* module visualizes the interpretable reasons of the recommendation by scoring user's historical list. (4) The *Retrieval* module enables the users to i) find people with similar preferences (with historical lists) and ii) explore the items that are similar to a user-specified item. This module effectively assists the users in finding more items in which they may be interested. (5) The *Logging* module collects user behaviors from the system, such as chosen items. The logging information is utilized to further improve the recommender system. In the rest of this section, we elaborate each module of the above.

3.1 Data Crawler Module

The data crawler collects three types of user-item interactive data: (i) movie rating data from IMDB[1]; (2) books rating data from Amazon[2]; and (3) daily goods rating data from Jingdong[3]. For movie rating data, users are selected at random for inclusion. All selected users have rated at least one movie. For book rating data, we focus on the top 1000 popular books. We also collect six categories of daily goods including clothes, shoes, cosmetics, foods, toys, and smart phones. In this work, all user-sensitive information is removed.

3.2 Recommendation Module

We implement the interpretable recommendation algorithm introduced in Section 2 to perform top-n item recommendation. Our recommendation model is implemented with the TensorFlow[4] library and trained on a NVIDIA Titan Xp GPU. After training, we can obtain the user and item representations for each user and item, which are then used to predict the rating scores and assign weights to items in user's historical list for interpretation. In addition, we can obtain the *similar users* and *similar items* results easily with the learned user and item representations. The results learned by Recommendation module can be directly used by the Interpretation module and the Retrieval module.

Note that during the bootstrap process NAIRS provides users a navigation page in which the users can choose the items that they are interested in. This process can alleviate the cold start problem in recommendation to some extent, especially for new users.

3.3 Interpretation Module

Given a user u, the historical items $\mathcal{R}_u$, and a recommended item q_i, the *Interpretation* module provides the top-n recommendation results and interprets the reasons of the recommendation by visualizing the attention scores of user u's historical list $\mathcal{R}_u$. In particular,

[1]https://www.imdb.com
[2]https://www.amazon.com
[3]https://www.jd.com
[4]https://www.tensorflow.org/

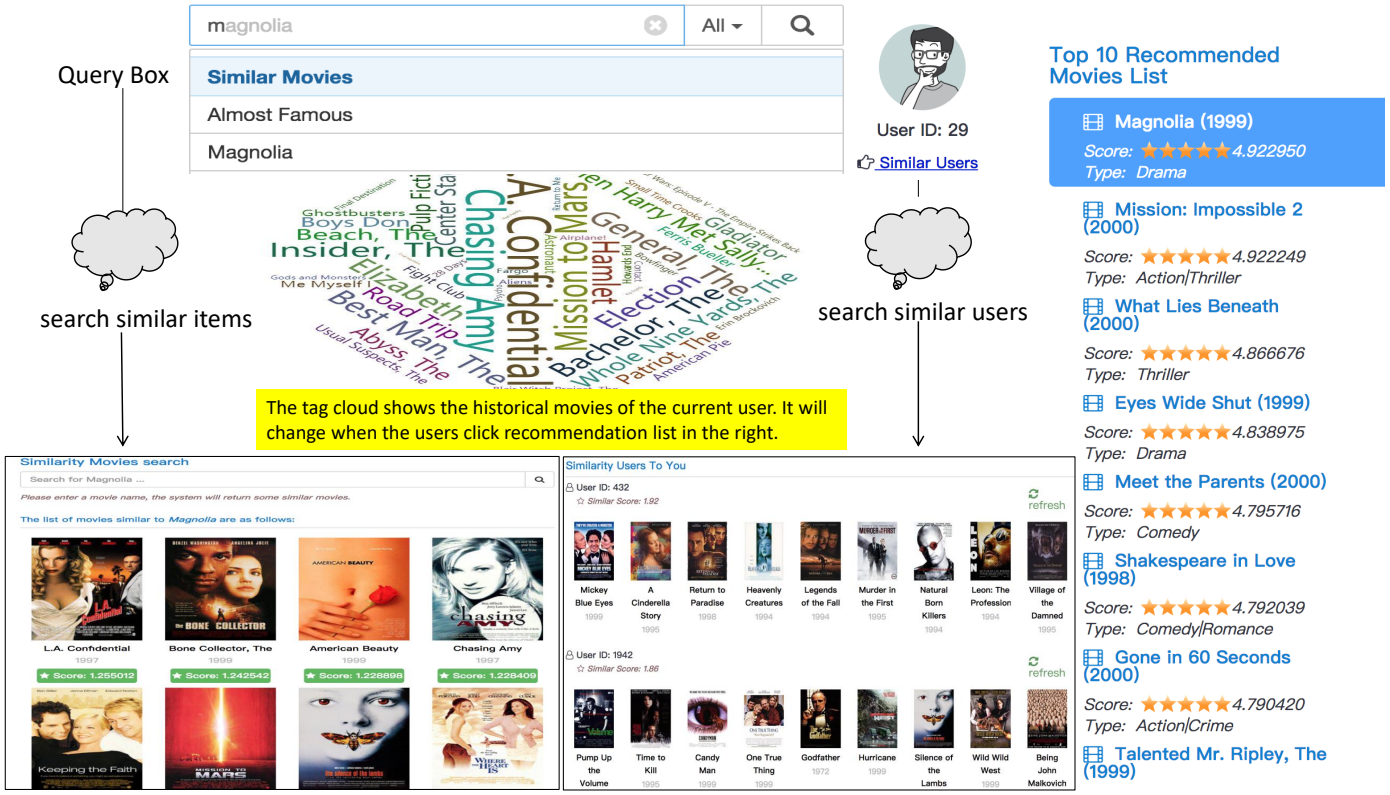


Figure 2: The *Neural Attentive Interpretable Recommendation System*. The top part shows the *Interpretable Recommendation* module, the bottom part shows the *Retrieval Module*. The user's historical interacted movies are displayed in the tag cloud. When the user clicks a movie in the recommendation list, the related movies in the tag cloud will become bigger. The user can either search similar items in the query box or click the movie's name in the tag cloud. The user can also click the link bellow the user logo to explore the users who have similar interests to her/him.

we support the users to add interested movies into their profile list or delete the movies they do not like. NAIRS then demonstrates the recommendation results (on the right of the interface) and interprets the reason of each recommendation with a tag cloud (in the center of the interface), as shown in Figure 2. The importance of each item in the user's historical list is shown with various font sizes. The larger the item names, the more important the items in contributing to the recommendation. For example, the movie *Men in Black* is recommended based on *Nikata* in the user's historical list which has the highest attention score. We show that these two movies both belong to the action movie category. On the other hand, the movie *Escape from New York* contributes little to recommend movie *In the Army Now* since they belong to different categories.

3.4 Retrieval Module

3.4.1 Similar Users. The *Similar Users* module can assist end users to find other users who have similar interests. This module plays an important role in helping the users who might not know exactly what they are looking for to discover potentially interesting items based on the observation that people who agree in the past are likely to agree again. In order to overcome the insensitive of average value, we calculate the similarity between users with *adjusted cosine similarity* as follows:

$$sim(u, u') = 1 + \frac{\sum_k (\mathbf{u}_k - \bar{\mathbf{u}}) \cdot (\mathbf{u}'_k - \bar{\mathbf{u}}_k)}{\sqrt{\sum (\mathbf{u}_k - \bar{\mathbf{u}})^2} \sqrt{\sum (\mathbf{u}'_k - \bar{\mathbf{u}}')^2}}, \quad (6)$$

where $\bar{u}$ and $\bar{u'}$ are the average values over the user's embedding dimensions. Note that we map the value space of the similarity from [-1, 1] to [0, 2] to provide positive similarity scores for better visualization. As shown in Figure 2, we visualize the similar users with their historical lists. In addition, we provide the "like" and "dislike" buttons for users to select/filter the displayed items. These feedback information can be used to update our Recommendation module. In NAIRS, the calculated similarities are cached after updating the model to speed up the results retrieval process.

3.4.2 Similar Items. Intuitively, a user is likely to have similar level of interest for similar items. The *Similar items* module finds items similar to the items liked or chosen by the user. In particular, we provide the end user with a search window for searching any items in the system. Then the items whose similarities are above a threshold are returned as the search results. The similarity between

items is calculated as follows:

$$sim(i, i') = 1 + \frac{\sum_k (\mathbf{i}_k - \bar{\mathbf{i}}) \cdot (\mathbf{i}'_k - \bar{\mathbf{i}}_k)}{\sqrt{\sum (\mathbf{i}_k - \bar{\mathbf{i}})^2} \sqrt{\sum (\mathbf{i}'_k - \bar{\mathbf{i}}')^2}}. \quad (7)$$

Similar to the *Similar Users* component, the similarities between items are cached in the system. When the end user requests similar items, we can obtain the results in $O(1)$ time.

4 DEMOSTRATION

4.1 Demonstration Setup

The *NAIRS* prototype has client and server ends. Clients can access the system by web, mainly for rendering recommendation, interpretation, search, and query results. The server is deployed on Apache Tomcat, which performs the recommendation algorithm and communicates with clients.

4.2 Walkthrough Example

The NAIRS demo consists of the following steps:

Step 1: The user can access to the system by web either on PCs or smart phones. After logging onto the system, the user can select a kind of recommendation service from three categories: movies, books, and daily goods.

Step 2: If the user is new to the system, a collection of randomly chosen items are presented, and the user is asked to choose some items in which the user is interested. After submitting the chosen items, the system offers the top-10 recommendation lists based the chosen items. Furthermore, the system shows tag cloud of the user profile, which reveals why the system recommends the specific items to the user. The user can click any item in the recommendation list, and the user profile tag cloud change accordingly.

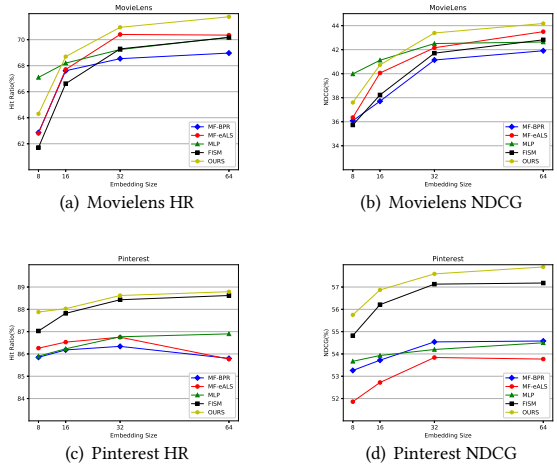
Step 3: The user can query other users that have similar interests by clicking the "similar users" button. Then the similar users with their historical lists are returned to the user, and the user can choose to follow them and find the potentially interesting items via this function.

Step 4: The user can also search the items similar to the item inputed by the user. If our system has items for which the user search, similar items are returned; otherwise, a warning message is shown. Note that to enhance user experience, we implement an Auto-suggestion query box.

5 QUANTITATIVE EVALUATION

In this section, we evaluate the performance of NAIRS quantitatively, then we investigate the interpretation of the proposed system. We conduct experiments on two widely used datasets: Movielens-1M and Pinterest, as the ones used in the study [5]. The results are judged with hit ratio(HR) and Normalized Discounted Cumulative Gain(NDCG), which have been widely used in top-n recommendation [5, 7]. NAIR is compared with several baseline methods including MF-BPR [9], MF-eALS [6], FISM [7], and MLP [5].

The experimental results are shown in Figure 3. We observe that our method outperforms other competitive methods for both of the datasets, which shows the effectiveness of the proposed approach on top-n recommendation used in the demonstration.



(a) Movielens HR (b) Movielens NDCG

(c) Pinterest HR (d) Pinterest NDCG

Figure 3: Performance comparison.

6 CONCLUSION

The paper proposes NAIRS for interpretable recommendation. A self-attention network is introduced to automatically assign different attention weights to interacted items. This attention mechanism can distinguish the importance of the interacted items and provide interpretable recommendations. In addition, the learned user and item embeddings of user profiles can be used in a variety of downstream applications. Both the experiments and demonstration in the paper show superiority of the proposed scheme.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR* (2015).
[2] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect Based Recommendations: Recommending Items with the Most Valuable Aspects Based on User Reviews. In *Proceedings of SIGKDD*.
[3] Muthusamy Chelliah and Sudeshna Sarkar. 2017. Product Recommendations Enhanced with Reviews. In *Proceedings of RecSys*.
[4] Li Chen and Feng Wang. 2017. Explaining Recommendations Based on Feature Sentiments in Product Reviews. In *Proceedings of IUI*.
[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of WWW*.
[6] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of SIGIR*. ACM.
[7] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of SIGKDD*. ACM.
[8] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proceedings of ICDM*.
[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*. AUAI Press.
[10] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW*.
[11] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of SIGIR*. ACM.
[12] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of SIGKDD*. ACM.
[13] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of SIGIR*.