Adaptive Feature Sampling for Recommendation with Missing Content Feature Values

Shaoyun Shi[1], Min Zhang[1]*, Xinxing Yu[2], Yongfeng Zhang[3], Bin Hao[1],
Yiqun Liu[1] and Shaoping Ma[1]
[1]Department of Computer Science and Technology, Institute for Artificial Intelligence,
Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China
[2]Zhihu, Beijing, China
[3]Department of Computer Science, Rutgers University, NJ, USA
shisy17@mails.tsinghua.edu.cn, z-m@tsinghua.edu.cn

ABSTRACT

Most recommendation algorithms mainly make use of user history interactions in the model, while these methods often suffer from the cold-start problem (user/item has no history information). On the other sides, content features help on cold-start scenarios for modeling new users or items. So it is essential to utilize content features to enhance different recommendation models. To take full advantage of content features, feature interactions such as cross features are used by some models and outperform than using raw features. However, in real-world systems, many content features are incomplete, e.g., we may know the occupation and gender of a user, but the values of other features (location, interests, etc.) are missing. This missing-feature-value (MFV) problem is harmful to the model performance, especially for models that rely heavily on rich feature interactions. Unfortunately, this problem has not been well studied previously.

In this work, we propose a new adaptive "Feature Sampling" strategy to help train different models to fit distinct scenarios, no matter for cold-start or missing feature value cases. With the help of this strategy, more feature interactions can be utilized. A novel model named CC-CC is proposed. The model takes both raw features and the feature interactions into consideration. It has a linear part to memorize useful variant information from the user or item contents and contexts (Content & Context Module), and a deep attentive neural module that models both content and collaborate information to enhance the generalization ability (Content & Collaborate Module). Both parts have feature interactions. The model is evaluated on two public datasets. Comparative results show that the proposed CC-CC model outperforms the state-of-the-art algorithms on both warm and cold scenarios significantly (up to 6.3%). To the best of our knowledge, this model is the first clear and powerful model that proposed to handle the missing feature values problem in deep neural network frameworks for recommender systems.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Neural Recommendation Model, Missing Feature Value, Feature Sampling, Feature Interaction

ACM Reference Format:
Shaoyun Shi[1], Min Zhang[1]*, Xinxing Yu[2], Yongfeng Zhang[3], Bin Hao[1],, Yiqun Liu[1] and Shaoping Ma[1]. 2019. Adaptive Feature Sampling for Recommendation with Missing Content Feature Values. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357942

1 INTRODUCTION

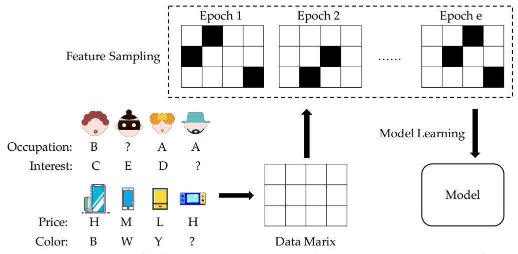

Figure 1: Model Learning with Feature Sampling

Traditional recommendation algorithms include two major approaches, i.e., Content-Based (CB) and Collaborative Filtering (CF) methods. CF methods [10, 15] analyze the user and item historical feedbacks. CB approach [1] takes advantage of the user and item content information for recommendation, such as the occupation and income of users, and the titles, genres of movies.

However, the two types of information are not always available. When a new user or item with no historical information comes into the system, CF models fail to make recommendations, which is called the "Cold-Start" problem – a considerable problem in real-world recommender systems. Content information is unavailable or insufficient sometimes, which refers to missing feature values problem. For example, some items may have particular values on a certain feature but for other items the feature value may be missing, or some users are glad to provide their location and income information, but others may prefer to leave it blank. Although the

* Corresponding Author
This work is supported by Natural Science Foundation of China (Grant No. 61672311, 61532011) and the National Key Research and Development Program of China (2018YFC0831900).

cold-start problem has been well studied, how to handle the MFV problem in recommendation is largely ignored in previous works. The availability of content features can be immensely complicated.

7: Wide&Deep
13: NeuFM
4: LatentX

Also, in deep neural networks, feature interactions like cross feature [7] and Bi-Interaction (BI) layer [13] have been verified to be helpful to improve the recommendation performance. Although multilayer perceptron (MLP) can learn low-rank relations, it is inefficient [4]. Cross features and BI layers are effective ways to help deep neural networks model feature interactions. However, MFVs have negative impacts on model performance, especially models relying on rich feature interactions. In deep neural networks, randomized values, zeros, or specific *Unknown* tags are usually used to fill in the missing values, making the model trainable and thus producing normal outputs. Once the model is trained, the missing value patterns in test cases can be different from what the model has ever seen in training cases and hence lead to errors. Besides, the true values of those unknown features on different users/items may be different from each other, so either considering them as randomized values or as the same *Unknown* label may introduce biases into the model learning. Feature interactions between unreliable representations will lead to even baneful influence. With these traditional treatments, the model will not perform well for the missing feature value cases, and our experiments show that this issue has a terrible impact on model performance.

To mitigate the effects of MFVs, we propose "Feature Sampling" (FS) strategies to help train the models. The brief illustration of the training process with FS is shown in Figure 1. The main idea is to introduce more MFVs into the training process and thus help the model learn a representation of those unknown features. Random FS is further improved by an adaptive sampling and learning process for different data scenarios to give a more robust performance. Note that FS strategies are not specific for one model, but they are verified to be helpful for various models.

With the help of FS strategies, more feature interactions can be introduced to the models in the situation of MFVs. We propose a Content & Context - Content & Collaborate (CC-CC) model, which has a neural Content & Collaborate Module to integrate CF and CB. To enhance the deep neural network, which is good at generalizing, the CC-CC model has a linear Content & Context Module with a strong memorization ability. The model considers both raw features and feature interactions. On the linear Content & Context Module, it takes automatically or manually designed cross-features as inputs. In the neural Content & Collaborate Module, it uses Bi-Interaction layer to model second-order feature interactions and then uses MLP to learn even higher order feature interactions.

The proposed model is able to adjust the weights of information from different resources (historical feedback and content information) to make recommendations with the attention mechanism. When the content parts meet MFVs or unreliable inputs, the outputs will be more dependent on the historical feedback information. In warm scenarios, it considers both kinds of information, but in cold scenarios, it gives higher weights to the content information. Our model is different from previous deep neural network models based on global weights to model different types of information. The attention weights in CC-CC are adaptively adjusted according to the inputs during the testing procedure. The "Cold Sampling" [32] we adopt and the new "Feature Sampling" strategies we proposed help train the attention mechanism. The model has been evaluated on two public datasets and outperforms the state-of-the-art algorithms on different scenarios.

The main contributions of this work are summarized as follows:

(1) We propose an adaptive Feature Sampling learning strategy to help deal with both the cold-start problem and the missing feature value problem. The strategy is verified to be helpful for different models to generate more robust results.
(2) A novel attentive model named CC-CC is designed, which considers both raw features and their interactions to integrate CF and CB recommendation. With the help of Feature Sampling, it works in different scenarios (warm, cold, MFVs).
(3) Experimental results show that our model CC-CC makes significant improvements against the state-of-the-art recommendation algorithms.

2 RELATED WORK

2.1 Deep Recommendation Models

In this work, we focus on deep neural recommendation. In recent years, deep learning has shown its outstanding power in many fields like computer vision and natural language processing. The influence of deep learning has spread to the fields of information retrieval and recommendation. It helps to enhance the ability of traditional methods to model the non-linearity in data. For example, early models such as MDA-CF [26] and CVAE [27] combine matrix factorization with auto-encoders. CNN is a powerful structure to help model the visual information [11, 12]. RNN can be used to model the text data [2] or reviews [25, 38]. Some other research redesigned traditional algorithms in a neural way, for example, NCF [14] adopted a deep learning approach and JRL [37] adopted a representation learning approach to collaborative filtering. Sequential information such as a list of user-item interactions can also be modeled by RNN [17–19] or memory networks [6, 20] to help provide sequential recommendations.

There exist some efficient models to combine content features with end-to-end ID embeddings. Wide&Deep [7] proposed by Google combines the deep neural network with a linear model. It has strong generalization ability based on the deep neural part and memorization ability based on the linear part. In the NFM [13], both sparse IDs and content features are embedded into vectors to model feature interaction. These methods have shown remarkable performance on many datasets and in many scenarios, indicating that both raw features and feature interactions are useful, and integrating the content features and user-item interactions is helpful. However, a disadvantage of these methods is that they use global weights to model the information, which lacks the ability of adaption to better solve for the cold-start and MFV problems.

2.2 Missing Data & Feature Values

In CF methods, the rating matrix of items by users has many unobserved values. Missing ratings are usually assumed to be missing at random. Some works have shown that incorrect assumptions about the missing data can lead to biased parameter estimation and prediction, and non-random missing data assumption performs better [16, 28].

Most of the previous works ignored that content features can also be missing in many situations. Although we can train the model with sufficient content features to handle the cold-start problem, the testing environment in real-world recommender systems is always changing and it usually suffers from the missing feature value problem. A traditional way to solve the problem is to fill the missing features with some valid values, what we call as "Missing Feature Value Imputation". Intuitively, MFVs can be filled with the most frequent feature values, uniformly randomized values, or values generated randomly according to the observed distribution. Furthermore, there exist some works using KNN-based [29] methods or autoencoders [3] to conduct missing data imputation. Most of these works are based on two assumptions: MFVs have the same or similar distribution with the observed feature values, and the data distribution of the training set is similar to that of the testing set. However, the testing distribution could be possibly different from training cases because user records in online environments are dynamic and difficult to predict. As a result, the imputation performance in the testing procedure is usually not guaranteed.

2.3 Cold-Start Recommendation

Cold-start is an important problem in recommendation and must be carefully considered in real-world recommender systems. For example, some works use a hybrid model to generate recommendation results [35, 36]. When new items come, the models give predictions according to the content-based part. Other researchers combine the content features with CF [33, 34]. These approaches take advantages of both CF and CB to handle the cold-start problems. However, they cannot adaptively balance the importance of different types of information for each user-item pair so as to benefit both cold-start cases and recommendation accuracy. ACCM [32] is proposed to address the problem by an attention mechanism to determine whether to use the user-item interactions or the content features to learn the user/item vectors. Warm users and items benefit from both kinds of information, but the vector representations of cold users/items are mainly learned from content features. However, it fails to consider the feature interactions and the adaptive importance between CF and CB, limiting its ability to model content information for the cold-start recommendation. ?

DASC'16
ESWA'17

Meta-Learning
DropoutNet
NIPS'17

3 FEATURE SAMPLING (FS)

Although content features are sometimes easier to collect than historical interactions, they can be partially missing in real-world systems. Sometimes we use a specific *Unknown* tag to represent the missing feature values and learn them in the training procedure. We may also use some predicted or random values to replace the unknown values. However, in the test procedure or real-world online environments, content features can be diverse and complex, and the missing features could be different from the training cases. To alleviate the problem, we aim to introduce more complex and a larger amount of MFVs into the training procedure by sampling and hiding some feature values – without losing any useful information – to train a more robust model that can handle MFVs in the testing procedure. Two new sampling strategies are proposed here, namely random feature sampling and adaptive feature sampling.

3.1 Random Feature Sampling (RFS)

The main idea of RFS is to change some values in each batch data matrix. Let $D = \{d_{i,j}\}$ be the total training data matrix, where each row contains the user ID, item ID, and their feature values. $d_{i,j} = Unknown$ represents that the feature j in row i is unknown. Then, we randomly change some feature values by:

$$d'_{i,j} = \begin{cases} d_{i,j}, & s_{i,j} = 0 \\ \text{"Unknown" Tag}, & s_{i,j} = 1 \end{cases} \tag{1}$$

where $s_{i,j} \in \{0, 1\}$ is a random variable used to control if $d_{i,j}$ is to be changed or not, and it is generated from $Bernoulli(\beta)$. β is the FS ratio controlling how many feature values are sampled in the strategy. When $s_{i,j} = 1$, no matter whether $d_{i,j}$ is an observed feature value or not, it is modified to be unknown. The whole RFS process is shown in Figure 2.

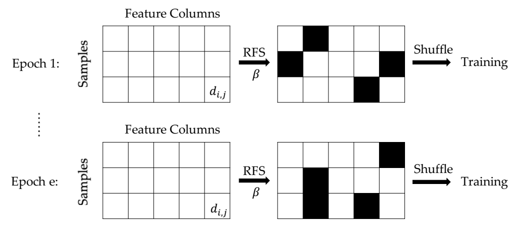

Figure 2: Illustration of Random Feature Sampling. The matrix is the training data matrix $D = \{d_{i,j}\}$, where white/black blocks represent observed/unknown feature values, respectively.

In each epoch, the $s_{i,j}$ is regenerated so that different batch samples different features. Overall, as the training procedure advances, it tends to use all the information available. In this way, RFS brings more unknown feature values into the training procedure, so as to force the model to handle complex missing feature value situations and make the model more robust.

3.2 Adaptive Feature Sampling (AFS)

The previous RFS strategy brings some MFV into the training process. However, in real-world data, some feature columns may be relatively complete but others may be mostly missing. For those features that are mostly complete, sampling some of them to be unknown indeed brings more difficult training instances and forces the model to learn how to handle the missing values. However, if a feature column has only a few values and FS still changes some values to *Unknown* in each epoch, there may not be sufficient values to learn the features. RFS uses a constant probability β for all feature columns, and it needs to be manually tuned for satisfactory performance.

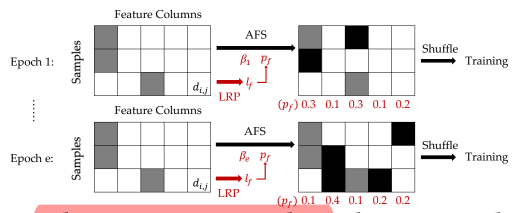

Figure 3: Adaptive Feature Sampling. The matrix is the training data matrix $D = \{d_{i,j}\}$. White blocks are observed feature values, gray blocks are those feature values that are naturally missing in the dataset, and black blocks are sampled feature values replaced as *Unknown*. β_e is to control how many values are sampled as unknown and p_f determines which features have higher probabilities to be sampled.

Thus, we proposed the Adaptive Feature Sampling (AFS) strategy to address the problem. The process is summarized in Figure 3. There are mainly two differences from RFS:

- *FS ratio.* During the training process, AFS will gradually increase the FS ratio β_e in each epoch e until it gets a maximum FS ratio β':

$$\beta_e = min(\beta', \beta_0 + \Delta \cdot e) \tag{2}$$

where β_0 is the initial FS ratio in the first epoch and Δ is the increment of β_e in each epoch. β_e determines how many values should be sampled as unknown in the data matrix in each epoch e.

- *Feature importance.* The intuition is that if the model depends heavily on some features, then missing those features will significantly influence the model performance. The solution is to introduce some samples that miss those important features during training to make the model more robust. In each epoch, we first calculate the relevance of the features to the model predictions by the Layer-wise Relevance Propagation (LRP) [5] technology. We assume that if the LRP results show that a feature contributes more to the final results of the sample, then it is more important than other features. To avoid that missing too much important features harm the training process, a small number of important features are sampled to be unknown at the beginning of the training process, but increasingly more important features will be sampled along with the training.

The LRP results keep changing along with the training process. In our experiments, we generate new LRP results during each epoch. Formally, let the l_f be the LRP feature importance of the feature f, ($\sum_f l_f = 1$). Then the probability to sample the value of feature f in a training instance (a row of the data matrix) is

$$p_f = \begin{cases} l_f, & l_f \leq \sigma(e) \cdot max\{l\} \\ \delta * l_f, & otherwise \end{cases} \tag{3}$$

where e is the current epoch, and $\sigma(e)$ is a monotonically increasing function used to adjust the threshold $\sigma(e) * max\{l\}$, so that the important features (l_f bigger than the threshold) have lower probabilities ($0 < \delta < 1$ is used to decrease the probabilities) to be sampled as unknown at the beginning of the training process. Note that p_f needs normalization so that $\sum_f p_f = 1$.

By doing these, AFS has at least two advantages compared to the simple RFS strategy. Firstly, the gradually increasing sampling probability provides a more robust training process. The max FS ratio β' is the parameter we should take into consideration, but it is less sensitive than a fixed FS ratio β. Secondly, AFS adaptively prevents the model from relying heavily on some features. If the values of some feature columns are mostly available or they are more important to the model, they will contribute more to the predictions, and LRP will be able to detect them by gradually sampling more of these features to be unknown.

3.3 Discussion on Feature Sampling

FS strategies are motivated by the "Cold Sampling" (CS) strategy [32]. The CS is designed for the attention mechanism in deep neural recommendation models. In traditional gradient-descent-based training processes, attention fails in cold-start scenarios while evaluating the vectors from historical information. This is because there are no cold samples in the training process – the model sees all users and items in the training set, but during the testing procedure, a new user's vector in the CF part remains untrained and its distribution is different from those trained vectors. In this situation, the output of the attention network will be unreliable. The key point to solve this problem is to introduce cold users/items into the training procedure, and hence help the model learn the attention weights in diverse cases. Shi et al. [32] proposed the CS training strategy to help the attention network learn what is cold-start and how to handle it.

However, FS strategy is different from CS. It is somehow a generalization of CS. If we regard the user/item IDs as some features and FS is conducted on the whole input data, including the user/item IDs, and let *Unknown* denote a cold user/item ID. Then we can find that by changing the user/item IDs to *Unknown* in some samples, FS brings some cold users/items into the training procedure, and a new ID that the model has never seen is considered as an *Unknown* ID. In this way, FS helps the model to learn a global embedding for the *Unknown* ID, which is a specific user/item ID. The embedding is used as the representation of all cold users/items, which can help to make recommendations in cold scenarios. Unlike FS, however, CS replaces the embeddings of user/item IDs with random vectors as the input of the attention network. It directly teaches the attention network to learn what the cold vectors look like. Then the attention network gives small weights to the cold user/item CF embeddings during testing. Overall, they both introduce some cold users/items into the training process, but CS focuses on teaching the attention network to adjust between different information resources – content or collaborative filtering. The FS strategy, instead, is not designed specifically for the attention network. So what is most important is that FS can be adopted on different deep neural models such as Wide&Deep [7] and NFM [13]. In our experiments, we conducted FS on content features in all of the models, while CS is conducted on the attention network of the ACCM [32] and our further proposed CC-CC model.

4 CONTENT & CONTEXT - CONTENT & COLLABORATE MODEL

With the help of FS strategies, more feature interactions can be utilized without worrying too much about MFVs. To verify the effects of FS strategies, the Content & Context - Content & Collaborate Model (CC-CC) is proposed, which uses attention mechanism to adaptively adjust the weights of different information resources. Further experiments show that FS strategies help CC-CC work on different scenarios. The model structure is in Figure 4. It mainly has two modules, a linear Content & Context Module and a deep Content & Collaborate Module. Their outputs are finally merged to make predictions. Detailed structures are introduced in the following subsections one by one.

4.1 Content & Collaborate Module

CF and CB are based on different kinds of information. We believe that integrating the advantages of CF and CB can improve recommendation performance. In our model, the neural collaborative filtering vectors and content features are separated and finally combined by attention mechanism. The purpose of the attention network is to adjust the information source, so that the model can use the correct recommendation strategy in different scenarios (e.g., cold or unreliable content features). User and item information is used to form the user vector **u** and item vector **v** independently.

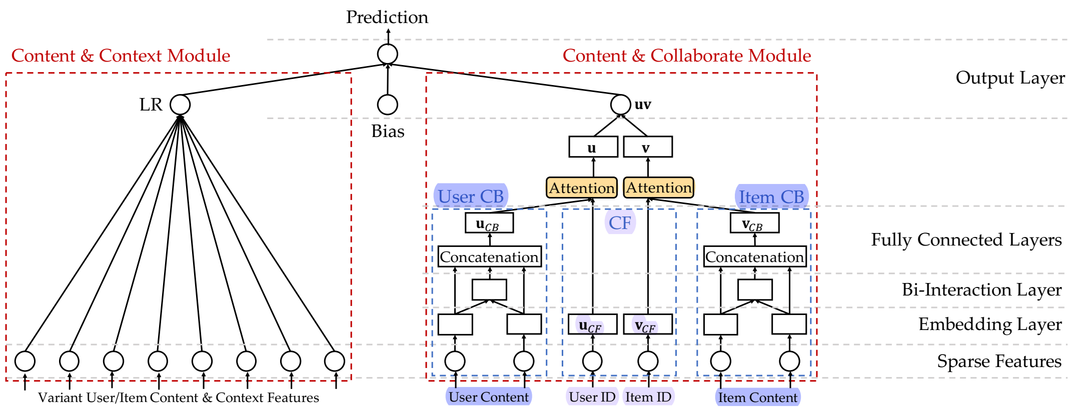

Figure 4: Content & Context - Content & Collaborate Model.

Note that in the formulas, bold fonts are used to represent vectors and italic fonts mean scalars or entities.

4.1.1 Neural Collaborative Filtering. CF usually performs better with sufficient feedback. An effective CF structure is necessary for a good recommender algorithm. In neural networks, it is common to transfer sparse IDs into vectors through the embedding layer. Also, in neural recommendation models, user and item IDs are embedded as vectors [13, 14]. Similarly, in CC-CC, the user ID and item ID are embedded as a user CF vector $\mathbf{u}_{CF}$ and an item CF vector $\mathbf{v}_{CF}$, respectively. The embeddings can be initialized randomly or with pre-trained CF vectors, and they are trained end-to-end by the historical feedbacks during the training procedure.

NeuFM
NeuCF

4.1.2 Neural Content Interaction. The content features of users and items are usually diverse which contain real-values, multi-classes or even texts and images. To feed them into the neural networks, they are usually preprocessed and transferred into discrete and sparse one-hot or multi-hot vectors. The sparse inputs are then transferred into vectors through the embedding layers. In the deep neural parts of Wide&Deep [7] and ACCM [32], they concatenate these embedding vectors and simply use fully-connected layers to model them. Although concatenation and fully-connected layers are the basic operations in the neural networks, they can hardly learn feature interactions. According to known experiences of feature engineering, feature interactions are helpful to learn the relationship between the features and the label. For example, the model may remember that a highly educated engineer ({education=Ph.D., occupation=engineer}) has a high income ({income=high}). Feature interactions make the model easier to remember this and are verified important to recommender systems [7, 13, 30]. One of the traditional ways to do feature interactions is Factorization Machine. In CC-CC, we adopt the Bi-Interaction (BI) layer to model the feature interactions. The BI layer is used in [13] and it works similarly as Factorization Machine in a neural way. Formally, let x_i be the feature value and $\mathbf{f}_i$ be the embedding vector of the i-th feature. Then the output vector of the BI layer is

W&D
NeuFM
FM

$$BI(\mathbf{x}) = \sum_i \sum_j x_i \mathbf{f}_i \odot x_j \mathbf{f}_j \tag{4}$$

The $\odot$ here means element-wise product. The process of BI layer can be regarded as two steps. The first step is calculating the feature interactions $x_i \mathbf{f}_i \odot x_j \mathbf{f}_j$ for each feature pair i, j. The second step is the sum-pooling which does an element-wise sum of these interaction vectors. It can also be the average-pooling here, and we take sum-pooling because it is easier to implement. So the output vector of the BI layer has the same size as feature vectors $\mathbf{f}_i$. Note that the BI layer can be computed in linear time by reformulating the Equation 4 as:

$$BI(\mathbf{x}) = \frac{1}{2}[(\sum_i x_i \mathbf{f}_i)^2 - \sum_i (x_i \mathbf{f}_i)^2] \tag{5}$$

Then it can be finished in $O(kN_x)$ time, the same as direct max (average) pooling of the first-order feature vectors, where k denotes the size of embeddings and N_x denotes the number of non-zero entries in the input vector.

The output vector of BI layer is concatenated with all the first-order feature embeddings, and the long concatenation vector is then fed into the fully-connected layers to calculate the user CB vector $\mathbf{u}_{CB}$ or item CB vector $\mathbf{v}_{CB}$. By taking this kind of feature interactions, the model captures second-order feature interactions in the low level, which can hardly be learned by fully-connected layers in Wide&Deep or ACCM. The formed interaction vector brings more information to the higher layer and provides a probability for fully-connected layers to learn even higher-order and non-linear feature interactions.

4.1.3 Attention Mechanism. In different scenarios, e.g. for heavy or new users, CF and CB may have different importance for a personalized recommendation. Many previous models like Wide&Deep and NFM do not consider this and use global weights to model different types of information, and thus they do not have the ability to adapt the importance of CF and CB to solve the cold-start and MFV problems. We think it is a better way to combine different information sources by attention mechanism by dynamic weights for each user (item). Not only in the warm scenario but also it is of great significance in the cold start scenario [32]. Besides, attention can distinguish content vectors from unreliable feature inputs, which helps reduce the impact of the MFV.

Taking the user CF vector $\mathbf{u}_{CF}$ and user CB vector $\mathbf{u}_{CB}$ as the inputs, the attention weights are calculated as follows:

$$\begin{aligned} lh^u_{CF} &= \mathbf{h}^T f(\mathbf{W}\mathbf{u}_{CF} + \mathbf{b}), \\ h^u_{CB} &= \mathbf{h}^T f(\mathbf{W}\mathbf{u}_{CB} + \mathbf{b}), \\ a^u_{CF} &= \frac{exp(h^u_{CF})}{exp(h^u_{CB}) + exp(h^u_{CF})} = 1 - a^u_{CB} \end{aligned} \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{t \times k}, \mathbf{b} \in \mathbb{R}^t, \mathbf{h} \in \mathbb{R}^t$ are the parameters of attention network, and t denotes the hidden layer size of the attention network. f is the activation function which can be *relu*, *tanh*, *sigmoid*, etc. The attention weights of item CF a^v_{CF} and CB a^v_{CB} are calculated in a similar way. The parameters of attention networks in the item part are shared with the user part because they have the same purpose to judge whether a vector is informative.

Then the user and item vectors are formed by the weighted sum:

$$\mathbf{u} = a^u_{CF}\mathbf{u}_{CF} + a^u_{CB}\mathbf{u}_{CB}, \quad \mathbf{v} = a^v_{CF}\mathbf{v}_{CF} + a^v_{CB}\mathbf{v}_{CB} \tag{7}$$

and the user and item vectors $\mathbf{u}$, $\mathbf{v}$ are used for further predictions.

4.2 Content & Context Module

Deep neural networks have strong generalization ability. By modeling the samples in the training set, they can predict with unseen or rarely seen feature combinations. However, they are weak in memorization, compared with traditional linear models, which can learn the direct relationships between features and final predictions [7]. For example, young teenagers may love comics more than the elders. Besides, except for the user and item interaction and content information, there is various informative context information in recommender systems such as time and other session-related features. Therefore, our CC-CC combines the deep Content & Collaborate module with a linear Content & Context module, inspired by Wide&Deep [7].

Taking the sparse features $\mathbf{x}$ as input, it calculates a linear regression of the features:

$$LR(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_i w_i x_i \tag{8}$$

where $\mathbf{w}$ is the weight vector and w_i is the weight of the i-th feature.

The module is also enhanced by the feature interactions. Different from the BI layer, cross features are used to better take advantages of the memorization ability of the linear module and form more variant features:

$$c_{\{x_j\}} = \prod_j x_j \tag{9}$$

where $\{x_j\}$ is a subset of raw features $\mathbf{x} = \{x_i\}$ and it means which features the cross-product is conducted on. For example, $x_i = 1$ means the user is from Japan and $x_j = 1$ means the book is in Italian. Then in a second order cross-product, since generally a Japanese does not prefer a book written in Italian, this $c_{\{x_i,x_j\}} = x_i x_j = 1$ will be negative; while if x_j is the book in Japanese, $c_{\{x_i,x_j\}} = x_i x_j = 1$ is positive. Cross features $\mathbf{c}$ can be automatically generated or manually designed with experience. The cross features $\mathbf{c}$ finally are fed into the linear regression together with the raw features.

4.3 Output Layer

To combine the Content & Context Module and the Content & Collaborate Module, we use a summarization layer for final prediction:

$$y = LR(\mathbf{x}|\mathbf{c}) + \mathbf{uv} + b \tag{10}$$

where b is the total bias, which is a summarization of the global bias, user bias and item bias:

$$b = b_g + b_u + b_v \tag{11}$$

The output of the Content & Collaborate Module, user vector $\mathbf{u}$ and item vector $\mathbf{v}$, are used to conduct a dot product. Surely they can be fed into more fully-connected layers or some other deep neural structures. We use the dot product in our tasks due to the computational efficiency and effectiveness, which is also commonly used in the literature [14, 32].

The summarization layer is used to keep the linearity between the Content & Context Module and the final predictions so that it keeps the memorization ability. The experiments of our model are conducted on rating and click prediction tasks, so linear regression is adopted.

4.4 Loss Function

We use the *RMSE* loss to train our model. We not only are concerned about the final predictions, but also expect that the predictions of both CF and CB parts themselves are accurate. As a result, the final loss is formulated as follows:

$$L = RMSE(\mathbf{y}, \mathbf{l}) + \gamma[RMSE(\mathbf{y}_{CF}, \mathbf{l}) + RMSE(\mathbf{y}_{CB}, \mathbf{l})] + \tau||\Theta||_2 \tag{12}$$

The first term is the difference between the final predictions and the labels, and the second term is used to guide the CF and CB parts to generate meaningful and comparable vectors, where γ is the weight of this term, $\mathbf{y}_{CF} = LR(\mathbf{x}|\mathbf{c}) + \mathbf{u}_{CF}\mathbf{v}_{CF} + b$ and $\mathbf{y}_{CB} = LR(\mathbf{x}|\mathbf{c}) + \mathbf{u}_{CB}\mathbf{v}_{CB} + b$. It encourages CF and CB to learn their own structures and prediction abilities. The attention network evaluates the vectors and gives different weights to different information resources so that the model takes the best advantages of CF and CB even if there is no content or historical information. The last term is a L_2-regularization, where τ is the L_2-weight, and Θ represents all the model parameters.

4.5 Model Discussion

Table 1: Comparison with State-of-the-Art Models

	Adaptive Weights	Feature Interactions
Wide&Deep	✗	Cross Feature
NFM	✗	Bi-Interaction
ACCM	Attention	✗
CC-CC	Attention	Cross Feature,Bi-Interaction

Our proposed CC-CC aims to model different types of information adaptively for each user and item. Wide&Deep, NFM and some other deep models are also able to handle content, context and interaction information, but they use global weights for all users and items. However, disadvantages exist when some information sources are unavailable or unreliable, e.g., the problem of missing feedbacks in cold-start scenarios. Our CC-CC model uses the attention mechanism to adjust the recommendation strategy, which is adaptive to different scenarios. In particular, CC-CC uses Content & Context Module and Bi-Interaction layer to enhance the utility of content and context information. Moreover, it considers the feature interactions and improves the memorization ability of deep neural recommendation models. Comparison of our CC-CC with some state-of-the-art models is concluded in Table 1.

5 EXPERIMENTAL SETTINGS

5.1 Datasets

We conducted our experiments on two datasets: ML-100k and ZhiHu. Both datasets can be easily downloaded from the Internet. Some detailed information of the datasets is shown in Table 3.

Table 2: Overall Performance

	ML-100k RMSE, lower is better			ZhiHu AUC, higher is better		
	Random	+Cold[1]	+MFV[2]	Random	+Cold[1]	+MFV[2]
Wide&Deep [7]	0.9097	0.9909	0.9877	0.6963	0.6757	0.6651
NFM [13]	0.9118	0.9822	0.9967	0.7264	0.6877	0.6873
ACCM [32]	*0.9018*	*0.9727*	*0.9739*	*0.7314*	*0.6885*	*0.6890*
CC-CC	**0.9012**	**0.9599***	**0.9664***	**0.7403***	**0.6983***	**0.6962***

1. Test sets: randomly 30% item cold, 30% user cold.
2. Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.
*. Significantly better than the best baseline (ACCM, italic ones) with $p < 0.05$

Table 3: Statistics of Evaluation Datasets

Dataset	Interaction#	User#	Item#	Sparsity
ML-100k	100,000	943	1,682	93.70%
ZhiHu	1,334,168	13,537	61,661	99.84%

• **ML-100k**. It is maintained by Grouplens , which has been used by many researchers for many years. It includes 100,000 ratings ranging from 1 to 5 from 943 users and 1,682 movies. Content information we used is the age, gender, occupation of users and release year, genres of items.

• **ZhiHu**. ZhiHu is a Chinese platform which allows users to ask or answer questions on it. It helps users share their knowledge and experiences with others. It held a competition and published its data on Biendata , which is available to the general public. The dataset provides a huge amount of users, items (answers) and their interactions, and aims at predicting whether a user is interested in an answer. User features include the ID, device, position, topic, total counts of user behaviors, etc, and items features include ID, time, total like/dislike counts, and other user behavior counts such as the number of clicks, etc. Privacy information is hidden or replaced by IDs.

5.2 Baselines

Our model CC-CC is compared with several state-of-the-art neural recommendation models as follows:

• **Wide&Deep** [7]. Google proposed it in 2016, which combines the deep neural network and linear models. It is verified as one of the best deep neural recommendation models.

• **NFM** [13]. The Neural Factorization Machine model proposed in 2017, which uses Bi-Interaction Layer to model feature interactions.

• **ACCM** [32]. The Attentional Content & Collaborate Model proposed by in 2018. It works on both warm and cold scenarios, and the model adopted a "Cold-Sampling" (CS) strategy to help the attention network learn how to handle the cold data.

These works have compared their models with some famous recommendation algorithms such as ItemKNN [8], UserKNN [22], LibFM [31], BiasedMF [24], SVD++ [23] and surpassed them. Here we do not compare CC-CC with these algorithms and focus on the three state-of-the-art neural recommendation models.

https://grouplens.org/datasets/movielens/100k/
https://www.zhihu.com/
https://biendata.com/competition/CCIR2018/data/

5.3 Evaluation

Each dataset is split into the training (80%), validation (10%) and test (10%) sets. To construct the evaluation sets in different cold ratios, for example, 30% item cold, we randomly choose 30% samples in the validation and test sets, and give each sample a new unique item ID. The validation set is used to conduct early stopping and optimize the hyper-parameters, and the test set is used to evaluate the model performance. To construct MFVs, we randomly choose some values in the feature matrix and give each value an *Unknown* tag. To evaluate the models, for the rating prediction task on ML-100k, we use Root Mean Square Error (RMSE, lower is better), and for the click prediction task on ZhiHu, we use Area Under Curve (AUC, higher is better). For a fair comparison, we do not use cross features in our model or the Wide&Deep model.

5.4 Parameter Setting

We use Adagrad [9] to train the model with mini-batches at the size of 128. NFM and Wide&Deep are trained in regular mini-batches, ACCM and CC-CC use "Cold-Sampling" (CS) to train the attention network to handle cold scenarios. The learning rate is searched for each model from 0.001 to 0.1, and early-stopping is conducted according to the performance on the validation set. Embedding size k of user IDs, item IDs and features are set to 64 and batch Normalization (BN) [21] is conducted on the fully-connected layers. To prevent the model from overfitting, we use both the L_2-regularization and dropout. The weight of L_2 τ is set between 1e-5 to 1e-3 and dropout ratio is set between 0.05 to 0.2. The CS and FS ratios $\alpha, \beta(\beta')$ are set between 0.0 to 0.2. In AFS, we set $\beta_0 = 0.1$ and $\Delta = (\beta' - \beta_0)/20$ so that the FS ratio β_e starts from 0.1 and achieves the max value around the 20th epoch, $\sigma(e) = \frac{1.1}{1+\exp(3-e)}$ and $\delta = 0.1$. We provide the codes for CC-CC in Github at https://github.com/THUIR/CC-CC.

6 EXPERIMENTAL RESULTS

We conduct our experiments mainly to answer the following questions:

RQ1 Does the specific Feature Sampling help handle the missing feature values?
RQ2 How does CC-CC perform compared with the state-of-the-art methods?
RQ3 Does Feature Sampling work for different models?

6.1 Performance Comparision (RQ1, RQ2)

6.1.1 Overall Performance. Firstly, the overall performance is shown in the Table 2. We will give some further studies on FS and model in later sections.

All models are trained with AFS. From the results, we conclude that no matter on the warm/cold data or the data with MFV, CC-CC achieves the best performance. It shows that using the attention mechanism to adjust different information resources is helpful. CC-CC performs better than ACCM, especially in cold data because it has both the deep module and linear module, which considers both raw features and feature interactions, highlighting its advantage of modeling content features on cold data.

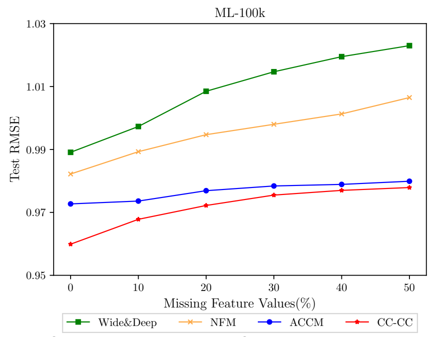

Figure 5: Different Amounts of Missing Feature Values. Test sets: randomly 30% item cold, 30% user cold.

We also test the models' performance on datasets suffering from different amounts of MFV. The results are shown in Figure 5. A percentage of feature values (from 0 to 50%) are randomly removed based on the test set that contains 30% cold users and 30% cold items. Models are trained with AFS. From the results, we can observe that the more feature values are missing, the worse the models perform, which is intuitive because MFV means missing information and unreliable inputs. Generally, our model CC-CC is better than the other baselines when the MFV problem exists in the input data.

6.1.2 Ablation Study. We did some ablation study to understand the effects of FS and feature interactions better.

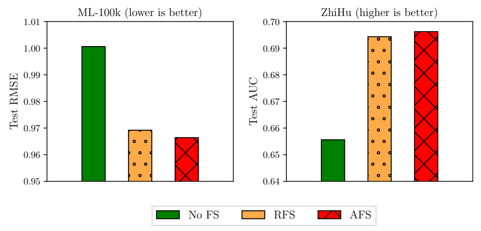

Figure 6: Performance of CC-CC with different Feature Sampling strategies (No FS, RFS, AFS). Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.

We compare the different training strategies: no FS, RFS and AFS. The results are shown in Feature 6. Models perform badly on data with MFV without FS, showing that the problem should be carefully considered, and FS helps to handle the MFV problem. The strategy introduces more complex feature situations into the training procedure and forces these models to learn how to predict with MFV and unreliable content information. AFS is slightly better than simple RFS because it analyzes the feature importance and provides a more adaptive sampling. Our further experiments in 6.2.2 show that AFS is better than RFS generally on other models and parameters (Figure 8, 9).

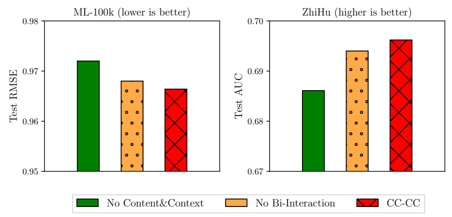

Figure 7: Performance of CC-CC w/o Content & Context Module and Bi-Interaction Layer. Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.

We remove the Content & Context Module or the Bi-Interaction Layer in the Content & Collaborate Module to observe whether they can improve the model performance, as shown in Figure 7. It is clear that CC-CC without either part performs worse. Content & Context Module provides cross-features and a linear memorization ability. The Bi-Interaction Layer models feature interactions with deep neural networks. Both parts are helpful, showing that feature interactions are important to the model performance, while Content & Context Module brings more improvement.

6.2 Impact of Proposed Feature Sampling Strategies (RQ1, RQ2)

In this section, we explore the impact of proposed Feature Sampling strategies. Experiments are conducted on a test set that contains 30% cold users, 30% cold items and 10% MFV, respectively.

6.2.1 Representations of MFV. In the experiments, we use a specific *Unknown* tag to represent the MFV. For example, in the age column there are three values 0,1,2 to represent *Unknown*, *Adult* and *Child*, respectively. Deep neural models embed these discrete features into vectors, and AFS training strategy helps learn these embeddings. In real-world applications, there are also some other ways to represent the MFVs, such as using the most popular feature value, randomizing a value uniformly or according to the distribution of the observed feature values. We compare these strategies with AFS and the results are shown in Table 4.

Table 4: RMSE under Different Representations of MFV.

ML-100k[1]	Wide&Deep	NFM	ACCM	CC-CC
Unknown Tag[2]	1.0410	1.0127	0.9796	1.0006
Random[3]	1.0126	1.0112	0.9882	0.9909
Frequency[4]	1.0077	1.0034	0.9821	0.9721
Distribution[5]	1.0037	0.9999	0.9827	0.9735
AFS	0.9877	0.9967	0.9739	0.9664*

1. Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.
2. Leave the MFV as *Unknown* tags in training and testing procedure without any specific processing.
3. Randomize feature values uniformly to replace MFV.
4. Replace the MFVs with the most frequent feature values in each feature column.
5. Randomize feature values according to the distribution of each feature column to replace the MFVs.
*. Significantly better than other results with $p < 0.05$

Based on the results, leaving the MFVs as the *Unknown* tag in training and testing procedure without any specific processing brings the worst performance (*Unknown* Tag). It means that MFV should be carefully considered because it brings uncertainty into the systems, leading to unfavorable impacts on the performance. Replacing the *Unknown* tag with some feature values can help a little bit (Random, Frequency, Distribution) because these methods construct meaningful inputs that may somehow match the true values. Intuitively, using the most frequent feature value (Frequency) or randomly according to the distribution of the known feature values (Distribution) is better than randomize a feature value uniformly (Random). The reason is that the first two methods give better simulation to the real distribution of the feature values.

However, all these methods force the model to make predictions according to some unknown or wrongly imputed feature values, which is undesirable. What we want is that the model can recognize MFVs and even learn the meaningful representations of the unknown feature values. Using the *Unknown* tags in the test procedure is fine, but the model should be taught how to handle these tags and what those tags mean. The distributions of the known and unknown feature values are different, and neither of them is the same as the distributions of the training set and testing set. AFS introduces more complex instances of missing feature values into the training procedure, and then forces the model to learn how to handle unreliable inputs. In this way, the model becomes more robust and performs better than other methods. The results show that CC-CC with a powerful content part suffers more from the MFV problem, and ACCM suffers the least because it has an independent CF part for rating prediction. But finally, CC-CC with AFS achieves the best performance.

6.2.2 *Different Feature Sampling Ratios.*

We test the performance of the models when training with different ratios of FS (Figure 8):

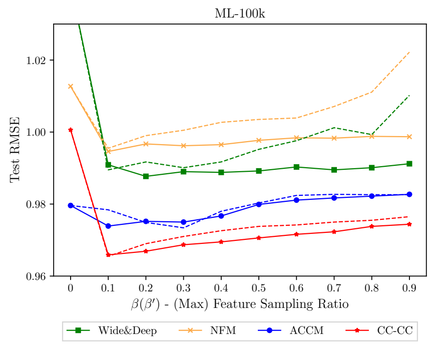

Figure 8: Performances of Different Feature Sampling Ratios. Solid/Dashed lines are experiments with AFS/RFS. Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.

The results show that a small ratio of FS (0.1-0.2) is enough for different models. Different models perform best on different feature sampling ratios. But a too large FS ratio will affect the model performance because it prevents the models from learning the relationship between the content features and the final predictions. AFS with large FS ratio performs better than RFS because the FS ratio is adaptively increasing during the training process instead of a fixed ratio. Performance of CC-CC and ACCM are more stable even with few content features thanks to the CF module in their structures. Generally, AFS performs better than RFS in most situations. AFS is less sensitive to the changes of the FS ratio because it provides an adaptive way to sample different amounts of values during the training process.

6.3 Feature Sampling Strategies for Different Algorithms (RQ3)

To verify that FS can help different algorithms, we conduct some experiments on the test sets with randomly 10% feature values missing and 30% items cold, 30% users cold, as shown in Figure 9.

Results show that without FS, Wide&Deep, NFM, and ACCM all suffer from the MFV problem. The reason is that all of the models are trained with sufficient content features. The training procedure does not carefully consider the MFV problem, but the testing set contains many MFVs and has a different data distribution with the training set. ACCM suffers the least because it has an independent CF part (and a simple CB part) for prediction. FS brings MFVs into the training process and improves the model performance. AFS also performs better on other models than RFS.

According to Figure 8 in the previous section, AFS performs better than RFS in most cases with different feature sampling ratios.

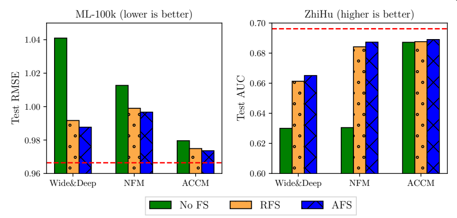

Figure 9: Performance of Wide&Deep, NFM and ACCM with different Feature Sampling strategies (No FS, RFS, AFS). The red dashed line is the CC-CC model with AFS. Test sets: randomly 30% item cold, 30% user cold, and 10% feature values missing.

7 CONCLUSION

In this work, Random Feature Sampling and Adaptive Feature Sampling strategies are proposed to handle the missing feature values, which is an important problem that has not been well studied in previous works. The FS strategies are simple but work for different models including Wide&Deep, NFM, and ACCM.

With the help of FS strategies, more feature interactions could be taken into consideration, which suffer greatly from the MFV problem. A novel Content & Context - Content & Collaborate model is proposed to leverage different types of information: feedback, content, and context. In both modules of the model, it utilizes both raw features and feature interactions. It is verified that CC-CC works not only on both warm and cold scenarios, but also with MFVs. Finally, CC-CC achieves better performance than the state-of-the-art methods on two different datasets.

However, giving reasonable outputs with missing feature values is not the terminal of our work. Our further study will work on using observed interactions and feature values to recover some missing feature values. We can use convinced values to partly replace the missing values, which to some extent is different from the missing feature value imputation approach, and it will be helpful to both user modeling and generating personalized recommendations.

REFERENCES

[1] Charu C Aggarwal. 2016. Content-based recommender systems. In *Recommender systems*. Springer, 139–166.

[2] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 107–114.

[3] Brett K Beaulieu-Jones and Jason H Moore. 2017. Missing data imputation in the electronic health record using deeply learned autoencoders. In *Pacific Symposium on Biocomputing 2017*. World Scientific, 207–218.

[4] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. ACM, 46–54.

[5] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA) 2016*. Springer, 913–922.

[6] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. ACM, 108–116.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[8] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[9] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research* 12 (2011), 2121–2159.

[10] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction* 4, 2 (2011), 81–173.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 507–517.

[12] Ruining He and Julian McAuley. 2016. VBPR: visual Bayesian Personalized Ranking from implicit feedback. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 144–150.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.

[14] 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*. International World Wide Web Conferences Steering Committee, 173–182.

[15] Anvitha Hegde and Savitha K Shetty. 2015. Collaborative filtering recommender system. *Int J Emerg Trends Sci Technol* 2, 07 (2015), 291–324.

[16] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*. JMLR. org, II–1512.

[17] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 843–852.

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[19] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 241–248.

[20] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 505–514.

[21] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*. JMLR. org, 448–456.

[22] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. 1997. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3 (1997), 77–87.

[23] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[25] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 345–354.

[26] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 811–820.

[27] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 305–314.

[28] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM conference on Recommender systems*. ACM, 5–12.

[29] Ruilin Pan, Tingsheng Yang, Jianhua Cao, Ke Lu, and Zhanchao Zhang. 2015. Missing data imputation by K nearest neighbours based on grey relational structure and mutual information. *Applied Intelligence* 43, 3 (2015), 614–632.

[30] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[31] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[32] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 127–136.

[33] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 6907–6917.

[34] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: addressing cold start in recommender systems. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 4964–4973.

[35] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. 2016. Collaborative filtering and deep learning based hybrid recommendation for cold start problem. In *Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*. IEEE, 874–877.

[36] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. 2017. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications* 69 (2017), 29–39.

[37] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1449–1458.

[38] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 425–434.