Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation

Shaoyun Shi, Min Zhang*, Yiqun Liu and Shaoping Ma
Department of Computer Science and Technology, Institute for Artificial Intelligence,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China
shisy17@mails.tsinghua.edu.cn, {z-m, yiqunliu, msp}@tsinghua.edu.cn

ABSTRACT

Nowadays, recommender systems provide essential web services on the Internet. There are mainly two categories of traditional recommendation algorithms: Content-Based (CB) and Collaborative Filtering (CF). CF methods make recommendations mainly according to the historical feedback information. They usually perform better when there is sufficient feedback information but less successful on new users and items, which is called the "cold-start" problem. However, CB methods help in this scenario because of using content information. To take both advantages of CF and CB, how to combine them is a challenging issue. To the best of our knowledge, little previous work has been done to solve the problem in one unified recommendation model. In this work, we study how to integrate CF and CB, which utilizes both types of information in model-level but not in result-level and makes recommendations adaptively. A novel attention-based model named Attentional Content&Collaborate Model (ACCM) is proposed. Attention mechanism helps adaptively adjust for each user-item pair from which source information the recommendation is made. Especially, a "cold sampling" learning strategy is designed to handle the cold-start problem. Experimental results on two benchmark datasets show that the ACCM performs better on both warm and cold tests compared to the state-of-the-art algorithms.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Attention Mechanism, Cold-Start, Neural Recommendation Model, Collaborative Filtering, Hybrid Recommendation, Cold Sampling

ACM Reference Format:
Shaoyun Shi, Min Zhang, Yiqun Liu and Shaoping Ma. 2018. Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271710

1 INTRODUCTION

With the continuous development of the Internet, the information explosion has become a great challenge that people are faced with [38]. Recommender systems are designed to help deal with this problem, which can help us get the information more efficiently. The two traditional recommendation techniques are Content-Based (CB) and Collaborative Filtering (CF). They are widely used, and many studies are based on them. They recommend according to two different kinds of information: content and historical feedback.

Content-based methods are based on the content information, which has mainly two types, i.e., user contents such as age, gender, occupation and item contents such as the color of clothes, taste of foods, melody of music. Besides, other specific information such as text reviews can be taken as both user and item content. They can give a brief impression of the user or the item even if there is no historical feedback information.

The main idea of CF is to analyze the historical feedback information of users and items. Two basic assumptions are that users who consume more similar items may have more similar preferences and items which are consumed by common users may have similar attributes or contents. The outstanding performance of CF methods is verified on many datasets, especially when the feedback information is rich.

From experiences [12, 15, 22, 35], no matter explicit or implicit, feedback information can help us achieve a better understanding of users' preferences than content information. Thus CF methods usually achieve better performance than CB methods. However, historical feedback information is not always available or sufficient. Especially, there is no previous feedback information of new users and items in the system, which is called cold-start. CF methods often fail in this situation. While CB methods suffer much less from the cold-start problem. CB methods can still recommend according to the brief impression drawn from the content, which is independent of the feedback information.

A promising way is to integrate advantages of CF and CB models in the recommendation. In many real applications, people learn to combine the results of CF and CB with different weights to make recommendations. In other cases, people first separate the scenario into warm or cold and use CF or CB respectively. In the former cases, the learned weights keep the same for different data, which makes the model less adaptive. In the latter cases, the model uses only content information or the feedback information, but actually, both of them are useful in the recommendation. Thus, hybrid in the model level has been paid more attention in recent

* Corresponding Author

years. The deep learning approach is one of the trails, which can embed content and feedback information into vectors [8, 14]. Deep neural networks can jointly take advantage of the two kinds of information in an easier way. They learn global weights of content and feedback information, which makes them not flexible enough for different users and items in different scenarios. Besides, it is hard to understand where a recommendation result comes from, no matter for a senior user with rich feedback information or a new user with no history.

What we want is a model that takes advantages of both content and feedback information. It should pay more attention to the content information when there is no historical information and leverage user feedback information when it is available. Hence, both warm and cold scenarios can be handled smoothly.

In this work, we proposed a novel deep recommendation model named Attentional Content&Collaborate Model (ACCM), which integrates both CB and CF uniformly. Unlike previous models, ACCM provides a more clear information flow with help us understand how much the content or the feedback information contributes to the final predictions, according to the attention weights. For each user-item pair, the weights for CB and CF are dynamically adjusted. And a novel learning strategy "cold sampling" is designed to "teach" the attention network what is cold-start. Experiments show that our ACCM significantly outperforms the state-of-the-art deep learning methods Wide&Deep [8], NFM [14] and AFM [42] on both cold and warm datasets.

The main contributions of this work are summarized as follows:

(1) We propose a new attention model to unify CF and CB recommendations for both warm and cold scenarios, in which both end-to-end features and traditional attributes are taken into consideration. To the best of our knowledge, it is the first work to select the information resource adaptively for each user-item pair in personalized recommendation.
(2) A novel "cold sampling" learning strategy is proposed which helps handle cold-start problem effectively. Noticeably, it is not a separate strategy but works in both warm and cold scenarios.
(3) Experimental results show it as a simple novel model which makes significant improvements to the state-of-the-art methods, especially in cold scenarios.

2 RELATED WORK

2.1 Traditional Recommendation Models

The two categories of traditional recommendation methods are Content-Based (CB) and Collaborative Filtering (CF).

No matter from explicit [23, 37] or implicit [16, 18, 34, 35] feedbacks, understanding the users and items from their historical feedback information is the basic idea of CF. One effective way to do CF is matrix factorization (MF), such as BiasedMF [23], WRMF [18], ExpoMF [26], SVDPP [22]. Given an interaction matrix $R = (R_{ij})_{m\times n} \in \Re_{+}^{m\times n}$ of m users and n items, the goal is to get user matrix $U_{m\times k}$ and item matrix $V_{k\times n}$. $R = UV$, where k is the dimension of the latent vector. Each row of U represents a user vector $\mathbf{u}_i$ and each column of V represents an item vector $\mathbf{v}_j$. We can use $\mathbf{u}_i$ and $\mathbf{v}_j$'s dot product as the rating prediction about how user i likes item j. But these methods can not deal with the cold-start problem, i.e., users or items with no historical feedback information.

CB methods make recommendation mainly based on users and items' content information. One of the typical CB methods is CBF [33]. The main idea is to average item features a user has interacted with to represent the user. A user feature vector can be formed to reflect what kind of items the user prefers. Then CBF recommends according to the similarities of the feature vectors of users and items. In this framework, when new items come, CBF still work. Another special source of content information is text reviews. A valuable review not only shows the attributes a specific user focus on but also indicates what the item is like. Many researchers are trying to take advantage of reviews in recent years. Models like HFT [29], RMR [27], EFM [46] have achieved excellent performance. CB methods are usually thought have the ability to handle the cold-start problem because the content information can help understand the users and items independent of historical feedback information.

The hybrid model is a promising way to combine CF and CB and use the advantages of CB to fix the disadvantages of CF, to work on both warm and cold scenarios [36]. Some works merge individual predictions of CF and CB methods into a single [3, 32]. They can work in the cold scenario but do not adaptively unify warm and cold starts recommendation into one model and need additional human efforts or knowledge to fit different scenarios. Other works add content information into a collaborative filtering model [1, 7, 31]. Although these models have the ability to handle the cold-start problem, they are parameter sensitive and have limitations on processing large data or high dimension features.

2.2 Deep Learning in Recommendation

Deep learning methods [24] have achieved remarkable results in many fields like computer vision [6, 13, 41] and natural language processing [2, 9, 28]. So many researchers are trying to introduce deep learning into recommender systems recently. Some strengthen the ability of traditional methods with deep learning. For example, Van den Oord et al. used CNN to extract latent vectors from music and achieved excellent performance [39]. MDA-CF [25] combines probabilistic matrix factorization with marginalized denoising stacked auto-encoders. Some works transferred traditional algorithms into deep learning framework. NCF [15] proposed by He et al. and CF-NADE [47, 48] proposed by Zheng et al. tried to do CF in a neural way. These methods achieved better results over many state-of-the-art methods.

Deep neural networks can provide an easier way to unify both content and historical feedback information to take advantage of both CF and CB. All the information can be embedded into vectors. For example, in the NFM [14], content information such as users and items features are embedded into different vectors and the same as their ids, which is end-to-end training by feedback information. It uses a bi-interaction layer to do the similar things as Factorization Machine (FM). Another example is Wide&Deep [8] proposed by Google, which combines the deep neural network and the linear model. The excellent performance of these methods is verified.

However, the weights in deep layers are fixed once the training process finished. Deep models can not adaptively adjust their weights in warm and cold scenarios which means when users or items are cold, the results are calculated from partly absent values. Although they can still work because content information contributes to the final results, they may give undesirable outputs because of the abnormal inputs. Besides, the explanation is also important for recommendation tasks. But the deep model is thought as the black box, which is hard for us to understand how they recommend.

2.3 Attention-Based Model

Attention mechanism has been shown effective in various machine learning tasks such as computer vision [5, 43] and natural language processing [17, 40]. It is a weighted sum technique but it can automatically analyze which parts, such as areas in images or words in sentences, are more important. For example, in neural machine translation, the attention mechanism can help decide which words in source sentence should be focused on when generating each target word. It makes neural networks more explainable and adaptive.

There are also trails to take advantages of attention mechanism in recommendation tasks. Miura et al. unified text, metadata, and user network representations with an attention mechanism in geolocation prediction to overcome the previous ensemble approaches [30]. Chen et al. introduced both component-level and item-level attention for multimedia recommendation into a CF framework [4]. Another example is AFM [42], which is an extension of NFM [14]. It improves FM by discriminating the importance of different feature interactions with the help of attention mechanism.

However, most of the attentional recommendation models need a list of historical information such as interacted items, linked cities or reviews of a user or an item. New items and users do not have such lists, making the models hard to recommend. Models like AFM can still give results in the cold scenario because the attention mechanism is applied to both content and historical feedback information. But they do not specifically handle the cold-start problem, which may affect the model performance because attention networks have not been trained to handle cold vectors and may give abnormal weights.

3 ATTENTIONAL COLLABORATE&CONTENT MODELS

Attention mechanism is a weighted sum technique. Assume there are a set of vectors $\{\mathbf{v}_i \in \mathbb{R}^k\}, i \in [1, n]$, where k is the vector size. We can form a vector $\mathbf{f}$ with attention as follows:

$$u_i = \mathbf{h}^T tanh(\mathbf{W}\mathbf{g}_i + \mathbf{b})$$
$$a_i = \frac{exp(u_i)}{\sum_i exp(u_i)} \tag{1}$$
$$\mathbf{f} = \textstyle\sum_i a_i \mathbf{g}_i$$

where $\mathbf{W} \in \mathbb{R}^{t \times k}$ and $\mathbf{b} \in \mathbb{R}^t$ are parameters of attention network and t is the attention size. The main idea is to dynamically adjust the weights by the network.

Here we want to use attention mechanism to decide where the recommendation should come from, the content or the historical feedback information. When feedback information is not sufficient,

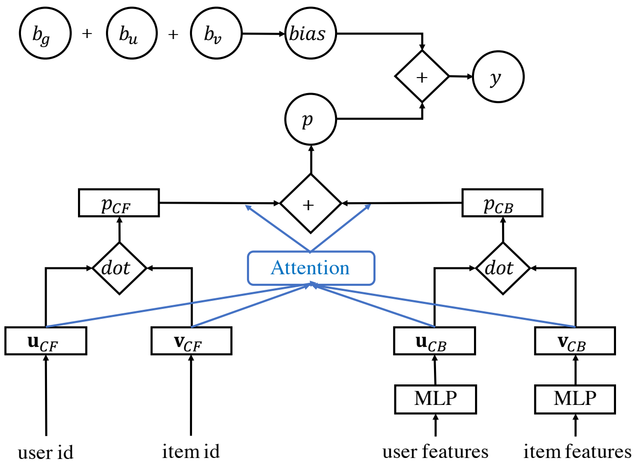

Figure 1: Model Structure of Result Level Attention

recommending according to the user profile and item attributes may be a better strategy.

3.1 Result Level Attention

A direct idea is using attention mechanism to combine CF and CB methods and give a weighted sum of two results, as shown in Figure 1. We can divide the prediction into two parts whose basic information is different.

The left part is a simple neural collaborative filtering framework in which each user u and item i is embedded as user vector $\mathbf{u}_{CF} \in \mathbb{R}^k$ and item vector $\mathbf{v}_{CF} \in \mathbb{R}^k$ directly from their ids, where k is the dimension of vectors. The embeddings can be learned during the training process from the historical feedback information. It can be seen as an end-to-end learning process. The input is which user and item, and the output is the preference prediction. The preference prediction of CF part comes from the dot product of $\mathbf{u}_{CF}$ and $\mathbf{v}_{CF}$:

$$p_{CF} = \mathbf{u}_{CF} \cdot \mathbf{v}_{CF} \tag{2}$$

Similarly, in the CB part, each user u and item i is embedded as user vector $\mathbf{u}_{CB} \in \mathbb{R}^k$ and item vector $\mathbf{v}_{CB} \in \mathbb{R}^k$. However, vectors here are calculated from the user and item feature embeddings, content information like user age, occupation or item color. It is not an end-to-end but feature-based learning. The preference prediction here keeps the same as CF part with dot product:

$$p_{CB} = \mathbf{u}_{CB} \cdot \mathbf{v}_{CB} \tag{3}$$

The model prediction p is a weighted sum of predictions from two parts, p_{CF} and p_{CB}. The attention network is responsible for giving the proper weight, which is the blue part in Figure 1. It can judge how reliable the predictions from two parts according to the information embedded in vectors. Ideally, not only warm vectors can be evaluated, but also cold vectors from CF part containing no feedback information can be identified. Formally, the attention network is defined as:

$$h_{CF} = \mathbf{h}^T tanh(\mathbf{W}(\mathbf{u_{CF}} \| \mathbf{v_{CF}}) + \mathbf{b})$$
$$h_{CB} = \mathbf{h}^T tanh(\mathbf{W}(\mathbf{u_{CB}} \| \mathbf{v_{CB}}) + \mathbf{b}) \tag{4}$$
$$a_{CF} = \frac{exp(h_{CF})}{exp(h_{CB}) + exp(h_{CF})} = 1 - a_{CB}$$

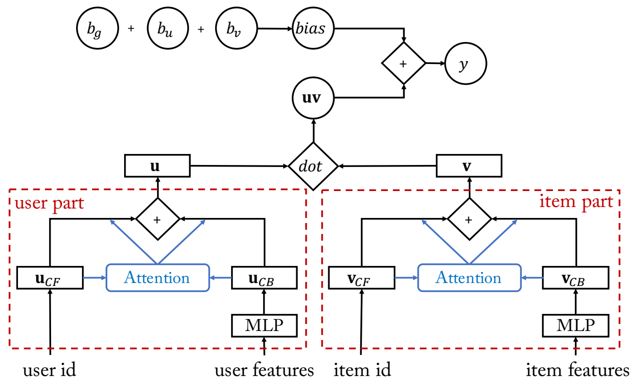

Figure 2: Model Structure of Vector Level Attention

where $\mathbf{W} \in \mathbb{R}^{t \times 2k}, \mathbf{b} \in \mathbb{R}^t, \mathbf{h} \in \mathbb{R}^t$ are model parameters, and t denotes the hidden layer size of the attention network. || here means vector concatenation. a_{CF} and a_{CB} are the output weights and the model prediction p is calculated as:

$$p = a_{CF} p_{CF} + a_{CB} p_{CB} \tag{5}$$

For rating prediction task, it is common to take bias into consideration, including global bias b_g, user bias b_u and item bias b_v. Then the predicted rating of item v given by user u is:

$$y = b_g + b_u + b_v + p \tag{6}$$

And the predicted ratings of two single parts, CF and CB, can be defined as:

$$\begin{aligned} y_{CF} &= b_g + b_u + b_v + p_{CF} \\ y_{CB} &= b_g + b_u + b_v + p_{CB} \end{aligned} \tag{7}$$

To encourage two single parts to give both meaningful predictions and avoid the situation that one part is much more stronger than the other part, RMSE of two single parts is added to the loss function. Otherwise, it may cause an unbalanced attention network giving larger weights to the same part all the time.

$$Loss = RMSE\{(y, r)\} + RMSE\{(y_{CF}, r)\} + RMSE\{(y_{CB}, r)\} \tag{8}$$

From the attention weights, we can clearly understand which part the prediction comes from, reflecting which kind of information the recommendation is based on. The structure is somehow like an ensemble learning method. But they are different because the weights given by attention network is dynamically adjusted for each user-item pair. Assume that there comes a new user, attention network may identify that he is new to the system and the recommendation should come from the CB part, which means $a_{CB} = 1$ and $a_{CF} = 0$. But in ensemble learning method, the weights of different parts will remain unchanged as long as the training process is finished. If the CF part performs better on the training data, it will get a higher weight $w_{CF} > 0$ in ensemble methods. Then no matter which user comes to the system, with sufficient or no feedback information, the model prediction comes from CF part on the weight of w_{CF}, which may bring unreliable results.

3.2 Vector Level Attention

A shortcoming of result level attention is that when a new user comes to the system, the model will predict according to the CB part even if the item contains rich historical feedback information. For example, if a specific user like puzzle games very much, much more than other users with the similar profile. Then if we want to know how the user prefers a newly published puzzle game, the model may give a medium rating according to the content information. What the model ignored is that the user loves puzzle games very much, which is embedded in the user vector $\mathbf{u}_{CF}$.

One way to solve this problem is to route the information flow in detail. We can decide how much should the user or item vector come from the content or historical feedback information. Attention mechanism can help when the user or item vectors are formed, as shown in Figure 2. In this structure, the attention weights in the user part only depend on the user information. User-specific historical feedback information can be reserved in user vector **u**. Considering the example we mentioned in the last paragraph, ideally, the attention can give suitable weight so that the user vector **u** contains the information that the user loves puzzle games very much and finally the model give a high rating to the newly published puzzle game. Formally, the attention here is similar to equation 4:

$$\begin{aligned} h^u_{CF} &= \mathbf{h}^T tanh(\mathbf{W}\mathbf{u}_{\mathbf{CF}} + \mathbf{b}), \\ h^u_{CB} &= \mathbf{h}^T tanh(\mathbf{W}\mathbf{u}_{\mathbf{CB}} + \mathbf{b}), \\ a^u_{CF} &= \frac{exp(h^u_{CF})}{exp(h^u_{CB}) + exp(h^u_{CF})} = 1 - a^u_{CB} \end{aligned} \tag{9}$$

where $\mathbf{W} \in \mathbb{R}^{t \times k}, \mathbf{b} \in \mathbb{R}^t, \mathbf{h} \in \mathbb{R}^t$ are model parameters, and t denotes the hidden layer size of the attention network. Then the user vector is a weighted sum:

$$\mathbf{u} = a^u_{CF}\mathbf{u}_{CF} + a^u_{CB}\mathbf{u}_{CB} \tag{10}$$

The item vector **v** is generated similarly to user vector **u**. The attention network in item part shares the same parameters with that in the user part. We have two reasons: the dot product makes that the same dimension of the user and the item vector should be a hidden factor with the same physical meaning; the purposes of attention mechanism in two parts are the same that to judge whether the vector coming from content or feedback information is reliable.

The model prediction is a sum of bias and the dot product of **u** and **v**:

$$y = b_g + b_u + b_v + \mathbf{uv} \tag{11}$$

The model is more general than the result level attention. This model indicates explicitly whether the prediction comes from content or feedback information or more specifically, how the model understands the user or the item from the historical feedback or their content. The vectors coming from the user or item features are content-based. The learning process of user and item id embeddings is end-to-end collaborative filtering. They are combined by attention mechanism. So we named the model Attentional Content&Collaborate Model (ACCM).

3.3 Model Learning: Attention on Cold users and items

In general cases, historical information on user-item interactions is used to train the model, hence the user's feedback information plays important role on reflecting his preference, usually even more than his profile information does. While what is more common in real scenarios, there are always some cold users or items, which have never (or rarely) been observed in past known user-item interactions. In such cold start cases, CF part does not work properly since no information of this new user/item is provided. Since all information is known to the model in the traditional training process, during test procedure, new user's vector from the CF part remains the initial state and is usually random at a distribution different from trained vectors. In this situation, the output of attention network will be unreliable.

In this section, we discuss and propose several strategies to solve the learning problem in cold start scenarios. The basic idea is how to introduce cold users and items in training procedure and hence help the model learn the attention weights in such real cases.

3.3.1 Attention Set Strategy. One intuitive way is to lay off a set of data, at a ratio α, from the training data, which contains both cold and warm users and items. We call it *"attention set"*, use it to train the attention network only. The training process alternate between two steps:

- *model update*: Keep the attention network unchangeable and use the remaining part of the training set to update the main stream of the model, which is the black part in Figure 2.
- *attention update*: Use the laying off "attention set" to train the attention network, which is the blue part in Figure 2. To keep the cold data remaining cold, the other part of the model should not be updated during this step.

Note that historical feedback information in the laying off "attention Set" is not used to train the id embeddings. When we want to make recommendations about these users or items, their feedback information will not contribute to the results due to this hard split, which is undesirable.

3.3.2 Cold Sampling Strategy. We propose a new alternative strategy to overcome the above problem in *"Attention Set Strategy"*. We name it *"cold sampling"*, which randomly shadows the historical feedback information of some users and items in each batch. The

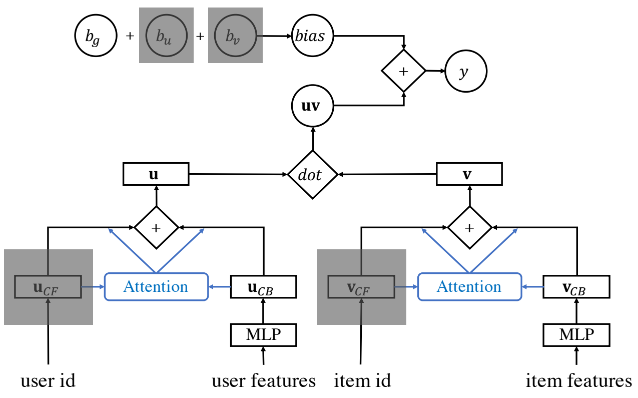

Figure 3: Learning of ACCM in Cold Scenarios

```
input      : training set $T = \{(u, v, r)\}$; learning rate $lr$;
             batch size $n$; sample ratio $\beta$
parameter  : cold signal for the user and the item $c_i^u, c_i^v$;
             user CF vectors$\{\mathbf{u}_{CF}\}$; item CF vectors$\{\mathbf{v}_{CF}\}$;
             global bias $g_b$; user bias $g_u$; item bias $g_v$;
             other feature embeddings, parameters
for epoch do
    $T \leftarrow$ shuffle($T$);
    for batch $B = \{(u_i, v_i, r_i)\}, i \in [1, n]$ do
        generate $c_i^u, c_i^v \in \{0, 1\} \sim Bernoulli(\beta)$ independently
          (0 denotes cold and 1 is not);
        for user $u_i$ do
            generate user random vector $\{\mathbf{u}_r^i\}$;
            calculate user CF and CB vector $\mathbf{u}_{CF}^i, \mathbf{u}_{CB}^i$;
            calculate attention weight $a_{CF}^{u_i}, a_{CB}^{u_i}$ with
              $(1 - c_i^u)\mathbf{u}_{CF}^i + c_i^u \mathbf{u}_r^i$ and $\mathbf{u}_{CB}^i$ according to
              equation 9;
            calculate user vector $\mathbf{u}_i$ according to equation 12;
        end
        foreach item $v_i$ do calculate item vector $\mathbf{v}_i$ similarly
          as above calculation on $\mathbf{u}_i$;
        foreach $i$ do $y_i \leftarrow b_g + c_i^u b_{u_i} + c_i^v b_{v_i} + \mathbf{u}_i \mathbf{v}_i$;
        Calculate the Loss according to equation 8;
        foreach $c_i^u == 1$ do update $\mathbf{u}_{CF}^i$ with Adagrad(lr,Loss);
        foreach $c_i^v == 1$ do update $\mathbf{v}_{CF}^i$ with Adagrad(lr,Loss);
        update other parameters with Adagrad(lr,Loss);
    end
end
```

Algorithm 1: Learning Process of "Cold Sampling"

process is concluded as Algorithm 1. The model is trained with the common mini-batches. In each batch with a size of n, some users and items are randomly set to be cold, i.e. their historical interaction information is shadowed as unknown. Let $c_i^u, c_i^v \in \{0, 1\}$ denote whether the feedback information of the user and the item with index $i \in [1, n]$ in the batch is shadowed, respectively. c_i^u and c_i^v are sampled independently in $Bernoulli(\beta)$.

Then the prediction during training process is:

$$\mathbf{u}_i = a_{CF}^{u_i}[(1 - c_i^u)\mathbf{u}_{CF}^i + c_i^u \mathbf{u}_r^i] + a_{CB}^{u_i}\mathbf{u}_{CB}^i$$
$$\mathbf{v}_i = a_{CF}^{v_i}[(1 - c_i^v)\mathbf{v}_{CF}^i + c_i^v \mathbf{v}_r^i] + a_{CB}^{v_i}\mathbf{v}_{CB}^i \qquad (12)$$
$$y_i = b_g + c_i^u b_{u_i} + c_i^v b_{v_i} + \mathbf{u}_i \mathbf{v}_i$$

where r_i is the label, $\mathbf{u}_r^i$ and $\mathbf{v}_r^i$ are randomly generated vectors in $\mathbb{R}^k$ in the same distribution as the initial user and item vectors. The attention weights of CF part $a_{CF}^{u_i}$ and $a_{CF}^{v_i}$ is calculated according to the $\mathbf{u}_r^i$ and $\mathbf{v}_r^i$ if u_i or v_i is chosen to be shadowed as cold one. Note that the bias will also be shadowed in order to simulate a real cold situation. The model structure with historical feedback information shadowed is shown in Figure 3.

Since users and items be shadowed as cold ones are randomly selected in each batch and epoch, overall, the model tends to use all information available. This *Cold Sampling Strategy* takes an unified learning process and is easy to be implemented.

4 EXPERIMENTS

To verify the model performance, we conduct experiments mainly to answer the following questions:

RQ1 How does ACCM perform compared to the state-of-the-art methods?

RQ2 How do the key hyper-parameters of ACCM (i.e., attention size, embedding size) influence the performance?

RQ3 Is attention mechanism helpful to adjust source information of recommendation? Can it recognize cold items or users while predicting?

RQ4 Does the specific "cold sampling" learning strategy enhance the model ability to make recommendations on cold data?

4.1 Experimental Settings

4.1.1 Datasets. We mainly conducted our experiments on two datasets: ML-100k and WeiboDouban. Some detailed information of the datasets is shown in Table 1

Table 1: Statistics of Evaluation Datasets

Dataset	Interaction#	User#	Item#	Sparsity
ML-100k	100,000	943	1,682	93.70%
WeiboDouban	354,929	5,796	14,468	99.58%

• **ML-100k**. It is a stable benchmark dataset maintained by Grouplens . It includes 943 users and 1,682 movies. Historical feedback information is 100,000 ratings in 1 to 5 given by a user to a movie. Content information we used is the age, gender, occupation of users and release year, genres of items.

• **WeiboDouban**. The data has been used by many previous works [20, 44, 45]. The dataset contains the footprints of users mainly from two domains: Douban and SinaWeibo . We use the ratings of books as the historical feedback information. We sampled 5,796 users and 14,468 books with non-empty tags. Tags are words which can give a brief capture of users and books. Users may tag themselves in SinaWeibo and tag books in Douban. We choose the top 100 most frequent tags of users as their content information, and the books are similar. 1 and 0 mean whether the user or the book has this tag or not. The WeiboDouban dataset is much sparse than the ml-100k, which provide verification of model performance with different sparsity.

4.1.2 Evaluation. To form a warm evaluation first, we remain one sample for each user and item in the training data, and then randomly split the other samples. We split the dataset into training (80%), validation (10%) and test (10%) sets. To form evaluation sets in different cold ratios, for example, 50% item cold, we randomly choose 50% samples in the validation and test sets and give each sample a specific item id only for the sample. The validation set is used to conduct the early stop and optimize hyper-parameters, and we use the test set to evaluate the model performance. The evaluation metric is Root Mean Square Error (RMSE).

4.1.3 Baselines and Variations of Our Model. We compare our proposed ACCM with the following state-of-the-art algorithms. Each algorithm has been trained and optimized on each dataset.

• **UserKNN** [21]. It is a CF method based on the k-nearest neighbors (KNN) algorithm. To predict the rating of an item given by a user, it finds the most similar users who have rated the item and output a weighted average of their ratings.

• **ItemKNN** [10]. It is similar to UserKNN but applies KNN on item level.

• **BiasedMF** [23]. It is one of the matrix factorization models of CF. Similar to ACCM, it uses dot product of user and item vectors with biases as the prediction. It makes recommendations only based on the historical feedback information.

• **SVD++** [22]. A well-known CF method which integrates both explicit and implicit feedback, which has been shown powerful in personalized recommendations in many previous researches.

• **UserItemCB**. It can be regarded as the right CB part of result level attention model in Figure 1 with biases. It makes recommendations mainly based on content information of users and items.

• **NCF** [15]. It is Neural network-based Collaborative Filtering proposed by He et al. in 2017. It performs well by doing collaborative filtering in a framework of neural networks.

• **NFM** [14]. It is a Neural Factorization Machine proposed by He and Chua in 2017. It is one of the state-of-the-art deep learning methods, which uses Bi-Interaction Layer to integrate both content and historical feedback information.

• **AFM** [42]. Attentional Factorization Machine, it is an extension of NFM, which uses attention network to achieve a weighted sum of feature interactions.

• **Wide&Deep** [8]. It is proposed by Google in 2016, which combines the deep neural network and linear model. One-hot vectors are directly fed into the wide linear part and embedded in the deep neural part.

To better understand the model structure, experiments were also conducted on some variation of ACCM.

• **RLAM**. It is Result Level Attention Model, mentioned in 3.1.

• **RLWS**. It is Result Level Weighted-Sum Model, which replaces the attention network in RLAM with weights as a weighted-sum of CF and CB results.

• **VLWS**. It is Vector Level Weighted-Sum Model, which replaces the attention network in ACCM with weights as a weighted-sum of CF and CB vectors.

RLWS and VLWS models are used to show the effectiveness of attention parts in the proposed ACCM. For RLAM and ACCM, experiments of different training strategies are also conducted, including regular training and using "attention set" or "cold sampling" mentioned in 3.3. We provide the codes for ACCM in Github at https://github.com/THUIR/ACCM.

4.1.4 Parameter Setting. Generally, all the methods are optimized with mini-batch Adagrad [11], in which the learning rate can adaptively slow down as the learning process going on. Learning rate is searched for each model from 0.001 to 0.1. Batch size is set to 128. The early stop is conducted according to the performance on the

https://grouplens.org/datasets/movielens/100k/
https://www.douban.com/
https://www.weibo.com/

Table 2: Overall Performance

	ML-100k		WeiboDouban	
	Random	Cold[1]	Random	Cold[1]
UserKNN	0.9376	1.0498	1.5550	1.7511
ItemKNN	0.9237	1.0431	1.4622	1.7125
BiasedMF	0.9375	1.0186	1.4278	1.5917
SVD++	0.9220	1.0151	1.4327	1.5960
UserItemCB	0.9370	*0.9931*	1.4225	1.5750
NCF	0.9450	1.1696	1.4818	1.8382
NFM	0.9143	0.9974	1.4072	1.5818
AFM[2]	0.9274	0.9934	-	-
Wide&Deep	*0.9099*	0.9966	*1.4064*	*1.5680*
RLWS	0.9133	1.0567	1.4158	1.7215
RLAM (Regular Train)	0.9132	1.1574	1.4053	1.6357
RLAM (Attention Set)	0.9123	1.0150	1.4063	1.5808
RLAM (Cold Sampling)	0.9134	0.9951	1.4084	1.5753
VLWS	0.9110	1.0608	1.4101	1.6573
ACCM (Regular Train)	**0.9006**[*3]	1.0176	**1.4048**[3]	1.5896
ACCM (Attention Set)	**0.9012**[*3]	0.9906	**1.4027**[3]	1.5812
ACCM (Cold Sampling)	**0.9027**[*3]	**0.9776**[*]	**1.3987**[*3]	**1.5541**[*]

1. Randomly item 30% cold and user 30% cold
2. Experiments of AFM on WeiboDouban cannot finish in acceptable time and computing resources
3. The bold values in the same column are not significantly different among themselves ($p > 0.01$)
*. Significantly better than the best baseline (italic ones) with $p < 0.01$

validation set. Embedding size (number of latent factors) is 64 in ML-100k and 32 in WeiboDouban. Specifically, there is no hidden layer in the AFM, UserItemCB or ACCM's CB part and one hidden layer with the size of 64 in NFM and Wide&Deep. All these parameters are well optimized by our efforts and near the best. To prevent the model from overfitting, we employ dropout on the attention network and fully-connected layers in the CB parts and the dropout ratio is set between 0.05 to 0.2. Batch Normalization (BN) [19] is conducted on the fully-connected layers.

4.2 Performance Comparison (RQ1)

4.2.1 Overall Performance. Table 2 shows the overall performance of ACCM and other baselines. We also show the performance of result level attention model (RLAM) as a comparison. Models are evaluated on two cold scenarios: randomly split data; independently random 30% of users and 30% of items to be cold. Note that experiments of AFM on WeiboDouban cannot finish in acceptable time and computing resources due to hundreds of tag features. So AFM could only be tested on ML-100k. From the results, following observations can be made:

(1) Methods taking advantages of both content and feedback information like ACCM, RLAM, Wide&Deep, NFM are generally better than methods using only one of them. It is reasonable because more information can help us know users better.

(2) On warm scenarios, using attention mechanism to adjust the recommendation source, content or historical feedback information, ACCM outperforms state-of-the-art approaches including

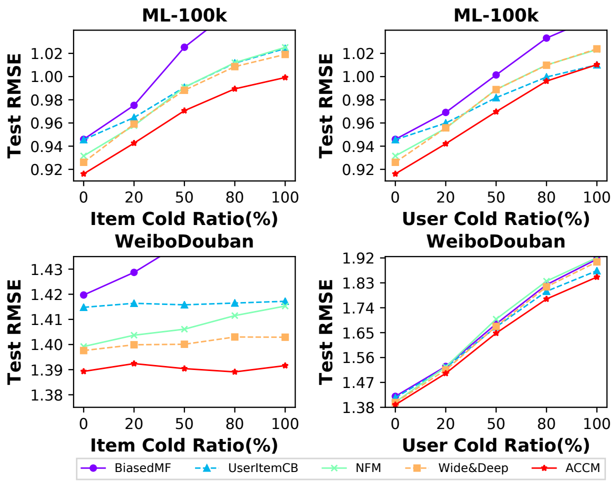

Figure 4: Performance in Different Scenarios. Note that the RMSE of BiasedMF keeps growing up to a high value in some scenarios. To have a clear comparison between other methods, lines of BiasedMF is not drawn entirely in those diagrams.

Wide&Deep, NFM and other baselines. The reason is that the richness of content and feedback information are different among users and items. Other methods use global weights, which may ignore this and lose some ability of adaptation in different scenarios.

(3) On cold scenarios, using attention and training in a regular way is not enough to handle the cold-start problem. Because samples in the training set are all warm and attention networks have not been taught how to handle cold items or users. Either cold training strategy, "attention set" or "cold sampling", improves the performance.

(4) To handle the cold-start problem, "cold sampling" is better because "attention set" throw out some historical feedback information and do not provide a unified learning process as "cold sampling", as mentioned in 3.3.

(5) In most situations, vector level attention (ACCM) is better than result level attention (RLAM). It is reasonable because vector level attention is more flexible and general than result level attention. It can not only identify cold user-item pairs but also differentiate cold and warm starts of each user or item.

4.2.2 Impacts of Different Cold Ratios. We also test the models' performance on datasets suffering from different ratios of cold-start, shown in Figure 4. It is clear that our ACCM perform better than other baselines on all conditions. Note that the performance of UserItemCB is getting worse at a slowest speed because it is a CB method. ACCM is no worse than UserItemCB in all scenarios. NFM and Wide&Deep perform worse than UserItemCB in the extremely user cold situations. The reason we think is that historical feedback information of users plays an important role in understanding the user, thus NFM and Wide&Deep may give a bigger global weight to user embeddings. But in extremely cold scenarios the embeddings of user ids contain no information and have a harmful impact on the model outputs.

Wide&Deep is the best baseline in most scenarios. So we mainly compare ACCM with Wide&Deep in the following experiments.

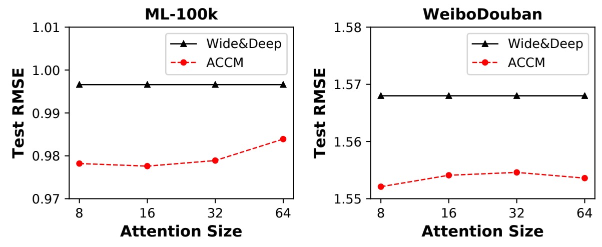

Figure 5: Impact of Attention Size. Models are evaluated on test sets in which randomly 30% of samples are item cold, and 30% of samples are user cold.

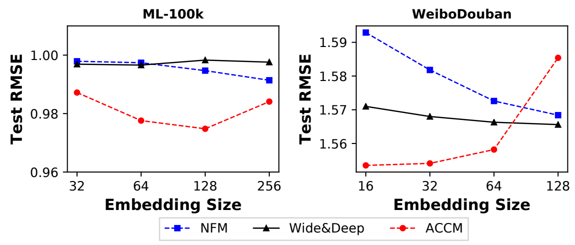

Figure 6: Impact of Embedding Size. Models are evaluated on test sets in which randomly 30% of samples are item cold and 30% of samples are user cold.

4.3 Hyper-Parameter Investigation (RQ2)

There are mainly two hyper-parameters in ACCM: attention size, embedding size.

4.3.1 Attention Size. Attention size may affect the ability of the model to evaluate the source information and choose proper recommendation strategy. To investigate the impact of attention size, we test the model performance when the attention size is 8, 16, 32, 64, shown in Figure 5. Results show that ACCM is robust to the changes of attention size. This may be due to that it is enough to measure two vectors from CF and CB parts with a small size of attention. But an over size of attention may lead to overfitting and cause a drop of model performance. We choose 16 to be the attention size for other experiments.

4.3.2 Embedding Size. Another hyper-parameter we focus on is the embedding size. Small embedding size may not be enough to represent the user and item features. Larger embedding size can enhance the model's ability to describe data. But an over size of embedding may cause overfitting and significantly affect the learning efficiency. Figure 6 shows the performance of models with different embedding sizes. In most situations, ACCM performs better with the same embedding size as NFM and Wide&Deep, except on the WeiboDouban when the embedding size is larger than 128. The reason is that our ACCM starts to overfit at a smaller embedding size than Wide&Deep. But ACCM performs well with small embedding size, better than NFM and Wide&Deep with much larger embedding size, which shows the effectiveness of our model.

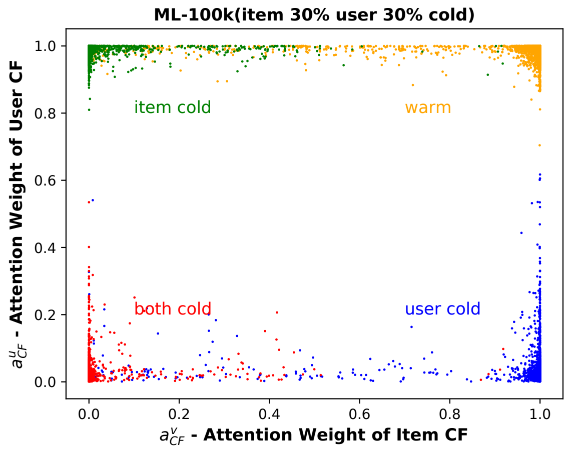

Figure 7: Visualization of Attention Weights. Samples are from the test set of ML-100k in which randomly 30% of samples are item cold, and 30% of samples are user cold. a^v_{CF} is the attention weight of item CF vector, and a^u_{CF} is the attention weight of user CF vector.

4.4 Recommendation Traceback with Attention (RQ3)

The attention weights reflect how the model recommend and which information the recommendation is based on. Ideally, for those samples suffer from the cold-start problem, attention mechanism should give higher weights to content information. To have a look at whether the attention network can identify cold items and users, we let the attention weights of the user and item CF vector, i.e., a^u_{CF} and a^v_{CF}, be the two features of a user-item pair. We draw a scatter diagram of samples according to their attention weights, as shown in Figure 7. The samples are from the test set of ML-100k in which 30% items and 30% users are cold.

Each point is a user-item pair and different colors represent different cold scenarios. Results show that ACCM can not only clearly identify which is a cold sample, but also distinguish the item cold and user cold. For example, when the sample is from a warm user and a cold item, which is among the green dots on the top left, the weight of CF vector in item part a^v_{CF} is near 0 because it is better to understand the items by its content information, but the weight of CF vector in user part a^u_{CF} is near 1 because historical feedback information is a better reflection of the users' preference. From the two dimensions of attention weights, four situations, i.e., item cold, user cold, both cold, warm, can be clearly differentiated.

Let us take some examples to show how the attention weights help us trace back the recommendation. Table 3 shows some samples in different cold scenarios from ML-100k. The first three (EX1-EX3) are all warm user-item pairs. Their attention weights of CF part are all bigger than 0, which means that there is some historical feedback information can be taken advantage of. The last three (EX4-EX7) are cold user-item pair. No matter the user or the item, if it is cold, not concluded in the training set, the attention weights of the CF vectors will be around 0. Note that the attention weights can also somehow reflect the richness of feedback information. The EX4 is an example that the item shows only once in the training set, whose attention weight of item CF is still near 0. And EX2 and

Table 3: Examples of Attention Weights on ML-100k

		# in Training		Attention		Rating	
	User	Item	a^u_{CF}	a^v_{CF}	y	r	
Warm	EX1	102	165	0.998	0.998	5.13	5
	EX2	25	121	0.746	1.000	3.799	4
	EX3	358	10	0.996	0.344	2.838	3
Cold	EX4	157	1	0.998	0.000	4.015	4
	EX5	0	52	0.024	0.985	2.385	2
	EX6	183	0	0.993	0.015	1.120	1
	EX7	0	0	0.015	0.000	4.059	4

EX3 show the historical feedback information is not rich enough to absolutely dominate the recommendation.

Generally, there are two advantages of taking this kind of attention framework:

(1) Instead of manual efforts to adjust recommendation strategies, the model can automatically select appropriate recommendation strategies in different cold scenarios and take the best advantages of content and historical feedback information.
(2) The attention weights make the recommendation strategies understandable by us. We can trace back what information dominates the prediction.

4.5 Impact of Cold Training (RQ4)

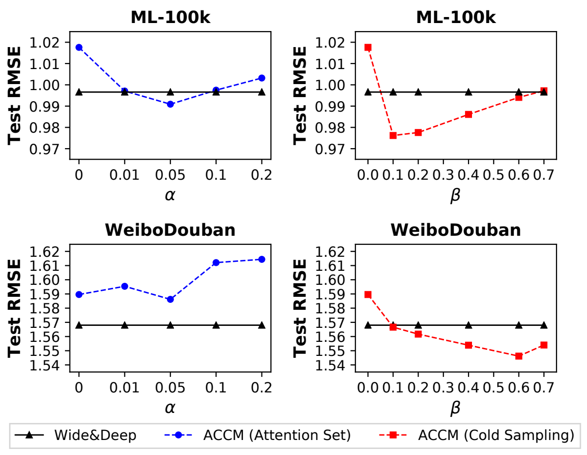

Figure 8: Impact of Cold Training. Models are evaluated on test sets in which randomly 30% of samples are item cold and 30% of samples are user cold. α is the size of "attention set", a relative ratio of the whole training set. β is the ratio of cold sampling in each batch.

As mentioned in 3.3, in ACCM, we conduct a special training strategy "cold sampling" to help handle the cold start problem. We first focus on whether "cold sampling" can help deal with the cold data. We changed the ratio β of vectors shadowed in each batch. The models are evaluated on test sets in which randomly 30% of samples are item cold and 30% of samples are user cold. The results are shown in Figure 8.

As a comparison, results of Wide&Deep are the black lines, which are trained in regular batches. Without "cold sampling", the performance of ACCM (red lines when $\beta = 0.0$) is worse than Wide&Deep. This is because in ACCM the vectors of cold items are remained untrained and usually are random mass. Attention networks have not been trained to handle such situations and may give random weights too, which directly affects the final prediction. Wide&Deep learns global weights of those features and has hidden layers to extract information from other content features. Fully connected hidden layers dilute the influence of cold embeddings. But the absent information still have weights and bad impact on the final result. Fortunately, a small number of cold samples can significantly improve the ACCM performance on cold data, i.e. red lines when $\beta \geq 0.1$. The reason is that a few cold samples are enough to let the model or the attention network know what is "cold" and what to do in such a situation. But the number of cold samples should not be too large, as it may affect the efficiency of the learning process.

We also test the performance of different sizes of "attention set", which are the blue lines in the figures. It uses part of the training set to specifically train the attention network. We show the performance changes as the size of "attention set" changing. α is the relative size to the whole training set. This training strategy is verified worse than "cold sampling". There are mainly two reasons: laying off an "attention set" will throw out much historical feedback information, which should be used to train the user and item embeddings; two step training is not unified and may impair the coordination of learning process.

For other experiments in previous sections, we set the β 0.1 to 0.4 in ML-100k and 0.3 to 0.6 in WeiboDouban. And the relative size of "attention set" α is 1% to 5% of the training set.

5 CONCLUSIONS

In this work, we proposed a novel model called Attentional Content&Collaborate Model (ACCM) to unify both content and historical feedback information in the recommendation. In the model, CB and CF information are combined in vector-level but not result-level. It uses the attention mechanism to control the ratio of the two types of information for each user-item pair when making recommendations. ACCM can automatically choose proper information to represent the user and the item. Adaptive learning strategies "cold sampling" are conducted to handle the cold-start problem. ACCM finally achieves better performance on both cold and warm data than the state-of-the-art methods like NFM, AFM and Wide&Deep. It reveals a new direction to handle both cold and warm starts in the same recommendation model adaptively and effectively with attention mechanism.

We notice that "cold sampling" is not a training strategy specific for attention set. In the future, we would like to introduce the "cold sampling" strategy on more models, including state-of-the-art approaches. Besides, more information such as social information or reviews can also be integrated, which can be considered as context information different from both content and historical feedback.

ACKNOWLEDGMENTS

This work is supported by Natural Science Foundation of China (Grant No. 61672311, 61532011), and is partly supported by the Tsinghua-Sogou Tiangong Institute for Intelligent Computing.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Bo Long. 2011. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 609–617.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Daniel Billsus and Michael J Pazzani. 2000. User modeling for adaptive news access. *User modeling and user-adapted interaction* 10, 2-3 (2000), 147–180.

[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 335–344.

[5] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, and Tat-Seng Chua. 2016. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. *arXiv preprint arXiv:1611.05594* (2016).

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

[7] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, Dec (2012), 3619–3622.

[8] 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.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

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

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[12] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings.. In *Aaai*. 123–129.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. (2017).

[15] 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.

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 549–558.

[17] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. 1693–1701.

[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[19] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[20] Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. 2016. Little Is Much: Bridging Cross-Platform Behaviors through Overlapped Crowds.. In *AAAI*. 13–19.

[21] 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.

[22] 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.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[25] 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.

[26] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 951–961.

[27] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 105–112.

[28] Christopher Manning, Richard Socher, Guillaume Genthial Fang, and Rohit Mundra. 2017. CS224n: Natural Language Processing with Deep Learning1. (2017).

[29] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[30] Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2017. Unifying text, metadata, and user network representations with a neural network for geolocation prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1260–1272.

[31] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 155–162.

[32] Michael J Pazzani. 1999. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review* 13, 5-6 (1999), 393–408.

[33] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[34] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 273–282.

[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[36] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. 2015. *Recommender systems handbook*. Springer.

[37] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

[38] Latanya Sweeney. 2001. Information explosion. *Confidentiality, disclosure, and data access: Theory and practical applications for statistical agencies* (2001), 43–74.

[39] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in neural information processing systems*. 2643–2651.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762* (2017).

[41] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3156–3164.

[42] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[43] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4651–4659.

[44] Nicholas Jing Yuan, Fuzheng Zhang, Defu Lian, Kai Zheng, Siyu Yu, and Xing Xie. 2013. We know how you live: exploring the spectrum of urban lifestyles. In *Proceedings of the first ACM conference on Online social networks*. ACM, 3–14.

[45] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, and Xing Xie. 2014. Mining novelty-seeking trait across heterogeneous domains. In *Proceedings of the 23rd international conference on World wide web*. ACM, 373–384.

[46] 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 the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 83–92.

[47] Yin Zheng, Cailiang Liu, Bangsheng Tang, and Hanning Zhou. 2016. Neural Autoregressive Collaborative Filtering for Implicit Feedback. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2–6.

[48] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477* (2016).