# Multi-Interest Network with Dynamic Routing for Recommendation at Tmall

Chao Li, Zhiyuan Liu*
Mengmeng Wu, Yuchi Xu
Huan Zhao, Pipei Huang*
{ruide.lc,michong.lzy,max.wmm,yuchi.xyc,chuanfei.zh}@alibaba-inc.com
pipei.hpp@gmail.com
Alibaba Group, Beijing, China

Guoliang Kang
Guoliang.Kang@student.uts.edu.au
Centre for Artificial Intelligence
University of Technology Sydney
Sydney, Australia

Qiwei Chen
Wei Li
{chenqiwei.cqw,rob.lw}@alibaba-inc.com
Alibaba Group, Hangzhou, China

Dik Lun Lee
dlee@cse.ust.hk
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Kowloon, Hong Kong

## ABSTRACT

Industrial recommender systems have embraced deep learning algorithms for building intelligent systems to make accurate recommendations. At its core, deep learning offers powerful ability for learning representations from data, especially for user and item representations. Existing deep learning-based models usually represent a user by one representation vector, which is usually insufficient to capture diverse interests for large-scale users in practice. In this paper, we approach the learning of user representations from a different view, by representing a user with multiple representation vectors encoding the different aspects of the user's interests. To this end, we propose the Multi-Interest Network with Dynamic routing (MIND) for learning user representations in recommender systems. Specifically, we design a multi-interest extractor layer based on the recently proposed dynamic routing mechanism, which is applicable for modeling and extracting diverse interests from user's behaviors. Furthermore, a technique named label-aware attention is proposed to help the learning process of user representations. Through extensive experiments on several public benchmarks and one large-scale industrial dataset from Tmall, we demonstrate that MIND can achieve superior performance than state-of-the-art methods in terms of recommendation accuracy. Currently, MIND has been deployed for handling major online traffic at the homepage on Mobile Tmall App.

## CCS CONCEPTS

• **Information systems → Recommender systems**.

*Chao Li and Zhiyuan Liu contribute equally to this research. Pipei Huang is the Corresponding author.


*CIKM '19, November 3–7, 2019, Beijing, China*

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357814

## KEYWORDS

Recommendation system; Deep learning; User representation; Dynamic Routing; Capsule Networks

**ACM Reference Format:**
Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357814

## 1 INTRODUCTION


**Figure 1: Left: The areas highlighted with dashed rectangle are personalized for billion-scale users at Tmall; Right: User A interacts with products from several different categories, including clothes, sports and food, while user B interacts with products of books, toys and cellphones.**

Tmall, the biggest Business-To-Customer (B2C) e-commerce platform in China, serves billion-scale users by providing billion-scale products online. On November 11-th of 2018, the well-known Tmall global shopping festival, the Gross Merchandise Volume (GMV) is around 213 billion yuan, achieving an increase rate of 26.9% compared with the same day of 2017. As the number of users and products is continuously growing, it becomes increasingly important to help each user find products that he/she might be interested in. In recent years, Tmall has spent huge efforts in developing personalized recommender systems (RS for short), which significantly contribute to the optimization of user experience and the increase of business value. For example, the homepage on Mobile Tmall

App (as shown in Figure 1 (Left)), which accounts for about half of total traffic at Tmall, has deployed RS for displaying personalized products to meet customers' personalized need.

Due to the billion-scale users and items, the recommendation process designed for Tmall consists of two stages, the matching stage and the ranking stage. The matching stage is responsible for retrieving thousands of candidate items that are relevant to user interests, after which the ranking stage predicts precise probabilities of users interacting with these candidate items. For both of the two stages, it is vital to model user interests and find user representations capturing user interests, in order to support efficient retrieval of items that satisfy users' interests. However, it is non-trivial to model user interests at Tmall, due to the existence of diverse interests of users. On average, billion-scale users visit Tmall, each user interacts with hundreds of products every day. The interacted products tend to belong to different categories, indicating the diversity of user interests. For example, as shown in Figure 1 (Right), different users are distinct in terms of their interests and the same user may also be interested in various kinds of items. Therefore, the capability of capturing user's diverse interests becomes vital for RS at Tmall.

Existing recommendation algorithms model and represent user interests in different ways. Collaborative filtering-based methods represent user interests by historical interacted items [21] or hidden factors [15], which suffer from sparsity problem or computationally demanding. Deep learning-based methods usually represent user interests with low-dimensional embedding vectors. For example, the deep neural network proposed for YouTube video recommendation (YouTube DNN) [5] represents each user by one fixed-length vector transformed from the past behaviors of users, which can be a bottleneck for modeling diverse interests, as its dimension must be large in order to express the huge number of interest profiles at Tmall. Deep Interest Network (DIN)[30] makes the user representation vary over different items with attention mechanisms to capture the diversity of user interests. Nevertheless, the adoption of attention mechanisms also makes it computationally prohibitive for large-scale applications with billion-scale items as it requires re-calculation of user representation for each item, making DIN only applicable for the ranking stage. Therefore, for the matching stage, one has to increase the dimension of both user and item representations in order to model diverse interests of billion-scale users if only one user representation vector is used. However, as the numbers of users and items are billion-scale, increasing dimension will induce huge costs for both computation and storage. Moreover, the increase of parameter size would make it difficult for model optimization. So, what we seek is a proper way to increase model capability without much additional cost.

Instead of using only one user representation vector, in this paper, we propose to use multiple representation vectors to model billion-scale users' diverse interests. Through this way, we have significantly increased model capability as well as recommendation accuracy, while no much additional cost is needed. Specifically, we propose the Multi-Interest Network with Dynamic routing (MIND) for learning multiple representation vectors for each user. Inspired by the dynamic routing algorithm [20], we design a novel layer called multi-interest extractor layer, and this layer can discover different aspects of interests and generate representations for diverse interests. Then, several feed-forward layers are stacked to transform these interest representation vectors into user representation vectors. These user representation vectors are computed only once and can be used in the matching stage for retrieving relevant items from billion-scale items. To summarize, the main contributions of this work are as follows:

- To capture diverse interests of users from user behaviors, we design the multi-interest extractor layer, which utilizes dynamic routing to adaptively aggregate user's historical behaviors into user representation vectors.
- By using user representation vectors produced by the multi-interest extractor layer and a newly proposed label-aware attention layer, we build a deep neural network for personalized recommendation tasks. Compared with existing methods, MIND shows superior performance on several public datasets and one industrial dataset from Tmall.
- To deploy MIND for serving billion-scale users at Tmall, we construct a system to implement the whole pipeline for data collecting, model training and online serving. The deployed system significantly improves the click-through rate (CTR) of the homepage on Mobile Tmall App.

The remainder of this paper is organized as follows: related works are reviewed in section 2; Section 3 elaborates the technical details of MIND; In section 4, we detail the experiments for comparing MIND with existing methods on several public benchmarks and online serving; Section 5 introduces the deployment of MIND in large-scale industrial application; The last section gives conclusion and future work of this paper.

## 2 RELATED WORK

**Deep Learning for Recommendation.** Inspired by the success of deep learning in computer vision and natural language processing [16], much effort has been put for developing deep learning-based recommendation algorithms [3]. Besides the industrial applications proposed by [5, 30], various types of deep models have gained significant attention. Neural Collaborative Filtering (NCF) [9], DeepFM [7] and Deep Matrix Factorization Models (DMF) [26] construct a neural network composed of several MLPs to model the interaction between users and items. [22] presents a novel solution to top-N sequential recommendation by providing an united and flexible network for capturing more features.

**User Representation.** Representing users as vectors is commonly used in RS. Traditional methods assemble user preference as vectors composed of interested items [10, 21], keywords [6] and topics [28]. As the emergence of distributed representation learning, user embeddings obtained by neural networks are widely used. [4] employs RNN-GRU to learn user embeddings from the temporal ordered review documents. [29] learns user embedding vectors from word embedding vectors and applies them to recommending scholarly microblogs. [2] proposes a novel convolutional neural network based model that explicitly learns and exploits user embeddings in conjunction with features derived from utterances.

**Capsule Network.** The concept of "Capsule", a small group of neurons assembled to output a whole vector, is firstly proposed by Hinton [11] at 2011. Instead of backpropagation, dynamic routing

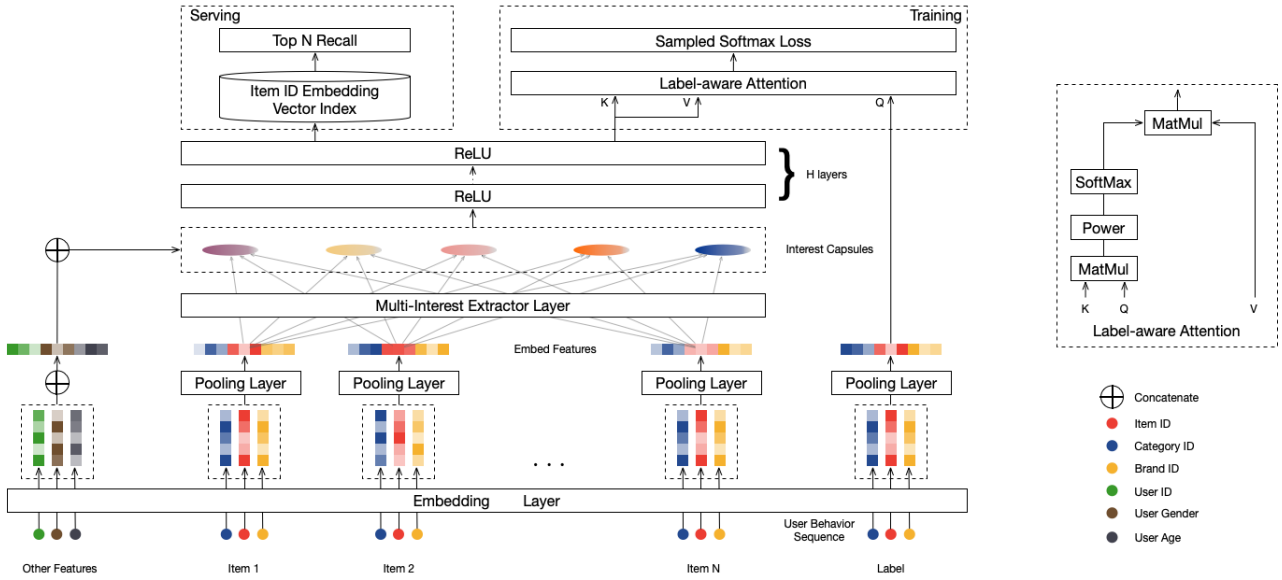


**Figure 2: Overview of MIND. MIND takes user behaviors with user profile features as inputs, and outputs user representation vectors for item retrieval in the matching stage of recommendation. Id features from input layer are transformed into embeddings through the embedding layer, and embeddings of each item are further averaged by a pooling layer. User behavior embeddings are fed into the multi-interest extractor layer, which produces interest capsules. By concatenating interest capsules with user profile embedding and transforming the concatenated capsules by several ReLU layers, user representation vectors are obtained. An extra label-aware attention layer is introduced to guide the training process. At serving, the multiple user representation vectors are used to retrieve items through an approximate nearest neighbor lookup approach.**

[20] is used to learn the weights on the connections between capsules, which is improved by utilizing Expectation-Maximization algorithm [12] to overcome several deficiencies and achieves better accuracy. These two main differences to conventional neural network make capsule networks capable of encoding the relationship between the part and the whole, which is adavanced not only in computer vision but also in natural language processing and knowledge graph. [27] investigates the capsule networks for text classification and proposes three strategies to boost the performance. [19] uses capsule network to model relationship triples for knowledge graph completion and search personalization.

## 3 METHOD

### 3.1 Problem Formalization

The objective of the matching stage for industrial RS is to retrieve a subset of items from the billion-scale item pool $\mathcal{I}$ for each user $u \in \mathcal{U}$ such that the subset contains only thousands of items and each item is relevant to interests of the user. In order to achieve this objective, historical data generated by RS is collected for building a matching model. Specifically, each instance can be represented by a tuple $(\mathcal{I}_u, \mathcal{P}_u, \mathcal{F}_i)$, where $\mathcal{I}_u$ denotes the set of items interacted by user $u$ (also called user behavior), $\mathcal{P}_u$ the basic profiles of user $u$ (like user gender and age), $\mathcal{F}_i$ the features of target item (such as item id, category id, brand id, seller id and title etc.).

The core task of MIND is to learn a function for mapping raw features into user representations, which can be formulated as

$$\mathrm{V}_u = f_{user}\left(\mathcal{I}_u, \mathcal{P}_u\right), \tag{1}$$

where $\mathrm{V}_u = \left(\overrightarrow{\boldsymbol{v}}_u^1, ..., \overrightarrow{\boldsymbol{v}}_u^K\right) \in \mathbb{R}^{d \times K}$ denotes the representation vectors of user $u$, $d$ the dimension, $K$ the number of representation vectors. When $K = 1$, one representation vector is used, just like YouTube DNN. Besides, the representation vector of target item $i$ is obtained by an embedding function as

$$\overrightarrow{\boldsymbol{e}}_i = f_{item}\left(\mathcal{F}_i\right), \tag{2}$$

where $\overrightarrow{\boldsymbol{e}}_i \in \mathbb{R}^{d \times 1}$ denotes the representation vector of item $i$, and the detail of $f_{item}$ will be illustrated in the "Embedding & Pooling Layer" section.

When user representation vectors and item representation vectors are learned, top $N$ candidate items are retrieved according to a scoring function. As a user may own several representation vectors, the item score is determined by the nearest representation vector so the scoring function is:

$$f_{score}\left(\mathrm{V}_u, \overrightarrow{\boldsymbol{e}}_i\right) = \max_{1 \le k \le K} \overrightarrow{\boldsymbol{e}}_i^{\mathrm{T}} \overrightarrow{\boldsymbol{v}}_u^k, \tag{3}$$

where $N$ is the predefined number of items to be retrieved in the matching stage.

### 3.2 Embedding & Pooling Layer

As shown in Figure 2, the input of MIND consists of three groups, user profile $\mathcal{P}_u$, user behavior $\mathcal{I}_u$, and label item $\mathcal{F}_i$. Each group contains several categorical id features, and these id features are of extremely high dimension. For instance, the number of item ids is about billions, thus we adopt the widely-used embedding technique to embed these id features into low-dimensional dense vectors (a.k.a embeddings), which significantly reduces the number of parameters and eases the learning process. For id features (gender,

age, etc.) from $\mathcal{P}_u$, corresponding embeddings are concatenated to form the user profile embedding $\overrightarrow{p}_u$. For item ids along with other categorical ids (category id, brand id and seller id etc.) that have been proved to be useful for cold-start items [24] from $\mathcal{F}_i$, corresponding embeddings are further passed through an average pooling layer to form the label item embedding $\overrightarrow{e}_i$. To enrich the feature space, other attributes of items such as color, texture and style, are easily to be embed and combined to the item embedding, which is expected to improve the expression ability of the embeddings. Lastly, for items from user behavior $\mathcal{I}_u$, corresponding item embeddings are collected to form the user behavior embedding $\mathrm{E}_u = \{\overrightarrow{e}_j, j \in \mathcal{I}_u\}$.

### 3.3 Multi-Interest Extractor Layer

We argue that representing user interests by one representation vector can be a bottleneck for capturing diverse interests of users, because we have to compress all information related with diverse interests of users into one representation vector. Thus, all information about diverse interests of users is mixed together, causing inaccurate item retrieval for the matching stage. Instead, we adopt multiple representation vectors to express distinct interests of users separately. By this way, diverse interests of users are considered separately in the matching stage, enabling more accurate item retrieval for every aspect of interests.

To learn multiple representation vectors, we utilize clustering process to group user's historical behaviors into several clusters. Items from one cluster are expected to be closely related and collectively represent one particular aspect of user interests. Here, we design the multi-interest extractor layer for clustering historical behaviors and inferring representation vectors for each cluster. Since the design of multi-interest extractor layer is inspired by the recently proposed dynamic routing for representation learning in capsule network [11, 12, 20], we firstly revisit essential basics in order to make this paper self-contained.

*3.3.1 Dynamic Routing Revisit.* We briefly introduce dynamic routing [20] for representation learning of capsules, a new form of neural units represented by vectors. Suppose we have two layers of capsules, and we refer capsules from the first layer and the second layer as low-level capsules and high-level capsules respectively. The goal of dynamic routing is to compute the values of high-level capsules given the values of low-level capsules in an iterative way. In each iteration, given low-level capsules $i \in \{1, ..., m\}$ with corresponding vectors $\overrightarrow{c}_i^l \in \mathbb{R}^{N_l \times 1}, i \in \{1, ..., m\}$ and high-level capsules $j \in \{1, ..., n\}$ with corresponding vectors $\overrightarrow{c}_j^h \in \mathbb{R}^{N_h \times 1}, j \in \{1, ..., n\}$, the routing logit $b_{ij}$ between low-level capsule $i$ and high-level capsule $j$ is computed by

$$b_{ij} = (\overrightarrow{c}_j^h)^T \mathrm{S}_{ij} \overrightarrow{c}_i^l, \tag{4}$$

where $\mathrm{S}_{ij} \in \mathbb{R}^{N_h \times N_l}$ denotes the bilinear mapping matrix to be learned.

With routing logits calculated, the candidate vector for high-level capsule $j$ is computed as weighted sum of all low-level capsules

$$\overrightarrow{z}_j^h = \sum_{i=1}^{m} w_{ij} \mathrm{S}_{ij} \overrightarrow{c}_i^l, \tag{5}$$

where $w_{ij}$ denotes the weight for connecting low-level capsule $i$ and high-level capsule $j$ and is calculated by performing softmax on routing logits as

$$w_{ij} = \frac{\exp b_{ij}}{\sum_{k=1}^{m} \exp b_{ik}}. \tag{6}$$

Finally, a non-linear "squash" function is applied to obtain the vectors of high-level capsules as

$$\overrightarrow{c}_j^h = squash(\overrightarrow{z}_j^h) = \frac{\left\|\overrightarrow{z}_j^h\right\|^2}{1 + \left\|\overrightarrow{z}_j^h\right\|^2} \frac{\overrightarrow{z}_j^h}{\left\|\overrightarrow{z}_j^h\right\|}. \tag{7}$$

The values of $b_{ij}$ are initialized to zeros, and the routing process is usually repeated three times to converge. When routing finished, high-level capsule's values $\overrightarrow{c}_j^h$ are fixed and can be used as inputs for next layers.

*3.3.2 B2I Dynamic Routing.* In a nutshell, capsule is a new kind of neuron represented by one vector instead of one scalar used in ordinary neural networks. The vector-based capsule is expected to be able to represent different properties of an entity, in which the orientation of a capsule represents one property and the length of the capsule is used to represent the probability that the property exists. Correspondingly, the objective of the multi-interest extractor layer is to learn representations for expressing properties of user interests as well as whether corresponding interests exist. The semantic connection between capsules and interest representations motivates us to regard the behavior/interest representations as behavior/interest capsules and employ dynamic routing to learn interest capsules from behavior capsules. Nevertheless, the original routing algorithm proposed for image data is not directly applicable for processing user behavior data. So, we propose Behavior-to-Interest (B2I) dynamic routing for adaptively aggregating user's behaviors into interest representation vectors, and it differs from original routing algorithm in three aspects.

*Shared bilinear mapping matrix.* We use fixed bilinear mapping matrix S instead of a separate bilinear mapping matrix for each pair of low-level capsules and high-level capsules in original dynamic routing due to two considerations. On the one hand, user behaviors are of variable-length, ranging from dozens to hundreds for Tmall users, thus the use of fixed bilinear mapping matrix is generalizable. On the other hand, we hope interest capsules lie in the same vector space, but different bilinear mapping matrice would map interest capsules into different vector spaces. Thus, the routing logit is calculated by

$$b_{ij} = \overrightarrow{u}_j^T \mathrm{S} \overrightarrow{e}_i, \qquad i \in \mathcal{I}_u, j \in \{1, ..., K\}, \tag{8}$$

where $\overrightarrow{e}_i \in \mathbb{R}^d$ denotes the embedding of behavior item $i$, $\overrightarrow{u}_j \in \mathbb{R}^d$ the vector of interest capsule $j$. The bilinear mapping matrix $\mathrm{S} \in \mathbb{R}^{d \times d}$ is shared across each pair of behavior capsules and interest capsules.

*Randomly initialized routing logits.* Owing to the use of shared bilinear mapping matrix S, initializing routing logits to zeros will lead to the same initial interest capsules. Then, the subsequent iterations will be trapped in a situation, where different interest capsules remain the same all the time. To mitigate this phenomenon, we sample a random matrix from gaussian distribution $\mathcal{N}(0, \sigma^2)$ for initial routing logits to make initial interest capsules differ from each other, similar to the well-established K-Means clustering algorithm.

*Dynamic interest number.* As the number of interest capsules owned by different users may be different, we introduce a heuristic rule for adaptively adjusting the value of $K$ for different users. Specifically, the value of $K$ for user $u$ is computed by

$$K'_u = \max(1, \min(K, \log_2(|\mathcal{I}_u|))). \tag{9}$$

This strategy for adjusting the number of interest capsules can save some computing resources for those users with fewer interests.

The whole dynamic routing procedure is listed in Algorithm 1.

**Algorithm 1** B2I Dynamic Routing.

**Input:** behavior embeddings $\left\{\overrightarrow{\boldsymbol{e}}_i, i \in \mathcal{I}_u\right\}$, iteration times $r$, number of interest capsules $K$

**Output:** interest capsules $\left\{\overrightarrow{\boldsymbol{u}}_j, j = 1, ..., K'_u\right\}$

1: calculate adaptive number of interest capsules $K'_u$ by (9)
2: for all behavior capsule $i$ and interest capsule $j$: initialize $b_{ij} \sim \mathcal{N}(0, \sigma^2)$
3: **for** $k \leftarrow 1, r$ **do**
4: for all behavior capsule $i$: $w_{ij} \leftarrow softmax(b_{ij})$
5: for all interest capsule $j$: $\overrightarrow{\boldsymbol{z}}_j = \sum_{i \in \mathcal{I}_u} w_{ij} S \overrightarrow{\boldsymbol{e}}_i$
6: for all interest capsule $j$: $\overrightarrow{\boldsymbol{u}}_j \leftarrow squash(\overrightarrow{\boldsymbol{z}}_j)$
7: for all behavior capsule $i$ and interest capsule $j$: $b_{ij} \leftarrow \overrightarrow{\boldsymbol{u}}_j^T S \overrightarrow{\boldsymbol{e}}_i$
8: **end for**
9: **return** $\left\{\overrightarrow{\boldsymbol{u}}_j, j = 1, ..., K'_u\right\}$

## 3.4 Label-aware Attention Layer

Through multi-interest extractor layer, several interest capsules are generated from user's behavior embeddings. Different interest capsules represent different aspects of user interests, and the relevant interest capsule is used for evaluating user's preference on specific items. Therefore, during training, we design a label-aware attention layer based on scaled dot-product attention [23] to make the target item choose which interest capsule is used. Specifically, for one target item, we calculate the compatibilities between each interest capsule and target item embedding, and compute a weighted sum of interest capsules as user representation vector for the target item, where the weight for one interest capsule is determined by corresponding compatibility. In label-aware attention, the label is the query and the interest capsules are both keys and values, as shown in Figure 2. The output vector of user $u$ with respect to item $i$ is computed as

$$\begin{aligned}\overrightarrow{\boldsymbol{v}}_u &= \text{Attention}\left(\overrightarrow{\boldsymbol{e}}_i, \mathrm{V}_u, \mathrm{V}_u\right) \\ &= \mathrm{V}_u \text{softmax}(\text{pow}(\mathrm{V}_u^{\mathrm{T}} \overrightarrow{\boldsymbol{e}}_i, p)),\end{aligned}$$

where pow denotes element-wise exponentiation operator, $p$ a tunable parameter for adjusting the attention distribution. When $p$ is close to 0, each interest capsule attends to receive even attention. When $p$ is bigger than 1, as $p$ increases, the value has bigger dot-product will receive more and more weight. Consider the limit case, when $p$ gets infinity, the attention mechanism becomes a kind of hard attention to pick the value who has the biggest attention and ignore others. In our experiments, we find out that using hard attention leads to faster convergence.

## 3.5 Training & Serving

With the user vector $\overrightarrow{\boldsymbol{v}}_u$ and the label item embedding $\overrightarrow{\boldsymbol{e}}_i$ ready, we compute the probability of the user $u$ interacting with the label item $i$ as

$$\Pr(i|u) = \Pr\left(\overrightarrow{\boldsymbol{e}}_i | \overrightarrow{\boldsymbol{v}}_u\right) = \frac{\exp\left(\overrightarrow{\boldsymbol{v}}_u^{\mathrm{T}} \overrightarrow{\boldsymbol{e}}_i\right)}{\sum_{j \in \mathcal{I}} \exp\left(\overrightarrow{\boldsymbol{v}}_u^{\mathrm{T}} \overrightarrow{\boldsymbol{e}}_j\right)}. \tag{10}$$

Then, the overall objective function for training MIND is

$$L = \sum_{(u,i) \in \mathcal{D}} \log \Pr(i|u), \tag{11}$$

where $\mathcal{D}$ is the collection of training data containing user-item interactions. Since the number of items scales to billions, the sum operation of the denominator (10) is computationally prohibitive. Thus, we use the sampled softmax technique [18] to make the objective function trackable and choose the Adam optimizer [14] for training MIND.

After training, the MIND network except for the label-aware attention layer can be used as user representation mapping function $f_{user}$. At serving time, user's behavior sequence and user profile are fed into the $f_{user}$ function, producing multiple representation vectors for each user. Then, these representation vectors are used to retrieve top $N$ items by an approximate nearest neighbor approach [13]. These items with highest similarities with user's representation vectors are retrieved and constitute the final set of candidate items for the matching stage of RS. Please note that, when a user has new actions, it will alter his/her behavior sequence as well as the corresponding user representation vectors, thus MIND enables real-time personalization for the matching stage.

## 3.6 Connections with Existing Methods

Here, we make some remarks about the relations between MIND and two existing methods, illustrating their similarities as well as differences.

*YouTube DNN.* Both MIND and YouTube DNN utilize deep neural networks to model behavior data to generate user representations, which are used for large-scale item retrieval in the matching stage of industrial RS. However, YouTube DNN uses one vector to represent a user while MIND uses multiple vectors for that. When the value of $K$ in Algorithm 1 equals to 1, MIND is similar to YouTube DNN, thus MIND can be viewed as generalization of YouTube DNN.

*DIN.* In terms of capturing diverse interests of users, MIND and DIN share the similar goal. However, the two methods differ in the way of achieving the goal as well as applicability. To deal with diverse interests, DIN applies an attention mechanism at the item level, while MIND employs dynamic routing to generate interest capsules and considers diversity at the interest level. Moreover, DIN focuses on the ranking stage as it handles thousands of items, however, MIND decouples the process of inferring user representations and measuring user-item compatibility, making it applicable to billion-scale items in the matching stage.

**Table 1: Statistics of the two datasets for offline evaluation.**

| Dataset | Users | Goods | Categories | Samples |
|---|---|---|---|---|
| Amazon Books | 351,356 | 393,801 | 1 | 6,271,511 |
| TmallData | 2,014,865 | 934,751 | 6,377 | 50,929,802 |

# 4 EXPERIMENTS

## 4.1 Offline Evaluation

In this section, we present the comparisons between MIND and existing methods in terms of recommendation accuracy on several datasets under offline settings.

*4.1.1 Datasets and Experimental Setup.* We choose two datasets for evaluating recommendation performance. One is Amazon Books[1] provided by [8, 17], representing one of the most widely-used public dataset for e-commerce recommendations. The other called TmallData is held out from Mobile Tmall App, containing historical behaviors of randomly sampled two millions of Tmall users in 10 days. For Amazon Books, we only keep items which have been reviewed at least 10 times and users who have reviewed at least 10 items. For TmallData, we filter out items clicked by less than 600 unique users. The statistics of the two datasets are shown in Table 1.

We choose next item prediction problem, that is predicting a user's next interaction, to evaluate the methods' performance, because it is the core task in the matching stage of RS. After dividing the user-item interaction data of each dataset randomly into training set and test set by a ratio of 19:1, for each user, a randomly selected item interacted by the user is used as target item, while the items interacted before the target item are collected as the user behaviors. Hit rate is adopted as the main metric to measure the recommendation performance, define as:

$$HitRate@N = \frac{\sum_{(u,i)\in\mathcal{D}_{test}} I(\text{target item occurs in top } N)}{|\mathcal{D}_{test}|}, \tag{12}$$

where $\mathcal{D}_{test}$ denotes the test set consisting of pairs of users and target items $(u, i)$ and $I$ denotes the indicator function.

Hyperparameter tuning for the dimension of embedding vectors $d$ and the number of user interests $K$ is conducted by experiments on a group of parameters predefined according to the scale and distribution of each dataset, and each method is tested with best hyperparameters for a fair comparison.

*4.1.2 Comparing Methods.*

- **WALS** [1] WALS, short for Weighted Alternating Least Squares, is a classical matrix factorization algorithm for decomposing user-item interaction matrix into hidden factors of users and items. Recommendation is made based on compatibilities between hidden factors of users and target items.
- **YouTube DNN** [5] As mentioned above, YouTube DNN is one of the most successful deep learning method used for industrial recommendation systems.
- **MaxMF** [25] The method introduces a highly scalable method for learning nonlinear latent factorization to model multiple user interests.

[1] http://jmcauley.ucsd.edu/data/amazon/

*4.1.3 Experimental Results.* Table 2 summarizes the performance of MIND as well as baselines on two datasets in terms of HitRate@N (N = 10, 50, 100). Clearly, MIND accomplishes comparable performance to all of the baselines on both datasets. The matrix factorization approach, WALS, is beaten by other methods, revealing the power of deep learning for improving the matching stage of RS. However, equipped without deep learning, MaxMF performs much better than WALS, which can be explained by the fact that MaxMF generalizes standard MF to a nonlinear model and adopts multiple user representation vectors. It can be observed that methods employing multiple user representation vectors (MaxMF-$K$-interest, MIND-$K$-interest) performs generally better than other methods (WALS, YouTube DNN, MIND-1-interest). Therefore, using multiple user representation vectors is proved to be an effective way for modeling user's diverse interests as well as boosting recommendation accuracy. Moreover, we can observe that the improvement introduced by multiple user representation vectors is more significant for TmallData, as the users of Tmall tend to exhibit more diverse interests. This increasement of diversity can also be reflected by the best $K$ for each dataset, where the best $K$ for TmallData is larger than that for Amazon Books. The improvement of MIND-1-interest over YouTube DNN shows that dynamic routing serves as a better pooling strategy than average pooling. Considering the results of MaxMF and MIND-$K$-interest, it verifies that extracting multiple interests from user behaviors by dynamic routing outperforms the nonlinear modeling strategy used in MaxMF. This can be attributed to two points: 1) The multi-interest extractor layer utilizes a clustering procedure for generating interest representations, which achieves more precise representation of user. 2) Label-aware attention layer makes target item attend over multiple user representation vectors, enabling more accurate matching between user interests and target item.

## 4.2 Analysis of Hyperparameters

In this section, we conduct two experiments on Amazon Books to study the influence of the hyperparameters within multi-interest extractor layer and label-aware attention layer.

**Initialization of routing logits.** The random initialization for routing logits adopted in multi-interest extractor layer is similar to the initialization of K-means centroids, where the distributions of initial cluster centers have strong impact on the final clustering results. As the routing logits are initialized according to gaussian distribution $\mathcal{N}(0, \sigma^2)$, we concern about different values of $\sigma$ may lead to different convergence which has effect on the performance. To study the impact of $\sigma$, we initialize the routing logits $b_{ij}$ with 3 different values of $\sigma$, 0.1, 1 and 5. The results are shown by the upper part of Figure 3, where each curve of 3 values almost overlap. This observation reveals that MIND is robust to the values of $\sigma$, and it is rational to choose $\sigma = 1$ for our practical applications.

**Power number in label-aware attention.** As mentioned before, the power number $p$ within label-aware attention controls the proportion of each interest to the combined label-aware interest representation. We compare the performance of MIND as $p$ varies from 0 to $\infty$ and show the results by the lower part of Figure 3. Clearly, the performance of $p = 0$ is much worse than the others. The reason is that, when taking $p = 0$ each interest has the same

**Table 2: HitRate of different methods on the two datasets, where best performance is in boldface. HP denotes hyperparameters, including $K$ the number of interests and $d$ the dimension of embeddings. Only the results with hyperparameters having best performance is shown to demonstrate the effectiveness of corresponding methods. Percentages in the brackets indicate the relative improvements over YouTube DNN.**

| Dataset | HP | Metric | WALS | YouTube DNN | MaxMF-$K$-interest | MIND-1-interest | MIND-$K$-interest |
|---|---|---|---|---|---|---|---|
| Amazon Books | $K = 3$ $d = 36$ | HR@10 | 0.0144 (-37.66%) | 0.0231 | 0.0285 (+23.38%) | 0.0273 (+18.18%) | **0.0309 (+33.77%)** |
| | | HR@50 | 0.0553 (-25.87%) | 0.0746 | 0.0862 (+15.55%) | 0.0978 (+31.10%) | **0.1101 (+47.59%)** |
| | | HR@100 | 0.0907 (-20.65%) | 0.1143 | 0.1304 (+14.09%) | 0.1459 (+27.65%) | **0.1631 (+42.69%)** |
| TmallData | $K = 5$ $d = 64$ | HR@10 | 0.0372 (-36.84%) | 0.0589 | 0.0628 (+6.62%) | 0.0720 (+22.24%) | **0.0972 (+65.03%)** |
| | | HR@50 | 0.0831 (-33.84%) | 0.1256 | 0.1820 (+44.90%) | 0.1512 (+20.38%) | **0.2080 (+65.60%)** |
| | | HR@100 | 0.1126 (-31.67%) | 0.1648 | 0.2567 (+55.76%) | 0.1930 (+17.11%) | **0.2699 (+63.77%)** |

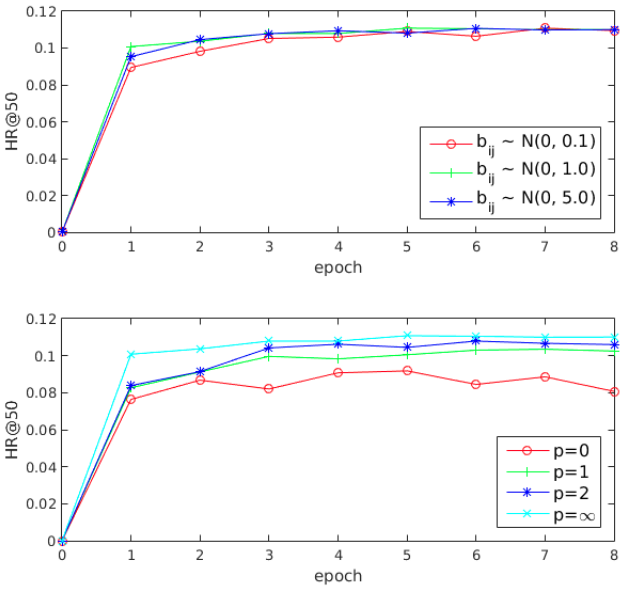


**Figure 3: Hyperparameters' impact. The upper part shows that MIND can achieve comparable results with different $\sigma$; the lower part shows that MIND performs better with bigger $p$.**

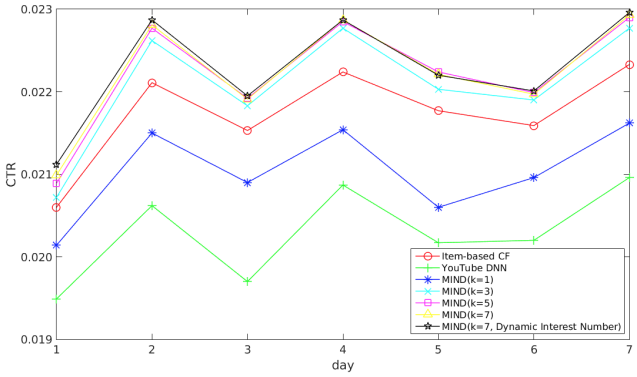


**Figure 4: Online CTRs in a week. MIND with 5 ~7 interests performs best in all comparing methods. MIND significantly beats the two baseline methods, item-based CF and YouTube DNN.**

attention thus the combined interest representation equals the average of interests with no reference to the label. Taking $p \geqslant 1$, the attention scores are proportional to the similarities between interest representation vectors and target item embeddings, which makes the combined interest representation a weighted sum of interests. It also shows that performance gets better as $p$ increases, since the representation vector of the interest with more similarity to the target item acquires larger attention, which evolves to a hard attention scheme as $p = \infty$. By this scheme, the interest representation nearest to the target item dominates the combined interest representation, enabling MIND converge faster and perform the best.

## 4.3 Online Experiments

We conduct online experiments by deploying MIND to handle real traffic at Tmall homepage for one week. To make comparisons fairly, all methods deployed in the matching stage are followed by the same ranking procedure. CTR, short for click-through-rate, a widely used industrial metric, is used to measure the performance of methods for serving online traffic.

There are two baseline methods for online experiments. One is item-based CF, which is the base matching algorithm serving the majority of the online traffic. The other is YouTube DNN, which is the well-known deep learning-based matching model. We deploy all comparing methods in an A/B test framework, and one thousand of candidate items are retrieved by each method, which then fed to the ranking stage for final recommendation.

The experimental results are summarized in Figure 4. It is clearly that MIND outperforms item-based CF and YouTube DNN, which indicates that MIND generates a better user representation. Besides, we make the following observations: 1) As is optimized by the long-term practice, item-based CF performs better than YouTube DNN which is also exceeded by MIND with single interest. 2) A very noticeable trend is that the performance of MIND gets better as the number of extracted interests increases from 1 to 5. 3) The performance of MIND peaks when the number of extracted interests reaches 5, after that the CTR remains constant and the improvement of 7 interests is ignorable. 4) MIND with dynamic interest number has the comparable performance with MIND with 7 interests. From the observations above, we make several conclusions. First, for Tmall, the optimal number of user interests is 5 ~7, which reveals the average diversity of user interests. Second, the dynamic interest number mechanism does not bring CTR gain, but during the experiments we recognize the scheme can decrease the cost of serving, which benefits large-scale service such as Tmall

and is more adoptable in practice. In a word, the online experiments validate that MIND achieves an better solution to model users with diverse interests and can significantly advance the whole RS.

## 4.4 Case Study

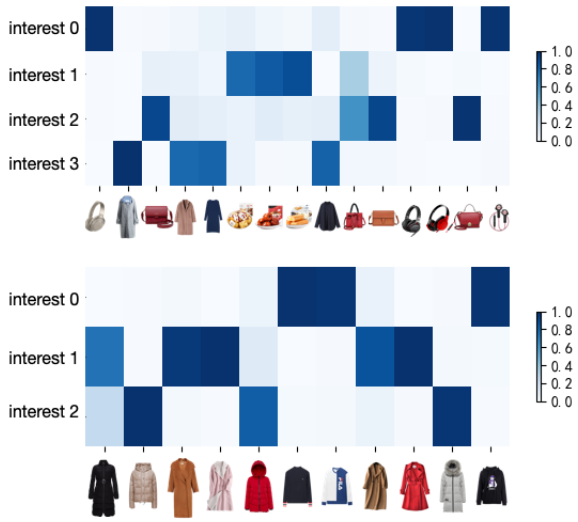


**Figure 5: Heatmap of coupling coefficients for two users. Each class of behaviors has the max coupling coefficients on the corresponding interest. User C (upper) and user D (below) have different granularity of interests.**

*4.4.1 Coupling Coefficients.* The coupling coefficients between behavior capsules and interest capsules quantify the grade of membership of behaviors to interests. In this section, we visualize these coupling coefficients to show that the interest extraction process is interpretable.

Figure 5 illustrates the coupling coefficients associated to two users randomly selected from Tmall daily active users, where each row corresponds to one interest capsule and each column corresponds to one behavior. It shows that user C (upper) has interacted with 4 classes of goods (headphones, snacks, handbags and clothes), each of which has the max coupling coefficients on one interest capsule and forms the corresponding interest. While user D (below) is interested only in clothes, thus the 3 interests with finer grain size (sweaters, overcoats and down jackets) are resolved from the behaviors. Regarding this result, we confirm that each class of user behaviors are clustered together and form the corresponding interest representation vector.

*4.4.2 Item Distribution.* At serving time, items similar to user interests are retrieved by nearest neighbor search. We visualize the distribution of these items recalled by each interest based on their similarity to the corresponding interest. Figure 6 shows the item distributions of the same user (user C) mentioned by Figure 5 (upper). The distributions are obtained by two methods respectively, where the upper 4 axes demonstrate the items recalled by 4 interests based on MIND while the lowest axis illustrates that based on YouTube DNN. The items are scattered at the axes according to their similarity with the interests, which has been scaled to 0 ~1 by min-max normalization and rounded to the nearest 0.5. One

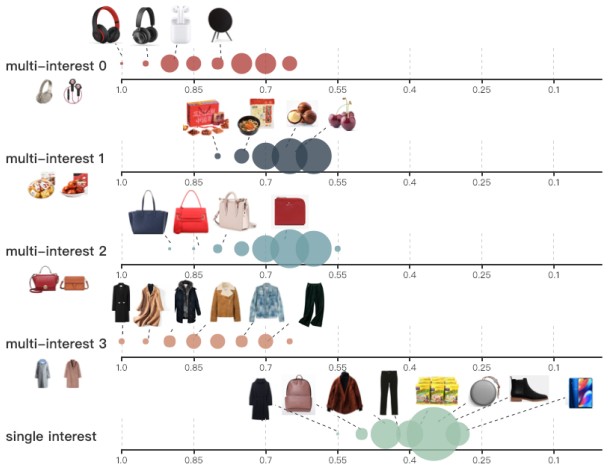


**Figure 6: The distribution of items recalled by each interest corresponding to the user behaviors exampled on the left. Each interest is demonstrated by one axis, of which the coordinate is the similarity between items and interests. The size of the point is proportional to the number of the items with the specific similarity.**

point is assembled by the items lying within the specific range, thus the size of each point represents the number of the items with the corresponding similarity. We also show some items selected randomly from all the candidates. As expected, the items recalled by MIND are strongly correlated with the corresponding interest, while that by YouTube DNN vary widely along the categories of items and have lower similarity to the user's behaviors.

## 5 SYSTEM DEPLOYMENT

In this section, we describe the implementation and deployment of MIND at Tmall. A typical workflow composed of several basic platforms is shown as Figure 7 and detailed as below:

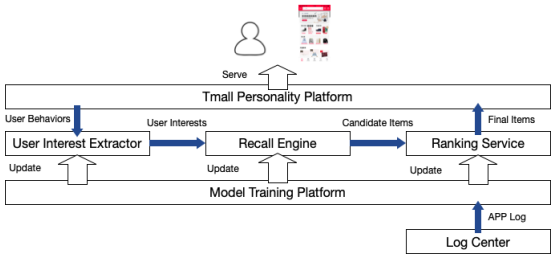


**Figure 7: Architecture of the RS at Tmall.**

As users launch Mobile Tmall APP, the requests for recommendation are sent to *Tmall Personality Platform*, the server cluster integrated with a bunch of plug-in modules and served as online recommender service of Tmall. Recent behaviors of users including real-time user feedback are retrieved by *Tmall Personality Platform* and sent to *User Interest Extractor* which is the main module implementing MIND for transforming user behaviors to multiple user interests. Subsequently, *Recall Engine* searches for the items with embedding vectors nearest to the user interests. Items triggered by different interests are merged together as candidate items and

sorted by their similarity to the user interests. The whole procedure of selecting thousands of candidate items from the billion-scale item pool by *User Interest Extractor* and *Recall Engine* can be fulfilled in less than 15 milliseconds, which is an increase of only 10% compared to the original version with single interest, due to the parallelism of searching items for each interest generated by MIND. Taking a tradeoff between the scope of items and the response time of the system, top 1000 of these candidate items are scored by *Ranking Service* which predicts CTRs with a bunch of features. Finally, Tmall Personality Platform completes the item list as the recommendation results shown to users. Both *User Interest Extractor* and *Ranking Service* are trained on *Model Training Platform* using 100 GPUs, by which the training can be executed in 8 hours. Benefiting from the superior performance of *Model Training Platform*, the deep network served for prediction is updated every day, which guarantees the newly released products to be calculated and exposured.

## 6 CONCLUSION AND FUTURE WORK

In this paper, we propose a new structure of neural network, namely Multi-Interest Network with Dynamic routing (MIND), to represent user's diverse interests for the matching stage in e-commerce recommendation, which involves billion scale users and items. Specifically, we design a multi-interest extractor layer with a variant dynamic routing to extract user's diverse interests which are then trained with a novel label-aware attention scheme. Offline experiments are conducted to demonstrate that MIND achieves superior performance on public benchmarks. Online CTRs are also reported to demonstrate the effectiveness and feasibility of MIND at Tmall's live production. For future work, we will pursue two directions. The first is to incorporate more information about user's behavior sequence, such as behavior time etc. The second is to optimize the initialization scheme of dynamic routing, referring to K-means++ initialization scheme, so as to achieve a better user representation.

## ACKNOWLEDGMENTS

We would like to thank colleagues of our team - Jizhe Wang, Andreas Pfadler, Jiaming Xu, Wen Chen, Lifeng Wang, Xin Guo and Cheng Guo for useful discussions and supports on this work. We are grateful to our cooperative team - search engineering team. Dik Lun Lee is supported by the Research Grants Council HKSAR GRF (No. 16215019). We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

## REFERENCES

[1] Christopher R Aberger. 2016. *Recommender: An analysis of collaborative filtering techniques.* Technical Report.
[2] Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling Context with User Embeddings for Sarcasm Detection in Social Media. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 167–177.
[3] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2018. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review* (2018), 1–37.
[4] Tao Chen, Ruifeng Xu, Yulan He, Yunqing Xia, and Xuan Wang. 2016. Learning user and product distributed representations using a sequence model for sentiment analysis. *IEEE Computational Intelligence Magazine* 11, 3 (2016), 34–44.
[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
[6] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 278–288.
[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, 1725–1731.
[8] 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.
[9] 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.
[10] Jon Herlocker, Joseph A Konstan, and John Riedl. 2002. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval* 5, 4 (2002), 287–310.
[11] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. 2011. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*. Springer, 44–51.
[12] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *International Conference on Learning Representations*.
[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734* (2017).
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.
[17] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.
[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[19] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, and Dinh Phung. 2018. A Capsule Network-based Embedding Model for Search Personalization. *arXiv preprint arXiv:1804.04266* (2018).
[20] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*. 3856–3866.
[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.
[22] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 565–573.
[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[24] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 839–848.
[25] Jason Weston, Ron J Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 65–68.
[26] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. 3203–3209.
[27] Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3110–3119.
[28] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)* 33, 3 (2015), 10.
[29] Yang Yu, Xiaojun Wan, and Xinjie Zhou. 2016. User embedding for scholarly microblog recommendation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 2. 449–453.
[30] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.