Knowledge Enhanced Multi-Interest Network for the Generation of Recommendation Candidates

Danyang Liu*
ldy591@mail.ustc.edu.cn
University of Science and Technology of China & Meituan
Hefei, China

Yuji Yang*
Meituan
Beijing, China
yangyuji02@meituan.com

Mengdi Zhang
Meituan
Beijing, China
zhangmengdi02@meituan.com

Wei Wu
Meituan
Beijing, China
wuwei19850318@gmail.com

Xing Xie
Microsoft Research
Beijing, China
xingx@microsoft.com

Guangzhong Sun
University of Science and Technology of China
Hefei, China
gzsun@ustc.edu.cn

ABSTRACT

Candidate generation task requires that candidates related to user interests need to be extracted in realtime. Previous works usually transform a user's behavior sequence to a unified embedding, which can not reflect the user's multiple interests. Some recent works like Comirec [4] and Octopus [21] use multi-channel structures to capture users' diverse interests. They cluster users' historical behaviors into several groups, claiming that one group represents one interest. However, these methods have some limitations. First, an item may correspond to multiple interests of users, thereby simply allocating it to just one interest group will make the modeling of users' interests coarse-grained and inaccurate. Second, explaining user interests at the level of items is rather vague and not convincing.

In this paper, we propose a **K**nowledge **E**nhanced **M**ulti-**I**nterest Network: KEMI, which exploits knowledge graphs to help learn users' diverse interest representations via heterogeneous graph neural networks (HGNNs)[26, 39] and a novel dual memory network. Specifically, we use HGNNs to capture the semantic representation of knowledge entities and a novel dual memory network to learn a user's diverse interests from his behavior sequence. Through memory slots of the user memory network and the item memory network, we can learn multiple interests for each user and each item. Meanwhile, by binding the entities to the channels of memory networks, we enable it to be explained from the perspective of the knowledge graph, which enhances the interpretability and understanding of user interests. We conduct extensive experiments on two industrial and publicly available datasets. Experimental results demonstrate that our model achieves significant improvements over state-of-the-art baseline models.

*Both authors contributed equally to this research.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557114

CCS CONCEPTS

• **Information systems** → *Recommender systems*; **Personalization**; **Data mining**; • **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

recommender systems, knowledge graph, user modeling

ACM Reference Format:
Danyang Liu, Yuji Yang, Mengdi Zhang, Wei Wu, Xing Xie, and Guangzhong Sun. 2022. Knowledge Enhanced Multi-Interest Network for the Generation of Recommendation Candidates. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557114

1 INTRODUCTION

Large-scale commercial recommender systems are usually divided into two stages: candidate generation stage and precise ranking stage. Candidate generation stage is responsible for retrieving thousands of candidate items matching to user interests in real-time. The precise ranking stage is responsible for accurately predicting the probability that users are interested in candidate items. In this work, we mainly focus on the candidate generation stage.

The mainstream candidate generation methods are based on similarity search (e.g., k-nearest neighbors algorithm), where users and items are represented in the same space and the relatedness of users and items is reflected by their representational similarity. In the early days, people used to exploit raw features intensively; e.g., users could be represented by a set of keywords and tags, where items associated with the same keywords and tags were regarded as proper candidates. Due to the coarse-granularity of raw features, it is hard to measure the relevance of a user and an item precisely. In recent years, increasing attention has been paid to representations learned by deep neural networks. Typical methods include YouTube DNN [6], DSSM [13] and CDSSM [27] in which users and items are encoded via deep neural networks and their relevance is reflected by their closeness in the common latent space. These methods assign each user and item with one single vector, which is hard to represent a user comprehensively when his interest is diversified.

As a result, some items that users are interested in will be missed during this stage.

Some recent "multi-channel structures" models seem to be promising solutions towards the above problem, which are more capable of representing users' diverse interests, e.g. MIND [17], Comirec [4] and Octopus [21]. With the deployment of multiple channels, a user can be modeled from different perspectives and his diverse interests can be jointly captured with different representations. However, these methods cluster users' behavior sequences into different groups to represent different interests, ignoring that an item may contain multiple fine-grained interests. As shown in Figure 1, the items below are from a real world news feed application, and all these news show diverse interest characteristics. Take item1 as an example, this is a news about a famous tennis player: *Djokovic investing in a Covid-19 treatment company.* This news involves at least three kinds of interests: sports, finance, and health. Simply allocating it to one interest channel will make the modeling of user interests inaccurate.

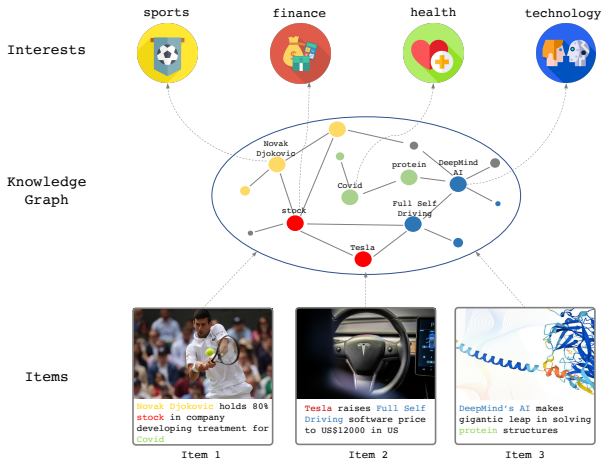


Figure 1: An example of knowledge graphs helping to extract the fine-grained interests of items, and further model users' multiple fine-grained interests. Each item that a user has interacted with is associated with different entities. Through the knowledge graph, different entities can learn different types of interests, thereby helping to understand the multiple fine-grained interests of the user.

To deal with the above "One item may contain multiple fine-grained interests" issue, we introduce knowledge graphs to enhance the modeling of multi-interests of users and items. As elaborately constructed semantic networks, knowledge graphs contain a large number of facts and relations. Incorporating knowledge graphs as side information has been proven to be promising in improving both accuracy and explainability for recommender systems [11, 33, 41]. By exploring the interlinks within a knowledge graph, we can get more comprehensive information about the item from the neighbor entities, which can help us understand the diverse interests of items and further model the fine-grained interests of users. Meanwhile, we can also link interests and knowledge entities to better explain the diverse interests of users from the perspective of the knowledge graph.

In this paper, a novel multi-interest representation model, KEMI, is proposed. We first apply HGNNs [39] to learn the rich semantic representations of items and their associated entities. To capture users' diverse interests representations from users' historical behavior sequences, we propose a novel dual memory network, which includes an interest memory network and a user memory network. The interest memory network learns the interest distribution from item level and knowledge entity level, aiming to obtain the overall and fine-grained interest information, while the user memory network learns the user's personal sequential representations. We get the user's multiple interest representations through the dual memory network. To better learn the interests distributions of items, we add a constrained interest loss. The 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 following contributions are made in this work.

- We propose KEMI, a novel knowledge enhanced multi-interest network, which comprehensively captures users' diverse interests with the help of knowledge graphs for candidate generation task in recommender systems.
- On top of the proposed framework, we propose to use HGNNs to learn the unified representations of items and knowledge entities with multiple types of relations, then we feed the items and knowledge entities to a novel dual memory network structure to learn the overall and fine-grained interests representations from users' behavior sequences. And we design a constrained interest loss to help better learn users' interests distributions.
- Extensive experimental studies are carried out with both industrial and publicly available datasets, where KEMI achieves substantial improvements over state-of-the-art baseline methods, and in the meanwhile, our method can give a good explanation for each interest channel with the help of knowledge graphs.

2 RELATED WORK

In this section, we introduce the related literature about deep recommendation system, generation of recommendation candidates, knowledge graph recommendation and memory networks.

Deep Recommendation System. Recently, deep learning has been revolutionizing recommender systems and achieving better performance in many recommendation scenarios such as e-commerce, online advertisement, and personalized content feed [13, 18, 22]. Generally speaking, the contribution of deep learning techniques to recommender systems has two aspects. First, compact and discriminative features can be automatically learned from raw data thanks to the superior representational capabilities of deep neural networks, e.g., high-level language models for text representation learning [19, 29], as well as through GNN and graph embeddings [35, 38]. Second, through the complex network structure, the complex behavior patterns of users can be modeled more accurately. For example, [7, 20, 24] propose to use a recurrent neural network to effectively represent the user's temporal interest; [24] uses memory networks to better capture the user's different

interests. [42, 43] make the user representation vary over different items with attention mechanisms to capture the diversity of user interests. It should be noted that most of today's deep recommendation algorithms are mainly used for the "ranking step", that is, the final selection of recommendation items from a small candidate set; by contrast, relatively limited progress has been made in candidate generation.

Generation of Recommendation Candidates. Candidate generation is the first step of recommender systems. Unlike the ranking stage, it requires both precision and efficiency: suitable candidates must be selected in realtime from a large number of items. As a result, mainstream algorithms shift the candidate generation problem to the similarity search problem. In addition to those well-established search paradigms, such as inverted-index [16], KNN search [31], Maximum Inner Product Search (MIPS) [15], candidates can be generated within a short time. Mainstream candidate generation algorithms have two common prerequisites. First, users and items must follow the same representation, e.g., keywords from a common set, or vectors in the same latent space. Second, the semantic relationship of users and items must be manifested through their representational similarity; for example, the co-occurrence of keywords, or the distance between two vectors. Obviously, the quality of a candidate is largely influenced by the ability to represent. Early on, people would turn to directly comparing raw features[1, 40], such as similarity in raw text. Due to the coarse-grained nature of raw features, it is difficult to accurately identify items of interest to users. In recent years, representations learned by deep neural networks have received increasing attention, such as [8, 44]. However, traditional representations generate a vector for each user, which is difficult to capture the different interests of users. Although existing multi-channel structures [5, 22]will generate different user representations, it is difficult to capture fine-grained user interests by only modeling isolated items that have multiple dimensions of interests, and these methods lack convincing explanations at the item level.

Knowledge Graph Recommendation. Knowledge graphs have been widely used to improve recommendation accuracy. [28] introduces an end-to-end framework, named RippleNet, which simulates the propagation of user preferences over the set of knowledge entities and alleviates the sparsity and cold start problem in recommender systems. [3] considers the joint learning of recommendation and knowledge graph completion. With the emerging techniques of graph neural networks (GNN), some researchers devise GNN-based models to utilize knowledge graphs for recommendation systems [30, 32]. [12] proposes to incorporate knowledge information to enhance the semantic representation of Key-Value Memory Network for sequence recommendation. Another major advantage of knowledge graphs is their ability to endow recommender systems with explainability. Connectivity between two nodes with a multi-hop path over the graph can serve as knowledge-aware reasoning because it reveals the semantic relationship between two nodes. [33] searches all potential paths connecting the user and the item, then adopts an LSTM on paths to capture the sequential dependencies of nodes for user preference inference. The reasoning is conducted by selecting the paths with the highest preference scores. To avoid enumerating all possible paths, [36] proposes a reinforced method called Policy-Guided Path Reasoning (PGPR), with three key components including a soft reward strategy, a user-conditional action pruning strategy, and a multi-hop scoring approach. In this work, we want to use knowledge graphs to represent users' diverse interests in a fine-grained way to improve recommendation accuracy and meanwhile gain better explainability.

Notation	Description
u	a user
i	an item
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
$\mathcal{I}_{pool}$	the item pool of candidate items
e	a knowledge entity
$\mathcal{G}$	the knowledge graph
m	the number of interests
M_I	the interest memory network
M_U	the user memory network
d	the dimension of embeddings

Table 1: Notations

Memory Network. The Memory Network extends existing neural network models with extra memory components, which could store informative facts from historical data[10] and better capture user's diverse interests. Memories will be read and updated when they are used to provide models with extra knowledge. Due to the advantages of memory networks in modeling long-sequence behaviors[5, 24] and diverse interests[23, 44] of users, they are widely used in recommendation systems to improve user modeling. [5] proposes to leverage external memory networks integrated with collaborative filtering for sequential recommendation. [24] proposes a memory-based MIMN model capturing long-term user interests from fairly long sequential user behavior data. [44] combines external memory and neural attention mechanism to capture fine-grained user preference across various interaction space. Due to the advantages of memory network in both modeling users' sequential behavior and users' diverse interests, we propose a novel dual memory network to better learn multi-interest representations from users' historical behavior sequences.

3 METHODS

3.1 Problem Formulation

Assume we have a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. For each user $u \in \mathcal{U}$, we have a sequence of user historical behaviors $H_u = \{i_1^u, i_2^u, ..., i_T^u\}$, sorted by time of the occurrence. Each item $i \in \mathcal{I}$ is associated with a set of knowledge entities $E_i = \{e_1^i, e_2^i, ..., e_k^i\}$. The objective of the candidate generation stage for industrial RS is to retrieve top-N candidate items from the billion-scale item pool $\mathcal{I}_{pool}$ for each user $u \in \mathcal{U}$ in real time, promising that each item is relevant to interests of the user. The core task of our KEMI model is to learn a function for mapping user historical behaviors into multiple user representations, which can be formulated as:

$$\mathbf{V}_u = f_{user}(H_u) \tag{1}$$

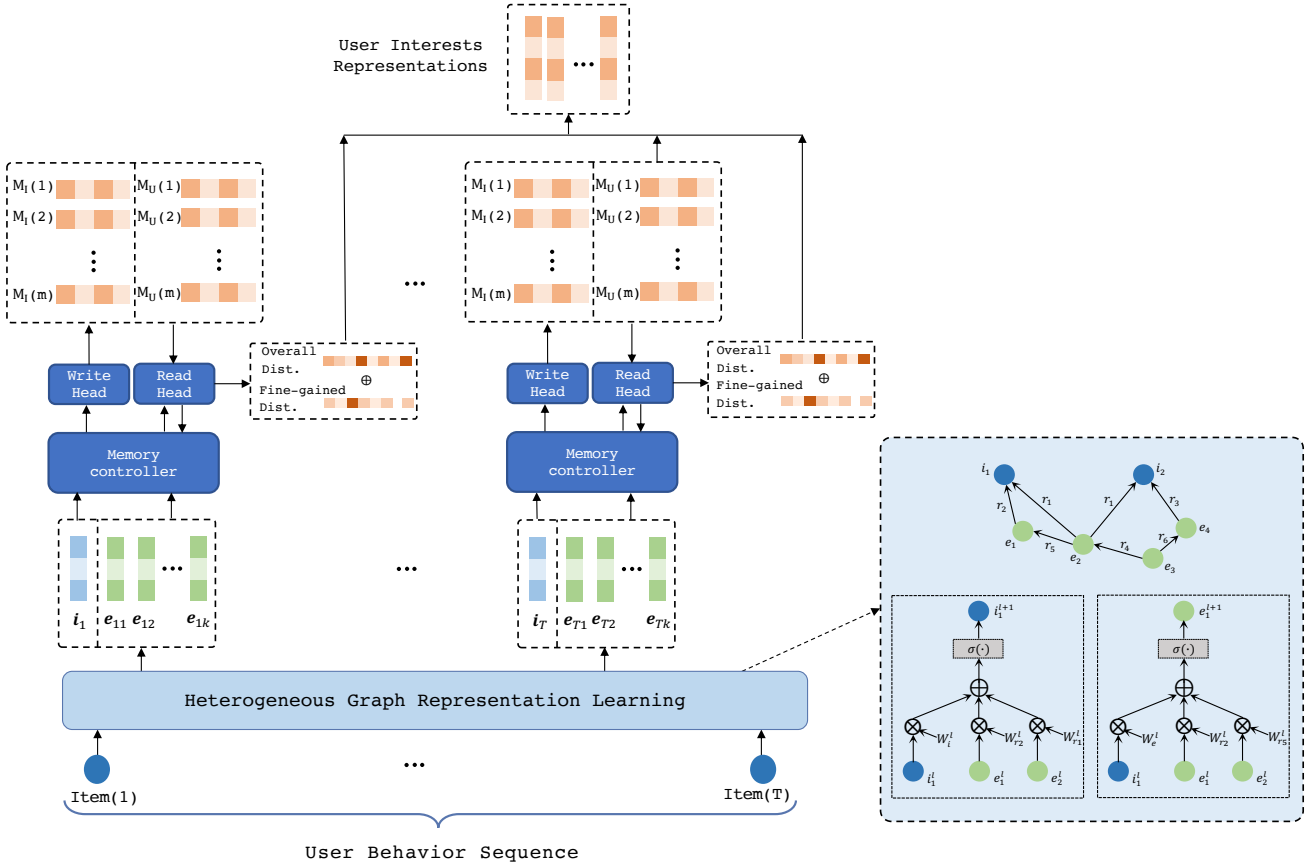


Figure 2: An overview of the proposed *KEMI* model. The input of our model is a user behavior sequence, which contains a list of items. The items and their associated knowledge sub-graphs are fed into the heterogeneous graph representation learning module. Items and their associated entities are transformed to fused representations, with which the user's overall and fine-grained interest distribution and the user's behavior sequence information are respectively learned by a dual memory network. Finally, the user's multi-interest representation can be obtained.

where $\mathbf{V}_u = \{\mathbf{v}_1^u, \mathbf{v}_2^u, ..., \mathbf{v}_m^u\} \in \mathbb{R}^{m \times d}$, d the dimensionality, m the number of user interest representation vectors. Notations are summarized in Table 1.

3.2 Graph Representation Learning

Since knowledge graphs have different types of features and various types of relations and items and knowledge entities may also have different types of features. Inspired by HGNNs [26, 39] and JKNet [37], we leverage heterogeneous graph neural networks and JKNet to learn unified representations of item and entities. The whole framework can be formalized as:

$$\mathbf{h}_v = JK(\mathbf{h}_v^0, ..., \mathbf{h}_v^n) \tag{2}$$

where $\mathbf{h}_v$ is the final representation for v (an item or an entity), $\mathbf{h}_v^0$ means v's initial node feature. $\mathbf{h}_v^{l+1}$ $(0 < l + 1 <= n)$ is the hidden state of v in the $(l+1)$-th layer, which can be combined by the l-th layer representations of v and its neighbors:

$$\mathbf{h}_v^{l+1} = U_l(\mathbf{h}_v^l, \mathbf{m}_v^{l+1}) \tag{3}$$

where U_l is a l-th layer combining function, and $\mathbf{m}_v^{l+1}$ can be got by aggregating the messages of v's neighbors N_v. We explain the most important modules in detail.

3.2.1 Encoding Node Heterogeneous Features. We have two kinds of node features: item content features and entity features. Item content features come from powerful pre-trained language models so that it can capture adequate text information, while entity features are generated by knowledge graph embedding methods to seize topology information. Specifically, We utilize Sentence-Bert [25] to generate a fixed-dimension vector for each item, e.g. $\mathbf{h}_v^{0B}$ for item v. All items and knowledge graphs are then combined and fed to a TransE [2] model to get fixed-dimension vectors for both items and entities, , e.g. $\mathbf{h}_v^{0T}$ for item v. Initial vectors of items and entities will be different as:

$$\mathbf{h}_v^0 = \begin{cases} \mathbf{h}_v^{0T} & v \in entities \\ concat(\mathbf{h}_v^{0B}, \mathbf{h}_v^{0T}) & v \in items \end{cases} \tag{4}$$

3.2.2 Aggregating Heterogeneous Neighbors. Items and Entities have different initial vector spaces. We transform them into a unified space utilizing RGCN [26]:

$$\mathbf{m}_v^{l+1} = \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_r^{(l)} \mathbf{h}_j^{(l)} \tag{5}$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in R$. $c_{i,r}$ is a problem-specific normalization constant.

3.2.3 Updating Nodes Representations. After we get neighbor representations, we can update node features $\mathbf{h}_v^{l+1}$ in equation 3 by

$$\mathbf{h}_v^{l+1} = \sigma\left(\mathbf{m}_v^{l+1} + \mathbf{W}_0^l \mathbf{h}_i^l\right) \tag{6}$$

where W_0^l is the l-th dense layer of self-loop relations, and σ is a nonlinear activation function.

3.2.4 Jumping Knowledge. We then apply jumping-knowledge operations (JK) to merge representations of different layers (see equation 2). JK can be embodied as concatenation, max-pooling, and so on. We choose concatenation in our paper.

3.3 Modeling User Interests with Dual Memory Network

Memory networks have been shown to be effective in modeling users' diverse interests [24] and users' sequential behaviors [12]. We propose a dual memory network to better learn fine-grained user interests from his/her historical behavior sequence. We propose to use an interest memory network M_I to learn the interest distribution and a user memory network M_U to learn the user's personal interest representation from his/her historical behavior sequence. M_I and M_U have the same number of memory slots, to keep the interests of the two memory networks consistent. The details of the modules are as follows:

3.3.1 Interest Memory Network. As the example described in Figure 1, a single item often contains multiple types of interests. Thus we want to not only learn the overall interests of an item from the item itself but also the fine-grained interests of it with the help of the associated knowledge entities.

Our model follows the classic Neural Turing Machine (NTM) model [9], which captures and stores information from sequential data with a memory network. In the time step of t, parameter of memory is indicated as M_t , which consists of m memory slots, representing m different interests. Two basic operations for NTM are memory read and memory write, which interact with memory through a controller. At time step t, we have $(\mathbf{i}_t, \{\mathbf{e}_t^1, \mathbf{e}_t^2, ...\mathbf{e}_t^k\})$, representing the vector learned above of the item and entities associated to it.

Read Interest Distributions: Given a user's behavior embedding vector at time step t, the controller generates a read key $\mathbf{k}_t$ to address memory. For item embedding vector:

$$\mathbf{k}_i^t = \mathbf{i}_t \mathbf{K} \tag{7}$$

where $\mathbf{K}$ is the key matrix in controller, it traverses all memory slots, generates a weight vector, represents the overall interest distribution of an item:

$$\mathbf{w}_l^t(i) = \frac{exp(f(\mathbf{k}_i^t, M_I^t(l)))}{\sum_j^m exp(f(\mathbf{k}_i^t, M_I^t(j)))}, \quad for\ l = 1, 2, ..., m \tag{8}$$

where,

$$f(\mathbf{k}_i^t, M_I^t(j)) = \frac{\mathbf{k}_i^{tT} M_I^t(j)}{\left\|\mathbf{k}_i^t\right\| \left\|M_I^t(j)\right\|} \tag{9}$$

For the entity embedding vectors, we have similar operations, generating a weight vector representing the fine-grained interest distribution of an item:

$$\mathbf{k}_e^t = (\textstyle\sum_j^k \mathbf{e}_j^t)\mathbf{K} \tag{10}$$

$$\mathbf{w}_l^t(e) = \frac{exp(f(\mathbf{k}_e^t, M_I^t(l)))}{\sum_j^m exp(f(\mathbf{k}_e^t, M_I^t(j)))}, \quad for\ l = 1, 2, ..., m \tag{11}$$

where,

$$f(\mathbf{k}_e^t, M_I^t(j)) = \frac{\mathbf{k}_e^{tT} M_I^t(j)}{\left\|\mathbf{k}_e^t\right\| \left\|M_I^t(j)\right\|} \tag{12}$$

The interest distribution for item i_t for user u is:

$$\mathbf{w}_u^t = \alpha * \mathbf{w}^t(i) + (1 - \alpha) * \mathbf{w}^t(e) \tag{13}$$

where α is the weight factor of overall interest distribution. $\alpha = 1$ means we only use items to model users' interests and $\alpha = 0$ means we only use knowledge entities to model users' interests.

Write Interest Memory Slots: We update the Interest Memory Slots after the whole user behavior sequence to keep the interests stable. Two additional keys of add vector $\mathbf{a}_i$ and erase vector $\mathbf{e}_i$ are also generated from the controller, which controls the update of memory.

$$\mathbf{a}_i = (\sum_{t=1}^{T} \mathbf{i}_t + \sum_{t=1}^{T} \sum_{j=1}^{k} \mathbf{e}_t^j)\mathbf{A} \tag{14}$$

$$\mathbf{e}_i = (\sum_{t=1}^{T} \mathbf{i}_t + \sum_{t=1}^{T} \sum_{j=1}^{k} \mathbf{e}_t^j)\mathbf{E} \tag{15}$$

$$\mathbf{M}_I = (1 - \mathbf{E}_I) \odot \mathbf{M}_I + \mathbf{A}_I \tag{16}$$

where $\mathbf{A}$ is add matrix in controller, $\mathbf{E}$ is erase matrix in controller, $\mathbf{E}_I = \mathbf{w}_u \otimes \mathbf{e}_i$, $\mathbf{A}_I = \mathbf{w}_u \otimes \mathbf{a}_i$, and $\odot$ and $\otimes$ means dot product and outer product respectively.

3.3.2 User Memory Network. Memory networks have been shown to be effective for modeling behavioral sequences due to their memory capabilities. In this paper, We propose to use a memory network with the same structure of interest memory network to model the user's behavior sequence, in this manner, we can maintain the same interest information with the interest memory network at each memory channel, and we can learn the user's behavior sequence information at the same time.

Write User Memory Slots: At time step t, we update user memory network as:

$$\mathbf{a}_u^t = (\mathbf{i}_t + \sum_{j=1}^{k} \mathbf{e}_t^j)\mathbf{A} \tag{17}$$

$$\mathbf{e}_u^t = (\mathbf{i}_t + \sum_{j=1}^{k} \mathbf{e}_t^j)\mathbf{E} \tag{18}$$

$$\mathbf{M}_U^t = (1 - \mathbf{E}_U^t) \odot \mathbf{M}_U^{t-1} + \mathbf{A}_U^t \tag{19}$$

where $\mathbf{E}_U^t = \mathbf{w}_u^t \otimes \mathbf{e}_u^t$ and $\mathbf{A}_U^t = \mathbf{w}_u^t \otimes \mathbf{a}_u^t$.

3.3.3 User Interests Representation. After the updating of user memory network, we learn the user's behavior sequence information from the user memory network, and we can get the user's interest distribution from the interest memory network, so we combine the two memory networks to get the user's multi-interests representation:

$$\mathbf{V}_u = (\sum_{t=1}^{T} \mathbf{w}_u^t)\mathbf{M}_U^T \tag{20}$$

3.4 Learning

3.4.1 constrained interest loss. Through the interest memory network, we can get the overall interest $\mathbf{w}^t(i)$ and the fine-grained interest $\mathbf{w}^t(e)$ of the item at time step t. We believe that these two interest distributions should be related. We can learn the fine-grained interests of the item through the knowledge entities associated to it, this interest distribution largely enriches the details of the overall interest, and the overall interest distribution of the item encoded from the whole text can also monitor the importance of the fine-grained interests. Therefore, We believe that these two interest distributions play a complementary role. So we add a constrained interest loss to the final loss function to help better learn the interest distributions:

$$\mathcal{L}_i = KL(\mathbf{w}(i)|\mathbf{w}(e)) = \sum_{j=1}^{m}\sum_{t=1}^{T} w(i)_j^t log\frac{w(i)_j^t}{w(e)_j^t} \tag{21}$$

3.4.2 Model training. After computing the interest embeddings from user behaviors through the dual memory network, we get multiple user representations, and different user representation is used to represent different user interest. When training the model, we don't require users to be interested in an item at each interest, so we only need the user representation of the corresponding interest $\mathbf{v}_u^k$ to be close to the item representation $\mathbf{i}^k$ at interest index k. The distance loss is minimized between the user representation and the ground-truth item:

$$\mathcal{L}_d = dist(\mathbf{v}_u^k, \mathbf{i}^k) \tag{22}$$

where dist(·) is the predefined distance function, and we use cosine similarity in our work.

The overall objective function for training KEMI is:

$$\mathcal{L} = \mathcal{L}_d + \lambda_1 \mathcal{L}_i + \lambda_2 \|\Phi\|_2^2 \tag{23}$$

where $\|\Phi\|_2^2$ is the regularization term to prevent over-fitting, and λ_1, λ_2 are control parameters. We obtain the values of λ_1 and λ_2 through tuning experiments.

Our training process is formalized as Alg. 1

Algorithm 1: Training Procedure of KEMI

Input: user behavior sequence H_u; knowledge graph $\mathcal{G}$;
Output: user representation function f_{user};

```
Randomly initialize all parameters ;
while not converge do
    for each user u in mini-batch U_i do
        for item i at time step t do
            Learn unified representations of item and entities:
              (i_t, {e_t^1, e_t^2, ...e_t^k}). ▷ Section 3.2 ;
            Read interest memory network M_I, get overall and
              fine-grained interest distributions: w^t(i) and w^t(e).
              ▷ Section 3.3.1 ;
            Write User Memory Network M_U. ▷ Section 3.3.2 ;
        Write Interest Memory Network M_I. ▷ Section 3.3.1;
        Get u's representation V_u. ▷ Section 3.3.3 ;
        Calculate overall objective function L. ▷ Section 3.4 ;
    Back-propagate the gradients and update.
```

3.5 Online Serving

After training, we can get the KEMI network f_{user}. At serving time, users' behavior sequences 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 in total by an approximate nearest neighbor approach. These items holding the highest similarities with user's representation vectors are retrieved and constitute the final set of candidate items for the candidate generation stage of recommender system. Please note that, when a user has new actions, it will alter his behavior sequence as well as the corresponding user representation vectors, thus KEMI enables real-time personalization for the matching stage.

As the size of item pool is far bigger than top-N, it is necessary to figure out how many neighborhood items to take from each user representation and how to merge them to get the best top-N items. A direct strategy is to partition the candidate set equally so that the same number of candidates are generated from each user representation. However, it may not optimize the overall quality of the whole candidates since different user interests could have distinct importance.

Since the relevance between an item and a user representation is reflected by their distance, it is quite natural to extend this principle as our matching strategy. Particularly, we first retrieve the N-nearest neighbors of each user representation $\mathbf{v}_i^k$ at interest index k. Then, for each item $\mathbf{i}^k$, we associate it with its distance towards $\mathbf{v}_i^k$. Next, we merge the neighborhood items of all user representations and sort them with the ascending order of their associate distances. Finally, the items with the top-N shortest distances are selected to be our candidates.

4 EXPERIMENTS

4.1 Experiment Settings

4.1.1 ***Datasets.*** We evaluate all compared models on the following realistic and knowledge-rich datasets:

	Microsoft News	Dianping Feed
# . users	1,000,000	100,000
# . items	161,013	915,493
#. interactions	24,155,470	1,518,490
#. associated entities per item	17.2	15.8
#. words per article	639	137

Table 2: Statistics of the two realistic datasets.

	Wikidata	Private KG
#. entities	3,275,149	2,502,554
#. triples	31,963,632	19,467,000
#. relations	1,091	47

Table 3: Statistics of the knowledge graphs.

Microsoft News: [34] is the largest open-source English news dataset for public research purpose, constructed from the user click logs of Microsoft News[1]. The knowledge graph of this dataset is Wikidata[2].

Dianping Feed: is collected from the online feed service of Dianping[3]. Dianping is one of the biggest UGC (User-Generated Content) websites in China, where users can create and interact with a large number of contents. We collect 100,000 users' click logs of 6 weeks from July 26, 2021 to September 05, 2021 from Shanghai. We build the dataset in the same manner as the Microsoft News dataset. The knowledge graph of this dataset is our private knowledge graph.

The basic statistics of the two datasets and knowledge graphs are shown in Table 2 and Table 3.

4.1.2 *Compared Models.*

- **Most Popular** is a traditional recommendation method that recommends the most popular items to users.
- **Youtube DNN** [6] is one of the most successful deep learning models for industrial recommender systems, where users' historical behaviors are encoded by deep networks and aggregated via mean-pooling.
- **MIND** [17] is a popular method where multiple channels are deployed. It designs a multi-interest extractor layer based on the capsule routing mechanism, which is applicable for clustering past behaviors and extracting diverse interests.
- **Comirec** [4] is a recently proposed framework following MIND to extract diverse interests using dynamic routing and incorporate a controllable aggregation module to balance recommendation diversity and accuracy.
- **Octopus** [21] is a framework modeling multiple user interests via elastic archive network for candidate generation.
- **MIMN** [24] a recent representative work for the ranking stage of recommendation, using memory networks to capture user interests from long sequential behavior data. We modified the matching function to adapt to our candidate generation task.

[1]https://news.microsoft.com
[2]https://www.wikidata.org/wiki/Wikidata:Main_Page
[3]https://www.dianping.com/

To make a fair comparison, we use the same Sentence-Bert vectors and TransE vectors to initialize the item embeddings for the above baseline models.

4.1.3 *Evaluation Metrics.*

We use the following metrics to evaluate all models' performance which have been widely used in previous methods.

- **Hit Rate**, Hit Rate (HR) measures the percentage that recommended items contain at least one correct item interacted by the user in the top-k ranking lists.

$$HR@N = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \delta(\left|\hat{\mathcal{I}_{u,N}} \cap \mathcal{I}_u\right| > 0) \tag{24}$$

  where $\hat{\mathcal{I}_{u,N}}$ denotes the set of top-N recommended items for u, and $\mathcal{I}_u$ is the set of testing items for user u, $\delta(\cdot)$ is the indicator function.
- **Recall**, we adopt per-user average in this paper.

$$Recall@N = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\left|\hat{\mathcal{I}_{u,N}} \cap \mathcal{I}_u\right|}{|\mathcal{I}_u|} \tag{25}$$

4.1.4 *Hyper-parameter Setup and Reproducibility.*

The number of dimensions d for embeddings is set to 128. The number of interest channels for multi-interest models is set to 10 for the dianping feed dataset and 20 for the microsoft news dataset. We use Adam optimizer [14] with learning rate lr = 0.002 for optimization, λ_1 is set to 0.1 and λ_2 is set to 0.00001 and α is set to 0.5. We release the source code at https://github.com/danyang-liu/KEMI.

4.2 Main Results

Table 4 summarizes the performance of KEMI as well as baselines on two datasets in terms of HR@N and Recall@N (N = 10, 50, 100). Our KEMI model outperforms all the baselines in terms of all the metrics, proving the effectiveness of our model combined with knowledge graphs for the multi-interest candidate generation task. We also have some observations:

First, on our dianping feed dataset, the multi-interest models like Octopus and MIND achieve better results than the single-interest models like Youtube DNN, using multiple user representation vectors is proved to be an effective way for modeling users' diverse interests as well as boosting recommendation accuracy.

Second, for the news dataset, some multi-interest models like MIND, Comirec are not as effective as the single-interest model in some metrics, like HR@10 and Recall@10. This may be caused by the unreasonable distribution of items in different interest channels when they generate candidates, thus when N is small, they don't perform well enough but Our KEMI and baseline like Octopus are better due to the usage of a good allocation strategy.

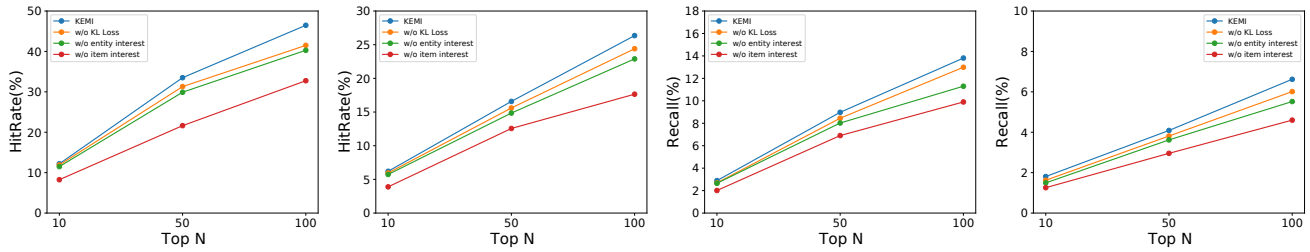
Third, Our model achieves the best results on both datasets in terms of all the metrics, demonstrating the effectiveness and consistency of our proposed KEMI model.

4.3 Ablation and Hyper-Parameter Study

In this section, we study the necessity of some key components and an important hyper-parameter: the number of interests m.

	Dianping Feed Dataset						Microsoft News Dataset					
	Metrics@10		Metrics@50		Metrics@100		Metrics@10		Metrics@50		Metrics@100	
	Hit Rate	Recall	Hit Rate	Recall	Hit Rate	Recall	Hit Rate	Recall	Hit Rate	Recall	Hit Rate	Recall
Most Popular	5.713	1.014	15.74	3.215	23.28	5.385	1.659	0.526	1.696	0.536	1.704	0.537
Youtube DNN	11.36	2.586	30.69	8.081	41.54	12.44	5.832	1.492	14.87	3.814	22.29	6.079
MIND	11.81	2.830	31.89	8.900	43.19	13.39	3.113	0.752	12.77	3.318	20.93	5.728
Comirec	11.44	2.742	31.47	8.905	42.27	13.28	3.753	0.941	13.57	3.489	22.22	6.160
Octopus	11.49	2.768	31.99	8.913	42.89	13.12	5.845	1.543	15.60	3.616	23.90	6.198
MIMN	11.28	2.601	30.98	8.426	42.35	12.90	5.032	1.454	14.31	3.732	21.50	5.833
KEMI	**12.19**	**2.889**	**33.50**	**8.976**	**46.47**	**13.81**	**6.198**	**1.808**	**16.58**	**4.088**	**26.35**	**6.621**

Table 4: Model performance on two datasets. Bolded numbers are the best performance of each column. The results are reported in percentage (%).



(a) HitRate on Feed Dataset **(b) HitRate on News Dataset** **(c) Recall on Feed Dataset** **(d) Recall on News Dataset**

Figure 3: Ablation study on removing the constrained interest loss, the overall interest weight or the fine-grained interest weight on two datasets.

Interests	News Title	Knowledge Entity
Interest 1	News1: Lloyd: Coming off bench at World Cup "rock bottom of my entire career" News2: Premier League player Power Rankings News3: Lukaku brace at Bologna fires Inter top, Roma beat Napoli to go third	Premier League Cristiano Ronaldo World Cup FC Barcelona
Interest 2	News1: Amazon CEO Jeff Bezos is no longer the richest person in the world News2: Starbucks gives $10 million to help boost small business in Chicago's neighborhoods News3: McDermott chief financial officer resigns following $1.9 billion loss	CFO stock SoftBank credict card
Interest 3	News1: Why macOS Catalina is breaking so many apps, and what to do about it News2: Pixel 4 and 4 XL hands-on: Google ditches fingerprint scanner for face unlock News3: Xbox's latest safety feature lets you filter out offensive trash talk	Xbox macOS Catalina Pixel 4 Apple

Figure 4: An example of news and knowledge entities related to the different interest channels of our KEMI model

4.3.1 Ablation Study. We study the necessity of key components, including the constrained interest loss (marked as w/o KL loss), the overall interest weight (marked as w/o item interest), and the fine-grained interest weight (marked as w/o entity interest). From figure 3 we can observe that removing any of these components will lead to a performance drop, which demonstrates the effectiveness of these components. Removing the overall interest weight causes the biggest performance drop, showing that the item itself contains the most user interest information.Removing knowledge information(w/o entity interest) will also lead to a big performance drop, this indicates with the help of knowledge entities, we can make a more comprehensive understanding of user interests. The constrained interest loss can help the overall interest and fine-grained interest learn better, so it is also helpful for the model performance.

4.3.2 Number of Interests. Table 5 illustrates the performance of our framework when the hyper-parameter m changes. $m = 1$ means we use a single vector to model user interests. The results show that using multiple interests achieves better results than using a single interest, thus illustrating the necessity of multi-interest modeling. We also observe that our model achieves the best results on the dianping feed dataset when $m = 10$ and on the news dataset when $m = 20$. We believe this is due to the relatively more diverse content of news, so more interest channels are needed to model user interests.

4.4 Case Study

To demonstrate the interpretability of our model on multi-interest candidate generation, we present some real examples on the news dataset. Different from the previous multi-interest models, we can explain not only at the item level but also at the entity level. As shown in the example in Figure 4, we select three interest channels with corresponding news and representative entities. Take interest 1 as an example, from the level of news, we can know that this is probably a sports-related interest. From the representative entity, we can more intuitively know that this is an interest in soccer.

		Microsoft News			Dianping Feed		
		N=10	N=50	N=100	N=10	N=50	N=100
m=1	HR@N	5.806	14.13	22.43	11.21	30.40	41.96
	Recall@N	1.518	3.725	6.002	2.667	8.142	12.48
m=5	HR@N	5.892	14.99	23.20	11.83	32.19	44.80
	Recall@N	1.653	3.816	6.190	2.878	8.951	13.38
m=10	HR@N	6.062	15.81	25.47	**12.19**	**33.50**	**46.47**
	Recall@N	1.775	3.901	6.472	**2.889**	**8.976**	**13.81**
m=20	HR@N	**6.198**	**16.58**	26.35	12.02	33.10	46.01
	Recall@N	**1.808**	**4.088**	**6.621**	2.841	8.953	13.46
m=30	HR@N	6.134	16.46	**26.48**	11.60	32.18	44.63
	Recall@N	1.769	3.957	6.608	2.841	8.880	43.69

Table 5: Model performance w.r.t. different number of interests, the results are reported in percentage (%). Bolded numbers are the best performance.

# users	# items	#interactions
2,659,562	2,123,510	160,409,918

Table 6: Statistics of the industrial large dataset

With the help of knowledge graphs, we associate user interests with entities, making our explanations more straightforward and convincing.

4.5 Industrial Large Dataset Results

We also conduct experiments on an industrial large dataset. We collect all the users' behavior logs of 6 weeks from July 26, 2021 to September 05, 2021 of Shanghai in our online system. The statistics of the industrial large dataset are shown in Table 6. We conduct offline experiments between our framework and all the state-of-the-art recommendation methods. The experimental results demonstrate that our KEMI model can improve Recall@100 and HR@100 with 3.73% and 6.25% compared to the best baseline results respectively.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose KEMI, a novel multi-channel model which comprehensively captures users' diverse interests with the help of knowledge graphs for candidate generation tasks in recommender systems. We propose to use HGNNs to learn the unified representations of items and knowledge entities with multiple types of relations, and propose a novel dual memory network structure to learn the overall and fine-grained interests from users' behavior sequences. Extensive experimental studies are carried out with both industrial and publicly available datasets, where KEMI achieves substantial improvements over state-of-the-art baseline methods, meanwhile, our method can provide a good explanation for each interest channel with the help of knowledge graphs. In the future, we will study how to leverage knowledge graphs to adaptively assign interest channels to different users, and study the effect of knowledge graphs enhanced multi-interest candidate generation on improving the diversity and satisfaction of users in recommender systems.

REFERENCES

[1] Ioannis Antonellis, Hector Garcia-Molina, and Chi-Chao Chang. 2008. Simrank++ query rewriting through link analysis of the clickgraph (poster). In *Proceedings of the 17th international conference on World Wide Web*. 1177–1178.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*. 151–161.

[4] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.

[5] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[7] Qiang Cui, Shu Wu, Qiang Liu, Wen Zhong, and Liang Wang. 2018. MV-RNN: A multi-view recurrent neural network for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* 32, 2 (2018), 317–331.

[8] 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*. 278–288.

[9] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).

[10] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538, 7626 (2016), 471–476.

[11] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

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

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 535–544.

[16] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of massive data sets*. Cambridge university press.

[17] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.

[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[19] Danyang Liu, Jianxun Lian, Shiyin Wang, Ying Qiao, Jiun-Hung Chen, Guangzhong Sun, and Xing Xie. 2020. KRED: Knowledge-aware document representation for news recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 200–209.

[20] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[21] Zheng Liu, Jianxun Lian, Junhan Yang, Defu Lian, and Xing Xie. 2020. Octopus: Comprehensive and Elastic User Representation for the Generation of Recommendation Candidates. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 289–298.

[22] Zheng Liu, Yu Xing, Fangzhao Wu, Mingxiao An, and Xing Xie. 2019. Hi-Fi Ark: Deep User Representation via High-Fidelity Archive Network.. In *IJCAI*. 3059–3065.

[23] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation.

In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 5045–5052.

[24] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2671–2679.

[25] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[26] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[27] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*. 101–110.

[28] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 417–426.

[29] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*. 1835–1844.

[30] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 968–977.

[31] Jingdong Wang, Naiyan Wang, You Jia, Jian Li, Gang Zeng, Hongbin Zha, and Xian-Sheng Hua. 2013. Trinary-projection trees for approximate nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 36, 2 (2013), 388–403.

[32] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[33] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

[34] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.

[35] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. 2018. Socialgcn: An efficient graph convolutional network based model for social recommendation. *arXiv preprint arXiv:1811.02815* (2018).

[36] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 285–294.

[37] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*. PMLR, 5453–5462.

[38] Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, and Philip S Yu. 2021. ConsisRec: Enhancing GNN for Social Recommendation via Consistent Neighbor Aggregation. *arXiv preprint arXiv:2105.02254* (2021).

[39] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.

[40] Wei Vivian Zhang, Xiaofei He, Benjamin Rey, and Rosie Jones. 2007. Query rewriting using active learning for sponsored search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 853–854.

[41] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192* (2018).

[42] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[43] 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*. 1059–1068.

[44] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. 2019. Collaborative metric learning with memory network for multi-relational recommender systems. *arXiv preprint arXiv:1906.09882* (2019).