Hierarchical Preference Hash Network for News Recommendation

Jianyong DUAN†,††(a), Member, Liangcai LI†,††(b), Mei ZHANG†(c), and Hao WANG†,††(d), Nonmembers

SUMMARY
Personalized news recommendation is becoming increasingly important for online news platforms to help users alleviate information overload and improve news reading experience. A key problem in news recommendation is learning accurate user representations to capture their interest. However, most existing news recommendation methods usually learn user representation only from their interacted historical news, while ignoring the clustering features among users. Here we proposed a hierarchical user preference hash network to enhance the representation of users' interest. In the hash part, a series of buckets are generated based on users' historical interactions. Users with similar preferences are assigned into the same buckets automatically. We also learn representations of users from their browsed news in history part. And then, a Route Attention is adopted to combine these two parts (history vector and hash vector) and get the more informative user preference vector. As for news representation, a modified transformer with category embedding is exploited to build news semantic representation. By comparing the hierarchical hash network with multiple news recommendation methods and conducting various experiments on the Microsoft News Dataset (MIND) validate the effectiveness of our approach on news recommendation.

key words: news recommendation, modified transformer, hierarchical hash network

1. Introduction

Online news services such as MSN News and Google News which aggregate news from various sources and distribute them to a large population of users [1]. A overwhelming amount of newly-sprung news is generated every day, making it difficult for users to seek for their interested news [2]. Thus, personalized news recommendation is very important for online news platforms to help users find their interested contents [3].

There are two major problems in news recommendation: how to represent news articles which have rich textual content and how to learn better representations of users [4]. In recent years, many deep learning based methods have been proposed for news recommendation [5]–[8]. For example, Wang et al. proposed DKN [1], which incorporates information from knowledge graph for better news recommendation. Specifically, DKN formed news representations from their titles and entities via convolutional neural network (CNN). Then they utilized an attention network to select important clicked news for user representations. However, CNN cannot capture the long-distance contexts of words and this method cannot model the relatedness between browsed news. Wu et al. proposed NRMS [7], which learns news representation from news title by using multi-head self-attention to model the interactions between words and learn representations of users from their browsed history by using multi-head self-attention to capture their relatedness. However, most of them build user representations only from users’ historically clicked news, but when the historical behaviors of users are sparse, it is difficult for them to learn accurate user representations.

Our work is motivated by several observations. First, the interactions between words in news title are important for understanding the news. For example, in this news title “The Toronto Raptors beat the Golden State Warriors to win the 2019 NBA championship”, the word “Raptors” has strong relatedness with “Toronto” and it also has semantic interactions with “championship”. Also, different words may have different importance in representing news, the word “championship” is more informative than “2019”. Second, different news articles browsed by the same user may also have different importance in representing this user. For example, a user browsed ten news before, eight news are related to sports and the other two are weather and pets. So, it can be inferred that this user may be a sports enthusiast, and the last two news articles are less informative than the previous eight. Third, there goes a saying that birds of a feather flock together, people in groups, Users who have similar preference can be clustered into one group. If a fresh user has very few click records, the user’s preference vector which is learned from his browsing history is not enough. The solution we’ve come up with is that users with rich browsing history in the same group can provide reference information for the preference representation of the users with less interaction history.

In this paper, we proposed a Hierarchical Preference Hash Network (HPHN) to enhance the representation of user’s interest. First, we utilize the transformer [9] to build news semantic representations from textual content. In this way, the multi-head self-attention network encodes word dependency in titles at both short and long distance and the additive attention to learn more accurate news representation by selecting important words. We also add the category embedding of news since it contains important classification
information. Second, learning user interest representation from previously clicked news articles has similarity with learning document representation from its sentences [10]. So, in history part, we learn representations of users from their browsed news by using multi-head self-attention to capture the relatedness between the news, and apply additive attention to learn more informative user representations by selecting important news. Then we further enhance user representation by mapping it into some more informative one’s via a hierarchical hash network and aggregating those two parts to form the final user representation. Finally, recommendation is made by taking the dot product between user and news representations. We conduct extensive experiments on a large real-world dataset. The improved performances over a set of well-known baselines validate the effectiveness of our approach.

2. Related Work

News recommendation has attracted increasing attentions from both datamining and natural language processing fields. It is critical for news recommendation methods to learn accurate news and user representations. Many previous news recommendation methods rely on feature engineering to represent news articles and user preference [11], [12]. For example, Li et al. [11] represented news articles using their URLs and categories, and represented users using their demographics, geographic information and behavior categories inferred from their consumption records on Yahoo!. However, these methods heavily rely on manual feature engineering, which needs massive domain knowledge to craft. In addition, the contexts information of news is not incorporated, which is important for understanding the semantic meanings of news and learning representations of news and users. In recent years, many deep learning based methods have been proposed for news recommendation [5]–[8]. For instance, Wu et al. [6] proposed an attentive multi-view learning framework to represent news articles from different news texts such as title, body and category. They used an attention model to infer the interest of users from their clicked news articles by selecting informative ones. Though effective in extracting information from textual content, these methods cannot learn accurate user representations especially when the historical behaviors of user are sparse. Different from these methods, our approach exploits both context meaning and clustering features among users.

3. Our Approach

In this section, we will introduce our Hierarchical user Preference Hash Network (HPHN) approach illustrated in Fig. 1, which consists of two major components, i.e., a news encoder to learn representation of news and a user encoder which includes the history part and hash part to learn the preference representation of users.

3.1 Transformer for News Representation

The news encoder we apply the modified transformer for news representation was motivated by Vaswani et al. [9] While the news titles are usually concise and clear, we just simplify the transformer with single layer of multi-head self-attention to avoid the degradation of performance caused by excessive parameters. This part contains three layers. The first bottom layer is the word embedding, which is used to convert a news title from a sequence of words into a sequence of low-dimensional embedding vectors. Denote a news title with M words \{w_1, w_2, \ldots, w_M\}, then through this layer it is converted into the embedded vector sequence \{e_1, e_2, \ldots, e_M\}. The following layer is a word-level multi-head self-attention network. Interactions between words are important for learning news representations. For instance, in the title “Pelicans’ Redick has best Halloween costume”, the interaction between “Redick” and “costume” helps understand the title. Moreover, a word may relate to more than one word in the title. For example, the word “Redick” has interactions with both words “best” and “costume”. Thus, we employ the multi-head self-attention to form contextual word representations. The representation of the \(i_{th}\) word learned by the \(k_{th}\) attention head is computed as:

\[
a_{i,j}^{k} = \frac{\exp(e_i^T W_k e_j)}{\sum_{m=1}^{M} \exp(e_i^T W_k e_m)}
\]

\[
h_i^k = W_k^s \left( \sum_{j=1}^{M} a_{i,j}^{k} e_j \right)
\]

Where \(W_k^s\) and \(W_k^e\) are the projection parameters in the \(k_{th}\) self-attention head, \(e\) is the word embedding vector and \(a_{i,j}^{k}\) indicates the relative importance of the interaction between the \(i_{th}\) and \(j_{th}\) words. The multi-head representation \(h_i^k\) of the \(i_{th}\) word is the concatenation of the representations produced by \(h\) separate self-attention heads, i.e., \(h_i = [h_i^1; h_i^2; \ldots; h_i^h]\). The total count of \(h_i\) is equal to the number of words \(M\). To mitigate overfitting, we add dropout [13] after self-attention.

The third layer is an additive word attention network to model relative importance of different words and aggregate them into title representations. For instance, the word “best” is more informative than “has” for understanding the news. Thus, we propose to use attention mechanism to select important words in news titles for learning more informative news representations. The attention weight \(\beta_i^t\) of the \(i_{th}\) word is computed as:

\[
\beta_i^t = \frac{\exp(q_i^t \tanh(U_w x_h + u_w))}{\sum_{j=1}^{M} \exp(q_j^t \tanh(U_w x_h + u_w))}
\]

Where \(q_i^t, U_w \) and \(u_w\) are trainable parameters in the word attention network. The news title representation \(v_t\) is then calculated as:
Fig. 1 The framework of our HPHN approach

Since category information of user clicked news may also reveal their preferences, we model news topics via an embedding matrix. Denote the output of this embedding matrix as $v_c$, then the final representation of the news is the concatenation of the title vector and the category vector, i.e., $v = [v_t; v_c]$.

3.2 User Preference Representation

In order to learn better user preference representation, we designed a history part to learn the representations of users from their browsed news and a hash part to further improve user representation from some more informative users.

3.2.1 History Part

The history part is used to learn the representations of users from their browsed news. While learning user preference representation from previously clicked news articles has similarity with learning news title representation from its words. Here we still use the modified transformer to learn history vector, which contains two layers. The first one is a news-level multi-head self-attention network. Usually, news articles browsed by the same user may have some relatedness and a news article may interact with multiple news articles browsed by the same user. Thus, we propose to apply multi-head self-attention to enhance the representations of news by capturing their interactions. The representation of the $i_{th}$ news learned by the $k_{th}$ attention head is formulated as follows:

$$
\gamma_{i,j}^k = \frac{\exp\left(n_i^T Q_k^* n_j\right)}{\sum_{m=1}^{N} \exp\left(n_i^T Q_k^* n_m\right)}
$$

(5)

Where $Q_k^*$ and $V_k^*$ are parameters of the $k_{th}$ news self-attention, $n_i$ and $n_j$ are the browsed news vectors, and $\gamma_{i,j}^k$ represents the relative importance of the interaction between the $i_{th}$ and $j_{th}$ news. The multi-head representation of the $i_{th}$ news is the concatenation of the representations output by $n$ separate self-attention heads, i.e., $h_i = [h_1^i; h_2^i; \ldots; h_n^i]$. The total count of $h_i$ is equal to the number of browsed news $N$.

The second layer is an additive news attention network. Different news may have different informativeness in representing users. For example, if a user is a technophile, the news about science and technology is more informative than the news about the weather today since it is usually browsed by massive users at the same day. Thus we propose to apply the additive attention mechanism to select important news to learn more informative user representation. The attention weight of the $i_{th}$ news is computed as:

$$
\eta_i^k = \frac{\exp\left(q_i^T \tanh (V_n \times h_i + v_n)\right)}{\sum_{j=1}^{N} \exp\left(q_i^T \tanh (V_n \times h_j + v_n)\right)}
$$

(7)

Where $V_n$, $v_n$ and $q_n$ are parameters in the attention network and $N$ is the number of the browsed news. The representation of history part is the weighted summation of the representation of the news browsed by this user, which is calculated as:

$$
u_i^b = \sum_{j=1}^{N} \eta_i^k h_j
$$

(8)

3.2.2 Hash Part

Only with the history vector from the user browsed history is not enough to capture the user preference[14], as some
fresh users do not have sufficient historical feedback (their history vectors provide limited information). So we design a hash part (shown in the red dashed box) to get users’ additional preference features based on previous warm users who browsed with similar news to the target user. Warm users are users who have browsed a lot of news, and the system knows their preference well. The function of hash part is to store and calculate the preferences of all users.

In the hash part, users with similar preferences share the same buckets to store their preference. In this way, when there come other users, these warm users are able to act as a reference. Specially, it will be very helpful when there are rare-interaction (cold-start) users. Besides, as the user browses more and more news, his/her hash vectors will also dynamically change.

The hash part mainly consists of three components, namely hierarchical hash, hash buckets, and top-k weighted sum. The workflow is shown in Fig. 1. Next we will introduce the hash part from bottom to top.

• Hierarchical Hash Network

In order to find useful buckets that store the preferences of users who have interacted with similar news, we design a hierarchical hash network to better calculate a special user’s preference vector. With the browsed history vector $u^h$ of a user as input, hierarchical hash works like a fully-connected tree where the children nodes and finally leaves (various buckets). Let $n^j_i$ denotes the $j$-th node in layer $i$. Each node $n^j_i$ has a decision vector $n^j_i$, and it inherits a relevance weight $r^j_i$ from its parent node. Suppose that node $n^j_i$ has $c$ children nodes $(n^{j+1}_{i+1}, ..., n^{j+1}_{i+c})$, then it allocates the weight $r^j_i$ to them by:

$$r^j_{i+1} = r^j_i \frac{\exp{(u^h n^{j+1}_{i+1})}}{\sum_{y=0}^{c-1} \exp{(u^h n^{j+1}_{i+y})}}, \quad x = 0, ..., c - 1$$  \hspace{1cm} (9)

where $r^j_{i+1}$ is the relevance weights of the $x$-th child. Note that relevance weights of all nodes on the same level have a total of 1, e.g., $\sum r^j_i = 1$, and initially, on the root $r^j_1 = 1$. The number of leaves equals to the number of buckets, and finally, the relevance weights of leaves are the weights of buckets corresponding to the current user history. As there are often thousands and even more representative anchor users in large-scale data, it is a little hard to differentiate these buckets in one step. So we proposed a hierarchical hash network to spread the weight and achieve the goal step by step. It automatically clusters the users at different levels. We use three layers in our experiments. More or fewer layers are also tried, but this setting provides a more stable result.

• Hash buckets

Suppose we have $h$ ($h \ll n$) buckets $B = \{b_1, ..., b_h\}$ (simplify, $h = 4$ in Fig. 1 as an example). The buckets are used to store some users with similar preferences. At first, we bind each bucket $b_i$ with a warm user $u_{b_i}$. We call these selected warm users “Anchor User” and their final hash vectors directly comes from the corresponding hash buckets $b_i$, i.e.:

$$u^h_j = \begin{cases} u_{b_i}, & \text{if } u_j \text{ is an anchor user} \\ \text{hash}(u^h_j), & \text{otherwise} \end{cases}$$  \hspace{1cm} (10)

where hash(·) is the hash part which can be regarded as a function that takes the user history vector as input for the hierarchical hash network and output the user hash vector. We bind each bucket an anchor user for mainly the following reasons:

1. These buckets provide references for users to enrich their preference representation. We need sufficient information stored in each bucket. So we selected for their informative preferences.
2. Each bucket only bind one anchor user. That’s because warm users have a large number of historical interactions, and their preferences are usually more significantly different than those of other users. If we use one bucket to store multiple warm users affects the accuracy of modeling preferences of modeling preferences of these users.
3. Obtaining the vector of anchor user is based on Collaborative Filtering (CF) methods that maps a user to a vector and only builds the matrix for anchor user. We know that CF works better in dense data than sparse data. Conducting CF among anchor users is more efficient and effective for their rich browsing history. In this way, our model remains the CF process between the anchor users for better recommendations.

These warm users or anchor users can be predefined in many ways, whether by some clustering methods or manually selecting. For simplicity, we manually choose the users with the most interactions in our implementation here.

For other users, our model uses a hierarchical hash to find the top-k related buckets, and weight-sum these bucket vectors to form the preference hash vector. The difference to the anchor user is that each of them does not have a specific fixed bucket to store their preferences but share with others. And their buckets may dynamically change as they interact with more items.

• Top-K Weighted Sum

In order to form the user preference hash vector, here we only consider the most related K buckets instead of the weight-sum of all the buckets. The reason is that it takes much time and space to weight-sum such a large number of bucket vectors. It is also unnecessary because the weights usually have long-tail small values (which contribute little, even considering them will significantly increase the cost of back-propagation). Let $r_j$ from the last layer in the hierarchical hash denote the relevance weight of bucket $b_j$. When making predictions, we calculate the user preference hash vector of user $u$ as follows:

$$u^h = \frac{1}{\sum_{j=1}^{K} r_j b_j} \sum_{j=1}^{K} r_j b_j$$  \hspace{1cm} (11)
where \( t_1, \ldots, t_K \) are the indexes of the largest \( K \) relevance weights and thus \( a_{t_1}, \ldots, a_{t_K} \) are the most relevant \( K \) bucket vectors. \( K \) is usually much smaller than the number of hash buckets (\( K \ll h \)).

However, we find that the module tends to visit only a small number of buckets if we always keep considering the top-\( K \) relevance buckets during the training process, that will cause our model performs badly. We think the reason is that some well-trained bucket vectors get higher weights than others. If the module always focuses on the vectors with the largest weights during training, they will be trained to store the preferences of too many users. Thus, many other buckets are never visited, and their corresponding decision nodes are rarely trained. These buckets have no chance to be updated and keep irrelevant. To solve this problem, we choose random \( K \) buckets when training each example:

\[
u^h = \frac{1}{\sum_{j=1}^{K} r_{t_j} b_{t_j}} \sum_{j=1}^{K} r_{t_j} b_{t_j} \tag{12}\]

where \( t_1, \ldots, t_K \) are \( K \) randomly selected indexes. In this way, for each training example, our approach explores some random buckets for the target user. The module is then trained to increase the weights of relevant buckets and decrease those irrelevant. The procedure can be regarded as an exploration of trying different buckets. Each sample visits different buckets in different epochs. The hash part takes the user history vectors as input, so better hashing a user also improves the users with similar preferences. Finally, the module is expected to calculate the relevance for all the buckets properly.

### 3.2.3 Route Attention for Final User Representation

To generate the final user preference vector, HPHN uses an attention network, named as Route Attention, to combine the vector of \( u^h \) from the history part and \( u^b \) from the hash part. The user history vector \( u^h \) records the preferences from the news the user has interacted with. The user hash vector \( u^b \) provides preferences of other informative users who have read with similar news. For a user with lots of historical records, \( u^h \) may be enough to capture his/her preferences. But for a user who has rare-interaction with the news, it is better to utilize the \( u^h \) so as to explicitly reference the history of other users. It is reasonable and essential to use the Route Attention to adjust sources of the user preference vector dynamically.

We combine the two vectors to form the final user representation \( u \) in the following steps:

\[
e^b_u = h^T f(Wu^b + b), \quad e^h_u = h^T f(Wu^h + b)
\]

\[
w^e_u = \frac{\exp(e^b_u)}{\exp(e^b_u) + \exp(e^h_u)} = 1 - w^h_u
\]

\[
u = w^h_u u^h + w^b_u u^b
\]

where \( W \in \mathbb{R}^{cd}, b \in \mathbb{R}, h \in \mathbb{R}^r \) are the parameters of attention network and \( r \) denotes the hidden layer size of the attention network and \( d \) is the user vector size. \( f \) is the non-linear activation function and we use \textit{relu} in our implementation.

### 3.3 Click Predictor

The click predictor is used to predict the probability of a user clicking a candidate news. The click probability score \( \hat{y} \) is computed by the inner product of the final user representation vector and the news representation vector, i.e., \( \hat{y} = u^T v \), \( u \) is the final user representation vector and \( v \) is the representation of a candidate news as we mentioned before. We also explored other kinds of scoring methods such as Multi-layer perceptron, but dot product shows the best performance and efficiency.

### 3.4 Model Training

Motivated by (Huang et al., 2013) [15], we use negative sampling techniques for model training. For each news browsed by a user which is regarded as a positive sample, we randomly sample \( K \) news which are shown in the same impression but not clicked by the user, regarded as negative samples (exposed but didn’t click). Denote the click probability score of positive and the \( K \) negative news as \( \hat{y}_i \) and \( [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K] \) respectively. We apply maximum likelihood method to minimize the log-likelihood on the positive class:

\[
\mathcal{L} = -\sum_i \log \left( \frac{\exp(\hat{y}_i)}{\exp(\hat{y}_i) + \sum_{j=1}^{K} \exp(\hat{y}_j)} \right) \tag{14}
\]

where \( \hat{y}^+_i \) is the label of the \( i \)-th positive sample and \( \hat{y}^-_{i,j} \) is the predicted label of the associated \( j \)-th negative sample. The models are evaluated according to the rank of the positive simple in these \( K+1 \) candidate news.

### 4. Experiments

#### 4.1 Dataset and Experimental Settings

In this paper, we conduct our experiments on the Microsoft News Dataset (MIND)\(^{1}\) [10], which is a large-scale dataset for news recommendation research. It was collected from the user behavior logs of Microsoft for those who had at least 5 news click records during 6 weeks from October 12 to November 22, 2019. It contains about 160k English news articles and more than 15 million impression logs generated by 1 million users. Every news article contains rich textual content including title, abstract, body, category and entities. Each impression log contains the click events, non-clicked events and historical news click behaviors of this user before this impression. An impression log records the news articles displayed to a user when he/she visits the news website homepage at a specific time, and their click behaviors on these news articles. The statistics of the MIND dataset are shown in Table 1:

\(^{1}\)The MIND dataset is public available at https://msnews.github.io
In our experiments, the word embeddings are 300-dimensional and initialized by the Glove [16] embedding. The self-attention networks have 16 heads, and the output of each head is 16-dimensional. The dimension of the additive attention query vectors is 200 and the category embedding is set to 100. As for hash part, the number of buckets (anchor users) $h$ is 1024 and we select $K=128$ of them to form the hash preference. For the hierarchical hash, we use a 3-layer structure with [1-64-1024]. And both the history vector and hash vector are 300. Following (Wu et al., 2019) [17], the negative sampling ratio is 4. Adam (Kingma and Ba, 2014) [18] is used for model optimization. We apply 20% dropout to the word embeddings to mitigate overfitting. The batch size is 64. The learning rate is 0.001, and early-stopping is conducted according to the performance on the validation set. We conducted all experiments on a machine with Xeon(R) CPU E5-2640 v4 CPUs and one GeForce TITAN Xp GPU. For evaluation, we independently repeated each experiment 10 times and reported average results in terms of AUC, MRR, nDCG@5 and nDCG@10.

4.2 Performance Evaluation

In this section, we will evaluate the performance of our approach by comparing it with some baseline methods, including general recommendation methods and news-specific recommendation methods. Some of the implementations can be found in Microsoft Recommenders open source repository.

- General recommendation methods:
  - LibFM [19]: a classic recommendation method based on factorization machine. Besides the user ID and news ID, we also use the TF-IDF features extracted from news texts as the additional features to represent users and candidate news.
  - DSSM [20]: deep structured semantic model, which uses tri-gram hashes and multiple feed-forward neural networks for query-document matching. We also use the TF-IDF features extracted from previous clicked news as query, and those from candidate news as document.
  - Wide&Deep [21]: a two-channel neural recommendation method, which has a wide linear transformation channel and a deep neural network channel. Wide&Deep is verified as one of the best deep neural recommendation models in many different scenarios.
  - DeepFM [22]: a widely used method that combines factorization machines and deep neural networks. The same content features of users and candidate news are fed to both components.

- News recommendation methods:
  - DKN [1]: a deep knowledge-aware network for news recommendation. It uses CNN to learn news representations from news titles with word embeddings, entity embeddings and entities context embedding. Then learns user representations based on the similarity between candidate news and previously clicked news.
  - NAML [6]: a method with attentive multi-view learning to incorporate different kinds of news information into the representations of news articles.
  - LSTUR [5]: a news recommendation method with long- and short-term user interest. It models short-term user interest from recently clicked news with GRU and models long-term user interest from the whole click history.
  - NRMS [7]: a neural news recommendation method which uses multi-head self-attention to learn news representations from the words in news text and learn user representations from previously clicked news articles.

(4) HPHN*: our approach, which uses a modified transformer to learn news representation from news title and category. And then learn user representation from the combination of the history part and the hash part. We apply a modified transformer to learn the history vector and use a hierarchical hash network to form the hash vector.

The results of these methods are summarized in Table 2: From Table 2, we derive several observations: First, news-specific recommendation methods such as DKN, LSTUR and NRMS perform better than general recommendation methods like LibFM, Wide&Deep and DeepFM. That is because in these neural news recommendation methods learned the representation of news articles and user preference in an end-to-end manner, while in the general news recommendation methods they usually represented using manual features. Second, among the news specific recommendation methods, the methods which exploit the relatedness between news (e.g., NRMS and HPHN) can outperform other methods. This may be because the news

| #News | #Users | #Impression |
|-------|--------|-------------|
| 161,015 | 1,000,000 | 15,777,377 |

### Table 2: The performance of different methods on news recommendation.

| Method   | AUC   | MRR   | nDCG@5 | nDCG@10 |
|----------|-------|-------|--------|---------|
| LibFM    | 59.93 | 28.23 | 30.05  | 35.74   |
| DSSM     | 64.31 | 30.47 | 33.86  | 38.61   |
| Wide&Deep| 62.16 | 29.31 | 31.38  | 37.12   |
| DeepFM   | 60.3  | 28.19 | 30.02  | 35.71   |
| DKN      | 64.6  | 31.32 | 33.84  | 39.48   |
| NAML     | 66.86 | 32.49 | 35.24  | 40.91   |
| LSTUR    | 67.73 | 32.77 | 35.59  | 41.34   |
| NRMS     | 67.76 | 33.05 | 35.94  | 41.63   |
| HPHN     | 68.09 | 33.37 | 36.33  | 42.08   |

*https://github.com/microsoft/recommenders
browsed by the same user usually have relatedness, and utilizes attention mechanism to capture the news relatedness is useful for understanding these news and modeling user interests. Third, our approach which combines the history part and hash part performs better than all baseline methods. That is because our method considers the rare-interaction users and enhances their user representation by hashing them to some rich-interaction users. The result validates the effectiveness of hash part to enrich the user representation by attentively exploiting the clustering features among users.

4.3 Pre-Trained Language Models

In this part, we explorer whether the quality of news representation can be further improved by the pre-trained language models such as BERT [23], which have achieved huge success in many different NLP tasks. We applied bert-base† model and roberta-base†† model to the news representation of our model, the results are summarized in Fig. 2.

We find that by replacing the original word embedding module with the pre-trained BERT architecture model, the performance of our model can be improved. It shows that BERT model pre-trained on large-scale corpus can provide useful semantic information for news representation. We also find that the roberta-base performs better than the bert-base model, that’s because RoBERTa [24] has over 160GB of uncompressed text including CC-News, etc. Which is larger and more diverse than the original BERT. And it is also trained with dynamic masking, full-sentences without Next Sentence Prediction loss, large mini-batches and a larger byte-level Byte-Pair Encoding. Generally, all results validate that the pre-trained language models are very helpful for news representation.

4.4 Ablation Study

To better understand the influence of different parts in our approach, we conduct an ablation study MIND dataset. Results are shown in Table 3. HPHN without hash means that we learn the user representation only from their browsing history news. To compare hierarchical hash network with some clustering methods, we adopt K-means on the user history (HPHN_K-means) to form h clusters, and use the cluster-ID to represent the users.

Results in Table 3 show that without the hash part, only using the history to represent the user’s preference is not enough. HPHN with a Hash Part, achieves the best results, show the effectiveness of our methods. The reason is that for all users, the history part of HPHN dynamically retrieves the preferences related to the target news in the user history. And for each target user, the hash part helps find some related buckets storing the preferences of users interacted with similar news. Their history helps recommend to the target user, especially for those rare-interaction users. In this way, HPHN improves the performance of both rich-history and rare-interaction users. The results also show that even though K-means can replace the hash part, it performs worse than manual selected method, which shows the effectiveness of the hierarchical hash network to model similar user preferences.

4.5 Hyperparameter Analysis

We also study the performances when the hierarchical hash network with different numbers of hash buckets and different top numbers in the hashing process (or random exploration number in the training process). The results are shown in Fig. 3. (K is equal to 128 when changing h, and h is fixed to 1024 when changing K). h = 0 means there is no hash part and only use the history to represent the user’s preference. As shown in the figure, a small number of hash buckets even work worse than no hash buckets. We think the reason is that storing too many users’ preferences in a bucket vector makes the vectors overload and probably chaotic. As the number of hash buckets increases the model performance keeps improving. It is because these buckets

---

Table 3 Results of ablation study on MIND dataset.

| Model              | AUC   | MRR   | nDCG@5 | nDCG@10 |
|--------------------|-------|-------|--------|---------|
| HPHN without hash  | 67.81 | 32.52 | 35.87  | 41.35   |
| HPHN_K-means      | 67.84 | 32.63 | 35.89  | 41.42   |
| HPHN               | 68.09 | 33.37 | 36.33  | 42.08   |

---

†https://huggingface.co/transformers/model_doc/bert.html
††https://huggingface.co/transformers/model_doc/roberta.html
store different categories of users, and thus, more buckets differentiate more categories of users. These buckets help model the preferences of different users without enough historical information. The performance stops growing when the number of hash buckets is larger than thousands. The reason is that too many buckets increase the difficulty of the hash process, and a more powerful hash strategy is needed in such cases. Similarly, by exploring and considering more hash buckets (larger K), the performance grows because of more accurate preference modeling and stops increasing when the K is large enough. We finally use $h = 1024$ and $K = 128$ for both effectiveness and efficiency.

4.6 Case Study

To find out what the hierarchical hash network has learned in the buckets, we take some buckets as examples. Figure 4 is the top 3 news in the two buckets. We collect the browsed history of users in each bucket and rank the news according to how many users in the bucket have read the news. It is clear that after the hierarchical hash, the two buckets are different from each other and the news in each bucket is related. Big news in the bucket No.100 is all about politics including the Britain officially out of the European Union, etc. News in the bucket No.101 is all the big sports news in 2019 including Warriors vs. Raptors NBA finals, etc. The results show the effectiveness of our approach, which stores different kinds of preferences in different buckets.

5. Conclusions

In this paper we propose a hierarchical hash enhanced user representation learning architecture for news recommendation. The core of our approach is a modified transformer news encoder and a user encoder with history part and hash part. In the history part, we learn representations of users from their browsed news use multi-head self-attention. In the hash part, a series of buckets are generated based on users’ historical interactions. Users with similar preferences are assigned into the same buckets automatically. Finally, a Route Attention is adopted to combine these two parts (history vector and hash vector) and get the more informative user preference vector. Extensive experiments validate that it is an effective module that helps apply previously proposed models to large-scale real-world recommender systems. In the future, we consider improving our hierarchical hash network by designing better anchor user selection methods. It is important to consider the typicality of anchor users and how to place them in different buckets.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61972003, 61672040), and the North China University of Technology Startup Fund. We would also like to thank the anonymous reviewers for their helpful comments. We would like to thank the referees for their comments, which helped improve this paper considerably. The corresponding author is Hao Wang.

References

[1] H. Wang, F. Zhang, X. Xie, and M. Guo, “DKN: Deep Knowledge-Aware Network for News Recommendation,” Proc. 2018 World Wide Web Conference, pp.1835–1844, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018.
[2] S. Ge, C. Wu, F. Wu, T. Qi, and Y. Huang, “Graph Enhanced Representation Learning for News Recommendation,” Proc. Web Conference 2020, pp.2863–2869, Association for Computing Machinery, New York, NY, USA, 2020.
[3] W. IJntema, F. Goossen, F. Frasincar, and F. Hogenboom, “Ontology-based news recommendation,” Proc. 2010 EDBT/ICDT Workshops, EDBT’10, New York, NY, USA, Association for Computing Machinery, 2010.
[4] S. Okura, Y. Tagami, S. Ono, and A. Tajima, “Embedding-based news recommendation for millions of users,” the 23rd ACM SIGKDD International Conference, 2017.
[5] M. An, F. Wu, C. Wu, K. Zhang, and X. Xie, “Neural news recommendation with long- and short-term user representations,” The 57th Annual Meeting of the Association for Computational Linguistics, 2019.
[6] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, and X. Xie, “Neural news recommendation with attentive multi-view learning,” the 28th International Joint Conference on Artificial Intelligence, 2019.
[7] C. Wu, F. Wu, S. Ge, T. Qi, and X. Xie, “Neural news recommendation with multi-head self-attention,” Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.
[8] Q. Zhu, X. Zhou, Z. Song, J. Tan, and L. Guo, “Dan: Deep attention neural network for news recommendation,” Proc. AAAI Conference on Artificial Intelligence, vol.33, no.01, pp.5973–5980, July 2019.
[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in neural information processing systems, pp.5998–6008, 2017.
[10] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, and M. Zhou, “MIND: A large-scale dataset for news recommendation,” Proc. 58th Annual Meeting of the Association for Computational Linguistics, Online, pp.3597–3606, Association for Computational Linguistics, July 2020.
[11] J. Liu, P. Dolan, and E.R. Pedersen, “Personalized news recommendation based on click behavior,” International Conference on Intelligent User Interfaces, 2010.
[12] J.-W. Son, A.-Y. Kim, and S.-B. Park, “A location-based news article recommendation with explicit localized semantic analysis,” International Acm Sigir Conference on Research Development in Informa-
Jianyong Duan is a Professor, born in 1978.10. He graduated from Department of computer science, Shanghai Jiao Tong University by 2007.12. His major research field including natural language processing and information retrieval.

Liangcai Li is a master in College of Informatics, North China University of Technology. His major research field is News Recommendation and Knowledge Graph.

Mei Zhang is an associate professor, born in 1980.11, from North China University of Technology. Her major research field focus on knowledge map.

Hao Wang received the Ph.D. degree in Computer Application Technology from Tsinghua University in 2013. He is now an assistant professor in College of Informatics, North China University of Technology. He has published more than 20 papers in international conferences and journals. His research interests include machine learning and data analysis.

Jianyong Duan is a Professor, born in 1978.10. He graduated from Department of computer science, Shanghai Jiao Tong University by 2007.12. His major research field including natural language processing and information retrieval.