Proceedings Paper:
Zhang, L., Shi, L., Zhao, J. et al. (4 more authors) (2022) A GNN-based multi-task learning framework for personalized video search. In: WSDM '22: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, 21-25 Feb 2022, Virtual Event (Phoenix, AZ, USA). Association for Computing Machinery , pp. 1386-1394. ISBN 9781450391320

https://doi.org/10.1145/3488560.3498507

A GNN-based Multi-task Learning Framework for Personalized Video Search

Li Zhang[1], Lei Shi[2], Jiashu Zhao[3], Juan Yang[2], Tianshu Lyu[2], Dawei Yin[2], and Haiping Lu[1]
[1]Department of Computer Science, University of Sheffield, Sheffield, United Kingdom
[2]Baidu Inc., Beijing, China
[3]Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, Canada
[1]{lzhang72, h.lu}@sheffield.ac.uk, [2]{shilei24, yangjuan03, lyutianshu}@baidu.com, yindawei@.acm.org
[3]jzhao@wlu.ca

ABSTRACT

Watching online videos has become more and more popular and users tend to watch videos based on their personal tastes and preferences. Providing a customized ranking list to maximize the user's satisfaction has become increasingly important for online video platforms. Existing personalized search methods (PSMs) train their models with user feedback information (e.g. clicks). However, we identified that such feedback signals may indicate attractiveness but not necessarily indicate relevance in video search. Besides, the click data and user historical information are usually too sparse to train a good PSM, which is different from the conventional Web search containing users' rich historical information. To address these concerns, in this paper we propose a multi-task graph neural network architecture for personalized video search (MGNN-PVS) that can jointly model user's click behaviour and the relevance between queries and videos. To relieve the sparsity problem and learn better representation for users, queries and videos, we develop an efficient and novel GNN architecture based on neighborhood sampling and hierarchical aggregation strategy by leveraging their different hops of neighbors in the user-query and query-document click graph. Extensive experiments on a major commercial video search engine show that our model significantly outperforms state-of-the-art PSMs, which illustrates the effectiveness of our proposed framework.

KEYWORDS

Personalized Video Search, Graph Neural Networks, Multi-Task Learning, User-query Graph, Query-document Click Graph.

ACM Reference Format:
Li Zhang[1], Lei Shi[2], Jiashu Zhao[3], Juan Yang[2], Tianshu Lyu[2], Dawei Yin[2], and Haiping Lu[1]. 2022. A GNN-based Multi-task Learning Framework for Personalized Video Search. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/10.1145/3488560.3498507

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/10.1145/3488560.3498507

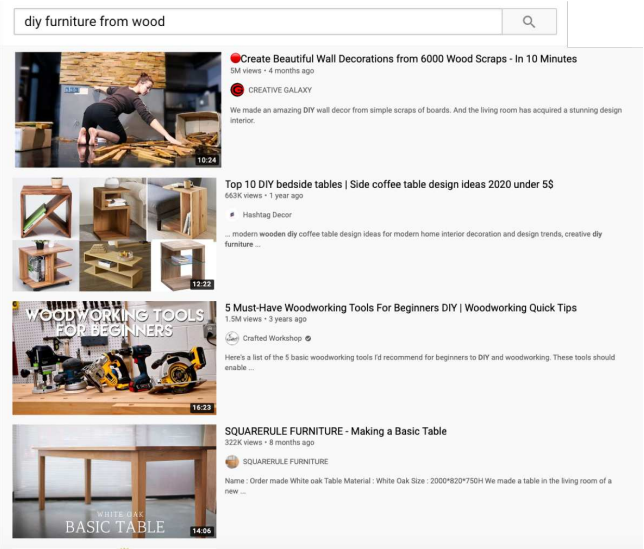


Figure 1: Motivation of multi-task learning. Example top ranked results for the query "diy furniture from wood": the first video is for "decoration with wood scraps", and the third video is for "woodworking tools" are not relevant to the query, but may still be interesting to the user.

1 INTRODUCTION

With the popularity of smart phones and advancement of communication technologies, people can easily record and edit videos. Everyday, billions of new videos are created, shared and watched [35]. Video search provides a convenient entry point for customers to browse and watch videos from numerous videos. Different from Web search, video search is mainly for entertainment and users' preferences could be very diverse because of their different backgrounds. For example, when issuing the query "*Welcome to New York*", a music fan may want to find the music video from *Taylor Swift*, while a film fan may want to watch the film directed by *Abel Ferrara*. Similar to many other vertical search areas, there is a strong need to return personalized search results to different users for the same query in video search.

The user click information is often used to train personalized search models (PSMs), while the click signal in video search may not necessarily indicate relevancy between the query and video [39]. Statistics show that clicks of irrelevant videos can be as high as 30% of all clicks in our system. This may be due to the characteristics of

video search, including: users tend to have more time when they are browsing videos; videos serve more entertainment purpose than non-video documents; videos contain richer information than non-video documents. Therefore, users can be easily attracted to interesting but irrelevant videos. Figure 1 shows a list of top videos returned from YouTube given the query "diy furniture from wood". We can see that the first video (showing the wall decoration) and the third video (showing the woodworking tools) are not very relevant to the query. Given this list of videos, however, a user who is currently doing some home improvement projects, may also be interested in furnishing the walls and clicking on the first video, or be interested in checking out the tools and clicking on the third video. Therefore, we propose that both customized search results and query-document relevance should be considered in personalized video search, which have rarely been exploited by current PSMs.

The most common paradigm of existing PSMs is to apply deep learning to learn the semantic similarity of query-document pair and personalize the search results by considering users' information, i.e., users' location, meta information, social connections [14, 15, 34] or search history [1, 8, 20, 40]. While, the data in the video search is usually too sparse to train a good PSM [13, 30]. Especially when queries are short and vague, it is more difficult for PSMs to learn accurate representations. Additional information such as the relationship among users, queries in user-query, click graphs respectively can provide rich information beyond textual and video content [2, 5, 21], which have rarely been exploited by current PSMs.

While intuitively useful to integrate similar users and click graph information into personalized video search, and graph neural networks (GNNs) [10, 18] can be applied to model the topological information of graph datasets. However, two unique challenges arise in achieving this goal in our scenario. (i) The graphs are heterogeneous and contain millions of nodes and edges. How to design an efficient GNN architecture for real-world graphs is the first obstacle we need to overcome. (ii) The user-query graph and query-document click graph in the real industry system are stored as user-query and query-document pairs, rather than entire graphs used in typical GNNs, which could prevent the message passing between different hops of neighbors. For a given user, both the local information (issued queries) and higher-order neighbors, such as similar users from the second-hop neighbors (user-query-user) in the user-query graph are important for the user's representation learning. Hence, how to jointly capture the local as well as the higher-order neighborhood information remains a significant challenge.

In light of the aforementioned motivations and challenges, we propose a GNN-based multi-task learning framework for personalized video search where two bipartite graphs: user-query graph and query-document [1] click graph are integrated into the learning process, in addition to the semantic representations learned from text (query and video title) and video content with BERR [6] and Two-Stream Inflated 3D ConvNet (I3D) [4]. To efficiently utilize the graph information, we perform the graph convolution by sampling fixed-size neighbors from graphs and alleviate the need to operate

[1]In our paper, document equals to video.

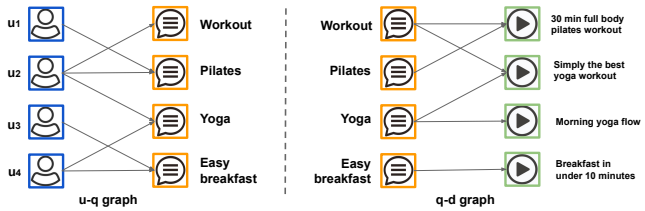


Figure 2: User-query graph and query-document click graph. In user-query graph, nodes are users and queries, and edges mean users issued queries. Query-document graph contains two types of nodes: queries and documents and links mean clicks for query-document pairs by any user.

GNN on the entire graph during training. To utilize different hops of neighbors, we propose a hierarchical GNN architecture to simultaneously capture both local and higher-order interactions among nodes. It learns user representations from their issued queries (users' first-hop neighbors), neighboring users (user's second-hop neighbors) in the user-query graph. Query representations are learned from clicked videos and neighboring queries in the query-document click graph. Document representations are learned from their associated queries and neighboring videos in the query-document click graph. Considering the heterogeneity of the used information, we design a hop-specific transformation strategy, which enables nodes to treat different hops of neighbors differently.

In summary, our main contributions are summarized as follows:

- We design an efficient GNN-based multi-task learning framework for real industry personalized video search. We utilize two bipartite graphs: user-query and query-document graphs to enrich the representation for users, queries and videos. To the best of our knowledge, this is the first attempt to apply graph information and GNN for personalized video search.
- In real industry system, we identified that the click signal may indicate attractiveness but not necessarily indicate relevance. Different from other PSMs trained only by click label, our model also considers the relevance between queries and videos.
- We conduct extensive experiments on a large-scale real dataset obtained from a well-known video search platform. Experimental results show that our proposed model can significantly outperform most state-of-the-art PSMs.

2 METHODOLOGY

In this section, we introduce our GNN-based multi-task learning framework for personalized video search, which mainly consists of two key part: 1) semantic representation learning that focuses on text representation learning (query and video title) and video representation learning; 2) graph representation learning that utilizes the hierarchical GNN architecture to leverage user-query graph and query-document graph to learn better user, query and video representations.

Table 1: Key Notations and Explanations.

Notations	Explanation
$\mathcal{G}_{uq}$	User-query graph
$\mathcal{G}_{qd}$	Query-document click graph
$\mathbf{U}$	User ID embedding matrix
$\mathbf{Q}$	Query ID embedding matrix
$\mathbf{D}$	Document ID embedding matrix
$\mathbf{q}_{q\_t}$	Query q_q's text embedding
$\mathbf{d}_{d\_t}$	Video d_d's text embedding
$\mathbf{d}_{d\_v}$	Video d_d's video embedding
$\mathcal{N}_{u_u\_q}$	User' first-hop neighbors in $\mathcal{G}_{uq}$
$\mathcal{N}_{u_u\_u}$	User' second-hop neighbors in $\mathcal{G}_{uq}$
$\mathcal{N}_{q_q\_d}$	Queries' first-hop neighbors in $\mathcal{G}_{qd}$
$\mathcal{N}_{q_q\_q}$	Queries' second-hop neighbors in $\mathcal{G}_{qd}$
$\mathcal{N}_{d_d\_d}$	Video's second-hop neighbors in $\mathcal{G}_{qd}$

2.1 Preliminaries

For the personalized video search task, we first formulate our problem as follows. When a user u issues a query q, the video search engine is required to retrieve the most relevant videos as a ranking list. Through re-ranking the unpersonalized list for different users according to their interests, backgrounds, the video search engine finally provides a personalized ranking list to each individual user.

Considering the sparsity of users' search history, vague and short queries, we leverage extra information from the click-through data to learn their better representations. The click-through data contains both the user search behaviours and user click-through behaviors, thus we utilize these information to construct two bipartite graphs: user-query graph $\mathcal{G}_{uq}$ and query-document graph $\mathcal{G}_{qd}$, as shown in Fig. 2. In addition to the text and video information, our goal is to leverage graph information $\mathcal{G}_{uq}$ and $\mathcal{G}_{qd}$ to learn high-quality embeddings. Then, for a given triple (*user, query, video*), the corresponding representations are fed into a neural network for the final click score and query-video relevance score estimation.

2.2 Semantic Representation Learning

For a given triple <u_u, q_q, d_d>, we firstly utilize text (query and video title) and video information to learn the semantic representations of q_q and d_d.

Inspired by the great success achieved by the large-scale pre-trained transformer-based language models, such as BERT [6], we design a BERT_based Query-Title Matching Model (QTMM) to obtain the embeddings of queries and video titles, as shown in Fig. 3. The input of the model includes three components: a *query*, a *positive title*, and a *negative title*, where *positive title* is the title of a clicked video given the query q_q and *negative title* is chosen randomly from the unclicked videos for q_q. A special token [*CLS*] is attached at the beginning of each input component, which can aggregate the sequence information to generate embeddings during learning. Then, three multi-layer Transformer BERT with shared parameters are adopted to capture the contextual information in the text and generate the embeddings of *query*, *positive title* and *negative title* ($\mathbf{q}_{q\_t}$, $\mathbf{d}_{d\_t}$, $\mathbf{d}'_{d\_t}$). We train QTMM with the triplet loss [22] as following:

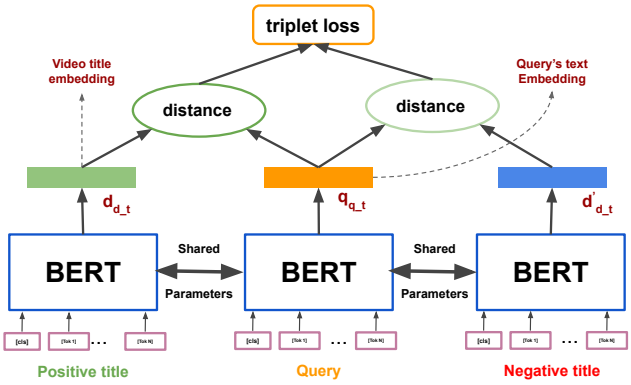


Figure 3: Query-Title Matching Model: train BERT with the triplet loss to get query text embedding $\mathbf{q}_{q\_t}$ and video title embedding $\mathbf{d}_{d\_t}$.

$$\mathcal{L}_t = max(d_{qp} - d_{qn} + margin, 0), \tag{1}$$

where d_{qp} is the euclidean distance of $\mathbf{q}_{q\_t}$, $\mathbf{d}_{d\_t}$ and d_{qn} is the distance of $\mathbf{q}_{q\_t}$, $\mathbf{d}'_{d\_t}$.

To learn the video representation $\mathbf{d}_{d\_v}$, we apply the similar strategy as in the text representation learning. Given the triple <*video, positive query, negative query*>, we feed the query to BERT and video to Two-Stream Inflated 3D ConvNet (I3D) [4], a popular model for video representation learning and train them with the triplet loss. To learn better semantic representations of queries and videos, both QTMM and I3D are trained on billions of the construct triples.

2.3 Graph Representation Learning

Beyond text and video information, we aim to apply GNNs to leverage graph information to alleviate the sparsity issue, as shown in Fig. 4. There are mainly two steps: 1) neighborhood sampling which samples fixed-size neighbors (from different hops in a graph) for a given node; 2) hierarchical aggregation that utilizes different hops of neighborhood information to enrich a given node's representation learning.

2.3.1 ***Neighborhood Sampling.*** GNN consists of two key steps: neighborhood aggregation and feature transformation. So, we start from how we define the neighborhood $\mathcal{N}_i$ in the real industry graphs, an important innovation of our approach [37].

For example, users' information demand can be clearly revealed by their issued queries (first-hop neighbors from $\mathcal{G}_{uq}$). Besides, the relationship among users (second-hop neighbors from $\mathcal{G}_{uq}$ who issued the similar set of queries) should not be ignored, which can be extremely helpful to overcome the sparsity problem of users' history. Thus both the first and second-hops of neighbors should be considered in the neighborhood aggregations process.

Conventional GNNs can leverage k-hop neighborhood information by stacking k GCN layers or perform random walks on the graph [10, 18, 19, 33, 36]. However, the user-query graph and query-document click graph in the real industry system are stored

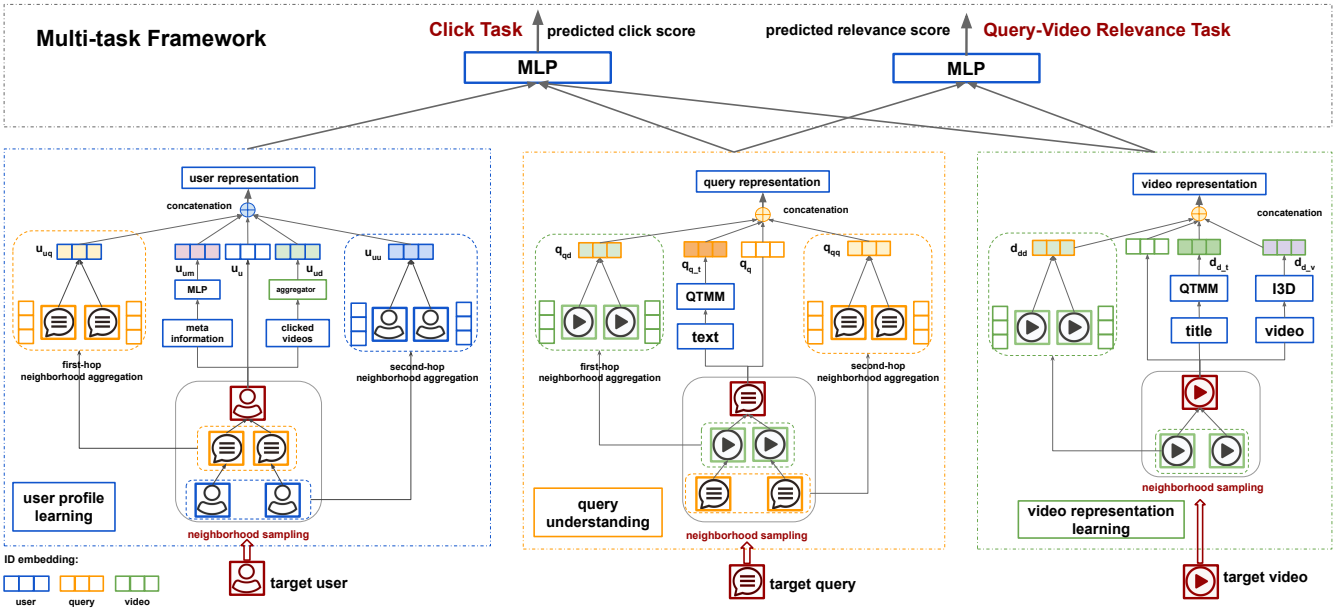


Figure 4: An illustration of our GNN-based multi-task framework. Given the triple <u_u, q_q, d_d>, we first apply the QTMM and I3D to learn the semantic representations of q_q and d_d. Then, we sample fixed-size neighbors for u_u, q_q, d_d from the u-q and q-d graphs and leverage the graph information with the proposed hierarchical GNN architecture simultaneously capturing both local and higher-order interactions among nodes to enhance their representation. Finally, we combine representations learned from text, video and graph for the click task and query-video relevance task for personalized video search.

as user-query and query-document pairs, rather than the entire graphs used in typical GNNs, which prevents the message passing between different hops of neighbors and the utilization of random-walk strategy. Besides, some hot queries and documents have tons of neighbors. Considering the memory and efficiency problem, we sample a fixed-size neighbors for each user, query and video from their first-hop (user-query pairs) and second-hop ((user-user pairs) neighborhood. For u_u, the sampled neighbors are issued queries (first-hop neighbors) $\mathcal{N}_{u_u\_q} = [q_{u1}, q_{u2}, ..., q_{uK}]$ and similar users $\mathcal{N}_{u_u\_u} = [u_{u1}, u_{u2}, ..., u_{uK}]$. Considering similar queries and clicked videos for a given query in the query-document graph can be exploited to enrich the current query and provide more search context to help disambiguation, especially for short and ambiguous queries, we sample K clicked videos $\mathcal{N}_{q_q-d} = [d_{q1}, d_{q2}, ..., d_{qK}]$ and similar queries $\mathcal{N}_{q_q-d} = [q_{q1}, q_{q2}, ..., q_{qK}]$ from $\mathcal{G}_{qd}$. Videos sharing many co-clicked queries should also be close in the vector space and we sample a fixed-size videos $\mathcal{N}_{d_d-d} = [d_{d1}, d_{d2}, ..., d_{dK}]$ from d_d's second-hop neighbors (clicked videos with same queries).

2.3.2 ***Hierarchical Aggregation.*** After getting sampled neighbors for users, queries and videos, a natural idea is to aggregate their neighborhood information to enrich their representations.

The conventional neighborhood aggregation in GNNs is

$$\mathbf{h}_i^{'} = f_a(\mathbf{h}_j, j \in \mathcal{N}_i), \tag{2}$$

where f_a is a predefined aggregation function (aggregator) and $\mathcal{N}_i$ is the neighborhood. In feature transformation stage, the central node $\mathbf{h}_i$ first combines with $\mathbf{h}_i^{'}$, followed by a linear mapping or MLPs to get its new representation.

Different from conventional GNNs, the sampled neighbors ($\mathcal{N}_i$) contain both homogeneous and heterogeneous neighbors for users and queries. Such as $\mathcal{N}_{u_u\_q}$ and $\mathcal{N}_{u_u\_u}$ are heterogeneous and homogeneous neighbors for u_u respectively. A naive approach is to ignore the node/edge types and treat them as in a homogeneous graph. [2] This, apparently, is suboptimal since different types of neighbors have different traits and their embeddings should fall in different feature spaces [9, 36]. Thus we design a hierarchical aggregation strategy and apply different aggregation functions to aggregate a given node's first-hop and second-hop neighbors respectively.

Take user u_u in the user-query graph as an example, we apply two different aggregation functions: f_{uq} f_{uu} to learn user's representations by leveraging its first-hop (queries) and second-hop (similar users) neighborhood information respectively, as shown:

$$\mathbf{u}_{uq} = f_{uq}(\mathbf{q}_{ui}, ui \in \mathcal{N}_{u_u\_q}). \tag{3}$$

$$\mathbf{u}_{uu} = f_{uu}(\mathbf{u}_{ui}, ui \in \mathcal{N}_{u_u\_u}), \tag{4}$$

where f_{uq} and f_{uu} are query-type and user-type aggregators that focus on aggregating homogeneous and heterogeneous information respectively. A natural follow-up question is how to design the aggregation function.

[2] A graph with one type of nodes and edges

Ideally, the aggregation function would be symmetric (i.e., invariant to permutations of its inputs), which ensures our model can be applied to arbitrarily ordered neighbors [10, 32, 38]. [3] Take f_{uq} as an example, candidate aggregation functions can be sum (f_{uq_s}), mean (f_{uq_m}), maxpooling, ($f_{uq_{mp}}$) [4] and attention ($f_{uq_{att}}$) aggregation [26, 27] as shown:

$$\mathbf{u}_{uq} = f_{uq_{att}}(\mathbf{q}_{ui}) = \sum_{i=1}^{K} \alpha_{ni}\mathbf{q}_{ui}(ui \in \mathcal{N}_{u_u\_q}), \tag{5}$$

where α_{ni} can be learned from

$$\alpha_{ni} = \frac{exp(LeakyReLU(\mathbf{W}_u[\mathbf{u}_u \parallel \mathbf{q}_{ui}] + \mathbf{b}_u))}{\sum_{d=1}^{K} exp(LeakyReLU(\mathbf{W}_u[\mathbf{u}_u \parallel \mathbf{q}_{ud}] + \mathbf{b}_u)}, \tag{6}$$

where $\mathbf{W}_u$ and $\mathbf{b}_u$ are trainable parameters in the attention network, $\parallel$ denotes concatenation.

Similarly, for a given query q_q, we apply the hierarchical aggregation strategy and utilize different aggregation functions f_{qd} and f_{qq} to aggregate its first-hop neighbors $\mathcal{N}_{q_{q\_d}}$ (clicked videos) and second-hop neighbors $\mathcal{N}_{q_{q\_q}}$ (similar queries) in the query-document graph to obtain $\mathbf{q}_{qq}$ and $\mathbf{q}_{qd}$ respectively. This can provide more search context to help disambiguation, especially for short and ambiguous queries.

Videos sharing many co-clicked queries should also be close in the vector space. For the given video d_d, We apply f_{dd} as the aggregator to aggregate its second-hop neighbors $\mathcal{N}_{d_{d\_d}}$ as following:

$$\mathbf{d}_{dd} = f_{dd}(\mathbf{d}_{di}, \mathcal{N}_{d_{d\_d}}). \tag{7}$$

2.4 Incorporating User Meta Information

Besides the user-query graph information, we also have users' additional features and search history information. To better characterize users and retrieve personalized search results, we augment our model with additional user features which are represented in a multi-field multi-hot encoding form. Each field contains multiple discrete categorical features, such as gender, job, position, which are translated into several high-dimensional sparse features via one-hot encoding. For example, [*gender=female, job=teacher*] can be represented as:

$$\underbrace{[1, 0, 0]}_{gender}\underbrace{[0, 1, ..., 0]}_{job}. \tag{8}$$

Then the raw sparse feature $\mathbf{u}_{u_{ms}}$ is fed into the MLPs to generate low-dimensional real-valued dense vector $\mathbf{u}_{um}$:

$$\mathbf{u}_{um} = MLP(\mathbf{u}_{u_{ms}}). \tag{9}$$

User clicked videos can directly reflect users' preference, such as preferred video type, watching habits (prefer long or short video) and so on. Applying the similar way as learning $\mathbf{u}_{uq}$ and $\mathbf{u}_{uu}$, we get $\mathbf{u}_{ud}$ as following:

$$\mathbf{u}_{ud} = f_{ud}(\mathbf{d}_{ui}, i = 1, 2, ..., K), \tag{10}$$

[3] There is no order between $\mathbf{u}_{u1}$, $\mathbf{u}_{u2}$... $\mathbf{u}_{uK}$. A user issued queries, $\mathbf{q}_{u1}, \mathbf{q}_{u2}, ..., \mathbf{q}_{uK}$ can be sequential datasets if we have their timestamp information.

[4] f_{uq_s} means summation of the embeddings of neighbors, f_{uq_m} means average of these embeddings. $f_{uq_{mp}}$ applies max-pooling operator to each of the computed feature.

where $\mathbf{d}_{ui}$ is the ID embedding from video ID embedding matrix $\mathbf{D} \in \mathbb{R}^{N_q \times D}$.

2.5 Ranking Score Generation

Finally, we concatenate u_u's ID embedding $\mathbf{u}_u$, embeddings learned from meta-information $\mathbf{u}_{um}$, u-q graph $\mathbf{u}_{uq}$, $\mathbf{u}_{uu}$ and clicked videos $\mathbf{u}_{ud}$ to get its new embedding $\mathbf{u}_u^{'}$

$$\mathbf{u}_u^{'} = \mathbf{u}_u \oplus \mathbf{u}_{um} \oplus \mathbf{u}_{uq} \oplus \mathbf{u}_{uu} \oplus \mathbf{u}_{ud}, \tag{11}$$

where $\oplus$ is the operation of vector concatenation.

For query q_q, we concatenate its ID embedding $\mathbf{q}_q$, text embedding $\mathbf{q}_{q\_t}$ and embeddings $\mathbf{q}_{qd}$ and $\mathbf{q}_{qq}$ learned from q-d graph to get its new embedding $\mathbf{q}_q^{'}$

$$\mathbf{q}_q^{'} = \mathbf{q}_q \oplus \mathbf{q}_{q\_t} \oplus \mathbf{q}_{qd} \oplus \mathbf{q}_{qq}. \tag{12}$$

For video d_d, we concatenate its ID embedding $\mathbf{d}_d$ and, semantic embeddings learned from text and video $\mathbf{d}_{d\_t}$, $\mathbf{d}_{d\_v}$ and $\mathbf{d}_{dd}$ learned from neighboring videos in q-d graph to get it new representation $\mathbf{d}_d^{'}$

$$\mathbf{d}_d^{'} = \mathbf{d}_d \oplus \mathbf{d}_{d\_t} \oplus \mathbf{d}_{d\_v} \oplus \mathbf{d}_{dd}. \tag{13}$$

After we got the new embeddings for user, query and videos, we need to further refine the query representation by injecting more personalized information. Instead of directly matching $\mathbf{q}_q^{'}$ and $\mathbf{d}_d^{'}$, we first combine $\mathbf{u}_u^{'}$ with $\mathbf{q}_q^{'}$ to get the personalized query embedding as following:

$$\mathbf{q}_q^p = f_u(\mathbf{u}_u^{'}) \oplus f_q(\mathbf{q}_q^{'}), \tag{14}$$

where f_u and f_q are MLPs and map $\mathbf{u}_u^{'}$ and $\mathbf{q}_q^{'}$ to the same vector space. Meanwhile, $\mathbf{d}_d^{'}$ will be also projected to the same space with $\mathbf{q}_q^p$ by an MLPs f_d

$$\mathbf{d}_d^p = f_d(\mathbf{d}_d^{'}). \tag{15}$$

The click probability of $<u_u, q_q, d_d>$ is calculated as the inner product of $\mathbf{q}_q^p$ and $\mathbf{d}_d^p$,

$$\hat{y}_c = (\mathbf{q}_q^p)^T \mathbf{d}_d^p. \tag{16}$$

Besides, our model also considers the relevance between queries and videos and we calculate their correlation $\hat{y}_r$ by the inner product of $\mathbf{q}_q^{'}$ and $\mathbf{d}_d^{'}$

$$\hat{y}_r = (f_0(\mathbf{q}_q^{'}))^T f_1(\mathbf{d}_d^{'}), \tag{17}$$

where f_0, f_1 are MLPs to map $\mathbf{q}_q^{'}$ and $\mathbf{d}_d^{'}$ to the same embedding space. The final ranking score can be obtained as following

$$y = \hat{y_c}^{\beta} \hat{y}_r^{(1-\beta)}, \tag{18}$$

where $\beta \in (0,1)$ is a hyper-parameter. We use the final ranking score y to rank the candidate videos.

2.6 Model Training and Optimization

The training objective of our model consists of two terms. The first one is the binary cross-entropy loss

$$\mathcal{L}_1 = -\frac{1}{M}\sum_{j=1}^{M} o_j \times logp(\mathbf{d}_j|\mathbf{q}_q) + (1-o_j) \times log(1-p(\mathbf{d}_j|\mathbf{q}_q)), \tag{19}$$

where M denotes the number of training pairs and o_j represents binary click label for $\mathbf{d}_j$. The second term is the mean squared error

$$\mathcal{L}_2 = \frac{1}{M}\sum_{j=1}^{M}(y_j - \hat{y}_j), \tag{20}$$

where y_j and $\hat{y}_j$ are respectively the real and predicted relevance score. The final objective is

$$\mathcal{L} = \alpha\mathcal{L}_1 + (1-\alpha)\mathcal{L}_2, \tag{21}$$

where α is the hyper-parameters to control the balance.

3 EXPERIMENT

3.1 Datasets

To evaluate the effectiveness of our proposed framework, we collect the online search logs of a major commercial video search engine to construct a personalized search dataset, which could be a proper test bed to evaluate personalized search models. We collect a total of 21 days' search logs, which are split into 3 parts. Graphs are constructed based on the first part, i.e., search logs from the first 15 days. The constructed graphs have millions of nodes and tens of millions of edges. When constructing the two bipartite graphs, we only preserve five first-order and five second-order neighbors for each node. The next five days' logs are used to construct the training set and the logs from the last day are used as testing data. There are totally 1,289,314 unique users, 1,315,851 unique queries and 5,368,904 unique videos, where the training set consists of 10,802,573 samples, and testing set consists of 3,335,752 samples. For users, besides the user ID features, we also include users' meta information, such as age, education level, gender, profession and city, for better user representation.

For the click task, we formulate it as a binary classification task and collect the online user click feedback as the click label. Click means one and not click means zero. For the query-video relevance task, we formulate it as a regression task. The relevance labels are collected using an internal tool which could automatically tag the relevance label of a query-document pair. [5] The relevance labels range from zero to four, where four indicates the highest relevance level and zero means irrelevant.

3.2 Baselines and Experimental Setup

We compare our model with two categories of baseline methods: classical information retrieval and semantic matching methods (NCF [12], DSSM [16], DNN) and three representative personalized search models: P-Click model (traditional statistic model) [7], two types of DL-based models: NN-PVS (neural ranking model with personalized information [15]) and RNN-PVS (RNN based sequential models to mining sequential information to learn users' search intents [1]). We evaluate above methods on the click task and the relevance task. In both tasks, we use AUC and a commonly used evaluation metric, i.e., nDCG@k (k=1, 3, 5), to evaluate these models and report results over 5 runs with random parameter initialization.

- **NCF** [12]. Neural Collaborative Filtering (NCF) is a neural network architecture to model latent features of users and items for top-K recommendation. In our scenarios, we feed it with the query IDs and video IDs instead of users and items.
- **DSSM** [16]. Deep Structured Semantic Model (DSSM) is proposed for web search semantic matching. We measure the similarities of query and videos' representations learned from their text information with BERT.
- **CDL**. Content-enhanced deep learning model (CDL) feed query ID embeddings concatenated with text embedding and video ID embeddings concatenated with text and video embeddings into the MLP layers.
- **P-Click model** [7]. P-Click model predicts the probability of clicking by counting the number of historical clicks. It focuses on the user's re-finding behavior.
- **NN-PVS** [15]. It is a neural ranking model with personalized information introduced in [15]. In our paper, NN-PVS utilizes users' meta information to refine the query representation. We concatenate the user embedding learned from users' meta information with the query embedding and use the MLPs to map the concatenated vector to the same embedding space with the video for the final matching.
- **RNN-PVS**. This is an RNN based sequential model that learns users' profile with their history information [1]. We concatenate the user profile embedding with query embedding for the final matching.
- **GNN-PVS**. Our proposed model, where only click labels are used for training. We utilize four different aggregation functions: mean (GNN-PVS_m), sum (GNN-PVS_s), maxpooling (GNN-PVS_mp) and attention (GNN-PVS_att). [6]
- **MGNN-PVS**: It is a variant of GNN-PVS, where a multi-task learning framework is incorporated to simultaneously learn the click and relevance task. Likewise, we also utilize mean, sum, maxpooling and attention four different aggregation functions in our model.

The the dimension of text embedding is 128 and 256 for the video embedding. The other main hyper-parameters are set as: , the dimension of ID embedding $\in$ {16, 32, 64}, the dimension of learned user profile $\in$ {16, 32, 64}, the batch-size: 512, the dropout rate $\in$ {0.2,0.4,0.6}, learning rate $\in$ {0.01, 0.001, 0.0001}. These hyper-parameters are finally selected according to the performance on the validation dataset. We use the Adam optimizer [17] and apply the early stopping strategy based on the validation accuracy to avoid overfitting. Considering both accuracy and efficiency, we choose the mean aggregation function in our framework. α is set as 1 for GNN-PVS which only learns the click label and set as 0.5 for MGNN-PVS to jointly model click and relevance.

3.3 Experimental Results.

3.3.1 ***Overall Performance Comparison.*** The experimental results are summarized in Table 2. Our proposed GNN-based models (GNN-PVS and MGNN-PVS) significantly and consistently outperform all other baselines on both click and relevance tasks. Compared with the best baselines, our model outperforms the best baselines by 7.94%, 9.74%, 15.38%, 13.04% for the click task and 5.12%, 5.43%, 4.42% for the relevance task, which illustrates that utilizing graph

[5] We could also get manual labels based on crowdsourcing platforms, such as Amazon Mechanical Turk.

[6] Considering accuracy, efficiency and stability, we use mean as our aggregation function if there is no specification.

Table 2: Overall performance of all models (%).
The best results are in bold and the best ones of other baselines are underlined.

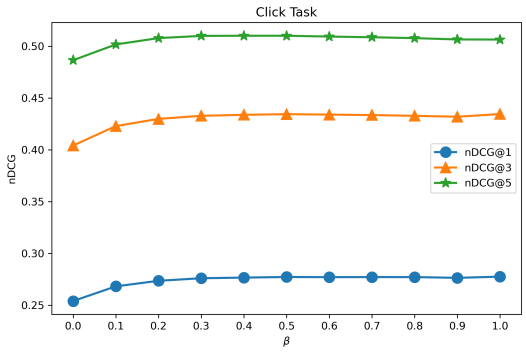
		Click Task				Relevance Task		
	Methods	AUC	nDCG@1	nDCG@3	nDCG@5	nDCG@1	nDCG@3	nDCG@5
	DSSM	67.82 ± 0.43	20.8 ± 0.55	35.0 ± 0.37	40.6 ± 0.30	67.0 ± 0.42	69.5 ± 0.40	72.4 ± 0.35
	NCF	72.48 ± 0.61	21.8 ± 0.62	37.4 ± 0.49	45.1 ± 0.44	66.8 ± 0.40	68.9 ± 0.33	71.7 ± 0.40
	CDL	73.02 ± 0.20	22.7 ± 0.39	37.6 ± 0.31	44.5 ± 0.42	68.6 ± 0.29	70.0 ± 0.22	72.3 ± 0.19
	Pclick	60.8	15.9	26.9	35.0	64.4	66.4	69.4
	NN-PVS	70.63 ± 0.27	23.3 ± 0.82	38.9 ± 0.62	46.2 ± 0.34	66.8 ± 0.34	67.6 ± 0.19	71.6 ± 0.23
	RNN-PVS	68.33 ± 0.51	21.6 ± 0.45	37.0 ± 0.31	39.3 ± 0.40	66.6 ± 0.42	67.3 ± 0.23	70.8 ± 0.16
GNN	GNN-PVS_m	76.93 ± 0.46	27.6 ± 0.62	44.0±0.34	51.6 ± 0.22	70.5 ± 0.20	71.8 ± 0.17	74.0 ± 0.21
GNN	GNN-PVS_s	76.00 ± 0.19	25.3 ± 0.31	42.9±0.19	49.8 ± 0.12	70.2 ± 0.34	72.4 ± 0.22	73.8 ± 0.21
GNN	GNN-PVS_mp	**78.87 ± 0.82**	27.8 ± 0.73	44.4±0. 59	51.7 ± 0.62	70.3 ± 0.74	71.8 ± 0.48	74.2 ± 0.46
GNN	GNN-PVS_att	78.18 ± 0.27	**27.9 ± 0.50**	**45.0 ±0.47**	**51.9 ± 0.19**	70.6 ± 0.35	72.2 ± 0.25	74.9 ± 0.22
MGNN	MGNN-PVS_m	75.52 ± 0.46	27.2 ± 0.72	42.9 ±0.33	50.6 ± 0.16	72.2 ± 0.41	73.3 ± 0.22	**75.6 ± 0.17**
MGNN	MGNN-PVS_s	74.67 ± 0.50	24.8 ± 0.39	40.5±0.27	48.6 ± 0.21	71.7 ± 0.33	73.5 ± 0.27	74.1 ± 0.16
MGNN	MGNN-PVS_mp	77.03 ± 0.69	27.5 ± 0.82	43.4±0.65	50.6 ± 0.58	**72.3 ± 0.66**	72.9 ± 0.49	75.0 ± 0.41
MGNN	MGNN-PVS_att	76.00 ± 0.26	26.8 ± 0.52	3.2±0.31	0.7 ± 0.24	71.9 ± 0.40	**73.8 ± 0.23**	75.3 ± 0.21
	Improvement %	+7.94%	+19.74%	+15.38%	+13.04%	+5.12%	+5.43%	+4.42%

information greatly benefits the representation learning for users, queries and videos. Compared with GNN-based models, MGNN-based models get better performance on the relevance task, but not the click task. This is because MGNN-based models are trained with both click and relevance labels, while GNN-based models are trained only with the click labels.

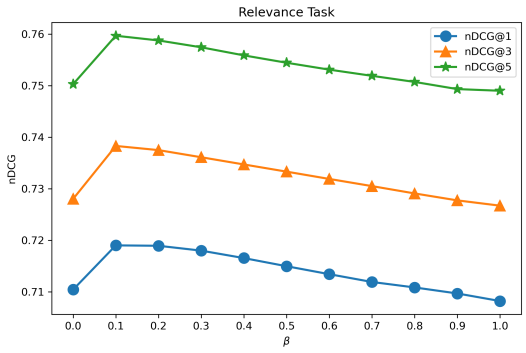
Besides, non-personalized baselines (DSSM, DCF, CDL) usually perform better than personalized search models (Pclick, NN-PVS, RNN-PVS) on the relevance task, but not the click task. The possible reason is that incorporating user's information benefits the click task, but may introduce some noisy information, such as users clicked the video only because of attractiveness, but it may be not relevant to their issued queries. This also indicates the necessity of training the model with both click and relevance labels.

The P-Click model, the traditional statistical method, gets poor results, while other deep learning based models have shown strong generalization ability by capturing the implicit similarities among users, queries and documents. NN-PVS gets better results than RNN-PVS. One possible reason is that RNN is more suitable for sequential data, but there is no timestamp information for users' history information in our scenario. Compared with RNN-PVS, NN-PVS is a more general and easily applied PSM.

3.3.2 ***Trade-off between Two Tasks.*** For MGNN-PVS, a key hyper-parameter is the trade-off parameter β. We would like to explore the influence of β to the performance on the click task and relevance task by varying the value of β from zero to one. Results are reported in Fig. 5. We found that MGNN-PVS achieves better performance on the relevance task when β is small. When the value of β increases, the performance of the click task increases while the performance of the relevance task decreases. This observation matches our intuition since β controls the relative impacts of click prediction score and relevance prediction on the final ranking score.

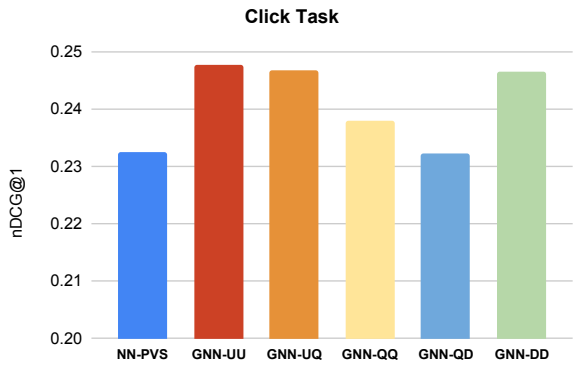


(a) nDCG of Click Task

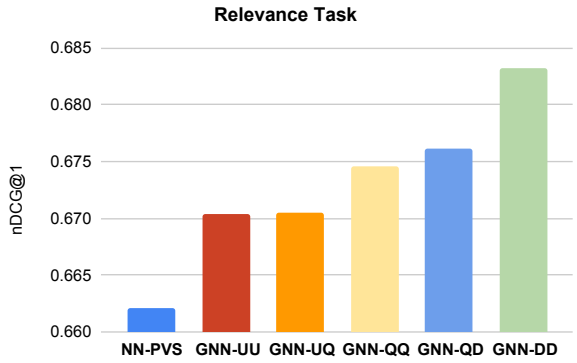


(b) nDCG of Relevance Task

Figure 5: β's influence to the prediction results.



(a) nDCG@1 of click task



(b) nDCG@1 of relevance task

Figure 6: Effectiveness of each graph module: (a) and (b) show the nDCG@1 of click task and relevance task. NN-PVS is the baseline, only considering the user's meta information. GNN-AB means only A-B graph information is used in the training process, such as GNN-UU means only the user's neighboring users (user-user graph information) are used in the learning process.

3.3.3 ***Effectiveness of Graph Information.*** To validate the effectiveness of graph information, we add each component of the graph to examine its relative importance as shown in Fig. 6. For example, GNN-UU means our model only leverages neighboring users' information (user-user) from the user-query graph, without considering other components of the two graphs. Several observations can be made from Fig. 6. First, adding user-user and user-query information has more improvements than adding query-query, query-documents information for the click task. When uses' history information is sparse, they may not be well represented. Hence, adding their neighboring users and queries may assist our model to learn better representation. Second, the improvement brought by second-order neighboring videos in the query-document graph outweighs that brought by other graph information for the relevance task. One possible reason is that videos can learn more accurate embeddings with considering their neighboring videos. Finally, compared with other graph information, the improvement brought by query-query and query-document graph information is not that obvious. We checked our dataset and found that the connection is very sparse for query in the query-document graph, and this may result that adding query-query and query-document graph information does not perform as good as adding user-user or video-video information.

3.3.4 ***Case Study.*** To get deep insights on how the graph information assists the personalized search task. We choose two triples <*user, query, video*> with the same query *Shepherd of the Cocoa Sea* (*SCS*) and video *Beautiful street performance of the Shepherd of the Cocoa Sea*, but different users with the ground truth labels zero for *user47* (no-click) and one for *user75* (click). We compare the click results of NN-PVS and our model. NN-PVS predicts both *user47* and *user75* click the video when issuing this query *SCS*, because the query and video title are very relevant, containing both *Shepherd of the Cocoa Sea*. But our model predicts zero for *user47* and one for *user75*.

From the user-query graph, we find three related first-hop neighbors of *user47*: *the original singer of SCS*, *the complete and original SCS*, *Wang CSC* [7], which illustrates that *user47* prefers the original song *CSC* much. Beyond the text information of the query and video, our model can utilizes these information and discover people's real interests, but NN-PVS can not. Besides we also observe that the query *Cocoa Shepherd*, a wrong expression of *SCS*, has three second-hop neighbors (similar queries): *SCS*, *SCS orignal*, *Song of SCS* in the query-document graph, which contains more related and accurate expressions for the query *SCS*. Aggregating these information can help the model learn a more accurate and robust representation of queries, even with wrong expressions.

4 RELATED WORK

The main idea of traditional static personalized search algorithms in [7, 24, 25] is that they evaluate the click probability by counting the number of documents clicked by the same user under the same query. Some studies attempt to extract the topics features by utilizing Open Directory Project (ODP) [23, 31] or latent semantic analysis(LSA) or Latent Dirichlet Allocation (LDA) techniques [3, 11, 28, 29]. But these topic-based models are often trained in an unsupervised manner, thus the performance of these models on personalized search is not as good as expected. Recently, deep learning methods have been successfully applied to a variety of language and information retrieval applications. One category of DL-based PSMs mainly follow the general framework of traditional DL search methods [16], meanwhile incorporate more personal information. [15] encodes both textual information of the query, and users' social connections [34] or location [14] to represent the query. Another category methods aim to learn the user profile from users' long-term or short-term search history by bi-LSTM, RNN, GRU or transformer to mining sequential information to learn users' search intents [1, 8, 20, 40].

One of the biggest concerns in personalized video search is the data sparsity issue, where each user only has a handful of queries in their search histories and therefore limits the learning capability of the personalization algorithms. Besides, current PSMs learn the query and document representation only from their textual content.

[7] Wang is the original singer of CSC

When queries are short and vague, it is difficult for current PSMs to learn accurate representations. What's more, current PSMs only use the click signal to train their models and this manner is not suitable in our scenario, because the real industry video platform exists much noisy click information.

5 CONCLUSION

In this paper, we proposed an efficient multi-task graph convolutional network architecture for personalized video search (MGNN-PVS) and optimized both the click task and relevance between videos and queries. In addition to learning the representations of users, queries and videos from their own ID, textual and video content, our approach can leverage the user-query graph and query-document click graph information to relieve the sparsity problem and help disambiguate short and vague queries. Considering the real data contains much noisy click information that users' click signals may indicate attractiveness but not necessarily indicate relevance. Thus, we jointly model the user's click behaviour and the relevance between queries and videos in our algorithm. Experimental results on the real-world dataset showed that our proposed model significantly outperforms state-of-the-art PSMs on both click task and relevance task, which illustrates the effectiveness of our proposed framework.

6 ACKNOWLEDGMENTS

This work was supported in part by the China Scholarship Council (CSC) under Grant 201706080010.

REFERENCES

[1] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. 2018. Multi-task learning for document ranking and query suggestion. In *ICLR*.
[2] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*.
[3] Mark J. Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. 2010. Towards Query Log Based Personalization Using Topic Models. In *CIKM*.
[4] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*.
[5] Nick Craswell and Martin Szummer. 2007. Random walks on the click graph. In *SIGIR*.
[6] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
[7] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A Large-Scale Evaluation and Analysis of Personalized Search Strategies. In *WWW*.
[8] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In *CIKM*.
[9] Yan Ge, Pan Peng, and Haiping Lu. 2021. Mixed-order spectral clustering for complex networks. *Pattern Recognition* 117 (2021), 107964.
[10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40 (2017), 52–74.
[11] Morgan Harvey, Fabio Crestani, and Mark J Carman. 2013. Building user profiles from topic models for personalised search. In *CIKM*.
[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.
[13] Winston H Hsu, Lyndon S Kennedy, and Shih-Fu Chang. 2007. Video search reranking through random walk over document-level context graph. In *MM*.
[14] Jizhou Huang, Haifeng Wang, Miao Fan, An Zhuo, and Ying Li. 2020. Personalized prefix embedding for POI auto-completion in the search engine of Baidu Maps. In *SIGKDD*.
[15] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *SIGKDD*.
[16] 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 *CIKM*.
[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[19] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *NeurIPS*.
[20] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. Psgan: A minimax game for personalized search with limited and noisy click data. In *SIGIR*.
[21] Hao Ma, Haixuan Yang, Irwin King, and Michael R Lyu. 2008. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM*.
[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.
[23] Ahu Sieg, Bamshad Mobasher, and Robin Burke. 2007. Web search personalization with ontological user profiles. In *CIKM*.
[24] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. 2007. Information re-retrieval: Repeat queries in Yahoo's logs. In *SIGIR*.
[25] Jaime Teevan, Daniel J Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *ICDM*.
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
[27] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[28] Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *ECIR*.
[29] Thanh Vu, Alistair Willis, Son N Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *ECIR*.
[30] Jörg Waitelonis and Harald Sack. 2012. Towards exploratory video search using linked data. *Multimedia Tools and Applications* 59, 2 (2012), 645–672.
[31] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *WWW*.
[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
[33] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*.
[34] Jing Yao, Zhicheng Dou, and Ji-Rong Wen. 2020. Employing Personal Word Embeddings for Personalized Search. In *SIGIR*.
[35] Tan Yu, Yi Yang, Yi Li, Xiaodong Chen, Mingming Sun, and Ping Li. 2020. Combo-Attention Network for Baidu Video Advertising. In *SIGKDD*.
[36] Li Zhang, Yan Ge, and Haiping Lu. 2020. Hop-Hop Relation-aware Graph Neural Networks. In *ECML(GEM)*.
[37] Li Zhang and Haiping Lu. 2020. A Feature-Importance-Aware and Robust Aggregator for GCN. In *CIKM*.
[38] Li Zhang, Heda Song, and Haiping Lu. 2018. Graph node-feature convolution for representation learning. In *CIKM (GRL)*.
[39] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *RecSys*.
[40] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding History with Context-aware Representation Learning for Personalized Search. In *SIGIR*.