TGCN: Tag Graph Convolutional Network for Tag-Aware Recommendation

Bo Chen[1], Wei Guo[1], Ruiming Tang[1†], Xin Xin[2], Yue Ding[3], Xiuqiang He[1], Dong Wang[3†]
[1]Noah's Ark Lab, Huawei, [2]University of Glasgow, [3]Shanghai Jiao Tong University, † Corresponding authors
{chenbo116,guowei67,tangruiming,hexiuqiang1}@huawei.com,x.xin.1@research.gla.ac.uk,{wangdong,dingyue}@sjtu.edu.cn

ABSTRACT

Tag-aware recommender systems (TRS) utilize rich tagging records to better depict user portraits and item features. Recently, many efforts have been done to improve TRS with neural networks. However, these solutions rustically rely on the tag-based features for recommendation, which is insufficient to ease the sparsity, ambiguity and redundancy issues introduced by tags, thus hindering the recommendation performance.

In this paper, we propose a novel tag-aware recommendation model named *Tag Graph Convolutional Network* (TGCN), which leverages the contextual semantics of multi-hop neighbors in the user-tag-item graph to alleviate the above issues. Specifically, TGCN first employs type-aware neighbor sampling and aggregation operation to learn the type-specific neighborhood representations. Then we leverage attention mechanism to discriminate the importance of different node types and creatively employ Convolutional Neural Network (CNN) as type-level aggregator to perform vertical and horizontal convolutions for modeling multi-granular feature interactions. Besides, a TransTag regularization function is proposed to accurately identify user's substantive preference. Extensive experiments on three public datasets and a real industrial dataset show that TGCN significantly outperforms state-of-the-art baselines for tag-aware top-N recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommendation, Collaborative Tagging, Graph Neural Network, Representation Learning

ACM Reference Format:
Bo Chen, Wei Guo, Ruiming Tang, Xin Xin, Yue Ding, Xiuqiang He and Dong Wang. 2020. TGCN: Tag Graph Convolutional Network for Tag-Aware Recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411927

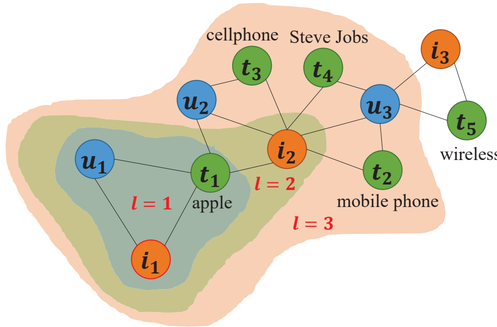


Figure 1: An example of graph for easing sparsity, ambiguity and redundancy. Tag "*apple*" is polysemy (ambiguity); "*cellphone*" and "*mobile phone*" are synonymous (redundancy).

1 INTRODUCTION

Social tagging systems, also known as folksonomies, are widely used in various websites and applications, where users can freely annotate online items (*e.g.*, movies, artists) with arbitrary tags [9]. Generally, these tags are composed by laconic words or phrases that express users' interests in the certain item. Benefit from the pithiness, tags are more direct and objective than reviews, which can not only indicate user preferences, but also summarize characteristics of items [5, 19]. Therefore, user-defined tags can be introduced into recommender systems for alleviating the cold-start problem [35] and improving recommendation quality.

To integrate social tagging information, a common paradigm is to transform the tags into a multi-hot feature vector and further feed into *feature-based* models for recommendation. For example, CFA [36] uses the sparse autoencoder (SAE) to obtain tag-based user latent representations and combines it with user-based collaborative filtering (CF). Besides, DSPR [31] and HDLPR [32] leverage the multi-layer perceptron (MLP) to process such sparse multi-hot feature vector and extract abstract user and item representations. However, the ***sparsity*** issue arises as some users annotate a small amount of tags to a few items, making it difficult to recognize users' preferences. Besides, tags may also suffer from ***ambiguity*** and ***redundancy*** issues [23] due to the lack of contextual semantics. For example, polysemous tag "*apple*" could be confused as a kind of fruit or a technology company if no other contextual information is given, which introduces ambiguity to the feature-based models. Moreover, limited by the diversity in user's writing or expression styles, some tags with different spellings have the same meanings and indicate similar preferences, such as "*mobile phone*" and "*cellphone*". However, due to the missing of relationship modeling in feature-based models, these synonymous tags may be interpreted as completely different semantics, resulting in redundancy.

To solve these issues, we argue that, by organizing the user tagging triple records as a heterogeneous user-tag-item graph, the multi-hop neighbors contribute to provide more contextual semantic information. Figure 1 illustrates an example. User u_1 only has tag t_1 ("*apple*"), making the feature vector highly sparse. By aggregating information of multi-hop neighbors, the sparsity can be well alleviated. Besides, not only polysemous tag t_1 ("*apple*") can be clearly interpreted as a technology company rather than a kind of fruit, but also synonymous tags t_2 ("*mobile phone*") and t_3 ("*cellphone*") can be semantically close due to similar neighborhood subgraphs. As such, ambiguity and redundancy can be eased commendably with the help of multi-hop neighbors in the graph.

Several recent works which utilize Graph Neural Network (GNN) for recommendation have demonstrated its effectiveness to integrate both node features and topological structure for representation learning, such as PinSage [33], NGCF [29], KGAT [28] and HGAT [18]. However, combining user tagging records with GNN-based models for recommendation poses three unique challenges:

C1: How to aggregate heterogeneous neighbors with all types of neighborhood information preserved and the different impact of neighbors considered is important in heterogeneous graph modeling. User-tag-item graph is sophisticated as each central node may have different numbers and types of neighbors. Therefore, it is essential to consider the diverse contributions of individual heterogeneous neighbors.

C2: Once different types of neighborhood are represented, how to model feature interactions among them is vital. Existing heterogeneous graph convolutional models simply perform pooling-based operations during the information updating stage, neglecting the informative feature interactive signals. However, effectively modeling feature interactions is critical in recommender systems [6, 15].

C3: As user annotated tags are the key factor to represent user's preference over items, how to capture these semantics in the user-tag-item triplets is essential. Such strong semantics among user, tag and item may be important to make the recommendation satisfactory. However, the aggregation strategy of GNN-based models overlooks such semantics, leading the learned representations sub-optimal.

To solve the issues faced by tag-aware recommendation and overcome the above three challenges, we propose a novel end-to-end neural recommendation framework named ***Tag Graph Convolutional Network*** (TGCN). Specifically, to alleviate sparsity, ambiguity and redundancy issues, we utilize the user tagging records to construct an undirected weighted ***Collaborative Tag Graph*** (CTG). Based on the CTG, our proposed TGCN meticulously designs some modules to tackle the above three challenges. **First**, to cope with the heterogeneity of graph nodes, we propose a type-aware neighbor sampling and aggregation operation. The heterogeneous neighbor sampling strategy is designed to sample a fixed size of neighbors for each type and the type-aware neighbor aggregator is proposed to learn the type-specific neighborhood representation attentively. **Second**, to effectively capture feature interactions among different types of neighborhood representations, we first leverage type-level attention to discriminate the importance of different node types and then creatively replace the general pooling-based method with CNN which performs vertical and horizontal convolutions to capture multi-granular feature interactions. **Third**, to capture the semantics in user-tag-item triplets and depict user's preference accurately, we design a TransTag regularization function to model the node representations on the granularity of annotation triples and perform jointly learning with the recommendation task.

To sum up, our contributions in this paper can be summarized as follows:

- We construct CTG based on the user tagging records and leverage the contextual semantics of multi-hop neighbors to ease sparsity, ambiguity and redundancy issues existing chronically in the tag-aware recommendation.
- We develop a novel tag-aware recommendation model TGCN to learn node representations. Through employing type-aware neighbor sampling and aggregation strategy, as well as CNN-based information updating operation, TGCN can sufficiently extract informative features from heterogeneous neighbors. Besides, we introduce TransTag function and perform jointly learning to further regularize node representations with the semantics of annotation triples preserved.
- We perform extensive experiments on three public datasets and a real industrial dataset, demonstrating significant improvements of TGCN over state-of-the-art methods for tag-aware top-N recommendation.

2 RELATED WORK

2.1 Tag-aware Recommendation

Recent research shows impressive performance by incorporating social tagging information into neural network-based methods for personalized recommendation. These methods transform the tags into a sparse feature vector and leverage neural network to extract latent representations. CFA [36] uses the stacked SAE to obtain tag-based user abstract representations and combines with user-based CF for recommendation. DSPR [31] leverages the MLP to map the tag-based user and item profiles into an abstract feature space and maximizes deep-semantic similarities between user and relevant items. Based on this model, HDLPR [32] uses the autoencoder (AE) with reconstruction errors for further accelerating the learning progress. As for rating prediction, TRSDL [16] utilizes MLP to extract item latent representations and then uses recurrent neural network (RNN) to process sequential historical items for building user portraits. Besides, social tags can be introduced into recommender systems as a kind of superior side information and combine with neural networks to make recommendation [25].

Despite a substantial amount of effort has been made to tag-aware recommendation, these feature-based solutions cannot effectively alleviate the sparsity, ambiguity and redundancy problems. Our proposed method organizes the user tagging records into a CTG where the multi-hop neighbors can provide more contextual semantics for enriching features and alleviating problems.

2.2 GNN-based Representation Learning

Graph neural networks which aim to extend the deep neural networks to deal with arbitrary graph-structured data, have received widespread attention in recommender systems recently. GCN [12] leverages a spectral graph convolutional operation to encode both graph structure and nodes features. Then GC-MC [1] first applies GCN on the user-item bipartite graph to achieve link prediction. Besides, NGCF [29] improves the recommendation effect by explicitly modeling the high-order collaborative signals. In order to identify the importance of neighbors, GAT [27] introduces attention mechanism to measure impacts of different neighbors. On the other hand, GraphSage [7] performs a non-spectral graph convolution over a fixed size of sampled neighbors to integrate neighbor features for learning accurate node representations. Based on this, PinSage [33] deploys it at Pinterest for web-scale recommendation.

To make full use of side information beyond modeling user-item interactions, KGAT [28] incorporates user-item graph with knowledge graph and combines graph convolution with attention mechanism to get finer node representations. For heterogeneous graph modeling, HGAT [18] utilizes a dual-attention network to discriminate the importance of neighbor nodes and node types. Moreover, HetGNN [34] jointly considers heterogeneous contents encoding, type-based neighbors aggregation and types combination to perform heterogeneous graph representation learning.

Although these GNN-based models have made progress in homogeneous or heterogeneous graph representation learning, applying them directly on CTG is difficult to overcome the challenges mentioned in Section 1, which prompts us to propose TGCN model. A more comprehensive comparison will be elaborated in Section 4.5.

3 PRELIMINARIES

3.1 Problem Formulation

User tagging behavior is defined as a set of tagging assignments generated by users after interacting with items. To be specific, a tagging assignment is aggregated into a triple, *i.e.*, $a = \langle u, t, i \rangle$, meaning that user u annotates tag t to item i. These personalized tags reflect users' subjective cognition and preferences of the items. Besides, the collaborative tagging procedure is also conducive to depict the characteristics of items comprehensively. By exploring these rich user tagging records, we can better infer user preferences, summarize item features and improve recommendation performance.

Suppose the sizes of user set $\mathcal{U}$, item set $\mathcal{I}$ and tag set $\mathcal{T}$ are N_u, N_i and N_t respectively, a folksonomy is a tuple $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{I}, \mathcal{A})$ [9], where $\mathcal{A} \subseteq \mathcal{U} \times \mathcal{T} \times \mathcal{I}$ is a set of assignments $a = \langle u, t, i \rangle$. The user-item interaction matrix is represented by $R \in \mathbb{N}^{N_u \times N_i}$, whose entries can be explicit ratings with range [1-5] or implicit feedback such as click or annotation. In TRS scenario, each entry R_{ui} is defined as: $R_{ui} = 1$ if the interaction (annotation) between user u and item i is observed, and 0 otherwise. Our research task is to mine user tagging records and learn a recommendation model for generating a ranked list of items that will be of interest for each user $u \in \mathcal{U}$, *i.e.*, top-N recommendation.

3.2 Collaborative Tag Graph

Definition 3.1 (***Collaborative Tag Graph***). Let $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{W}$ denote the sets of nodes, edges and edge weights respectively, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{T}$. A collaborative tag graph is defined as an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ with a node type mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{H}, \forall v \in \mathcal{V}$, where $\mathcal{H} = \{1, 2, 3\}$ whose elements represent *user*, *item* and *tag* node type, respectively. For each edge $e = (v, v', w) \in \mathcal{E}$, it represents that the edge weight between node v and v' is $w \in \mathcal{W}$.

For assignment $a = \langle u, t, i \rangle$ in $\mathcal{A}$, we construct three edges among them. Edge $e = (u, i, 1)$ reflects the interaction relationship between user u and item i. Edge $e = (u, t, w)$ describes the tagging relationship between user u and tag t. And edge $e = (i, t, w)$ indicates the passive annotated relationship between item i and tag t. We assign the weights between user and item as 1, and the weights between tag and user/item as the frequencies that tags have been annotated since it can reflect the degree of user interest and item attribute [3]. To be specific, $e = (u, t, w) \in \mathcal{E}$ means user u annotates some items with tag t for w times. Analogously, $e = (i, t, w) \in \mathcal{E}$ indicates item i is annotated with tag t by some users for w times. Take a simple toy example. Suppose the assignment set is represented as $\mathcal{A} = \{\langle u_1, t_1, i_1 \rangle, \langle u_1, t_1, i_2 \rangle, \langle u_2, t_1, i_1 \rangle, \langle u_2, t_2, i_1 \rangle, \langle u_2, t_1, i_3 \rangle\}$, after adding a self-loop with weight 1 to each node, the corresponding CTG $\mathcal{G}$ can be built and illustrated in the left of Figure 2.

4 TAG GRAPH CONVOLUTIONAL NETWORK

Tag Graph Convolutional Network (TGCN) is illustrated in Figure 2, which is composed of three core modules: 1) ***Type-aware Neighbor Aggregation***, which samples a fixed size of neighbors and aggregates them with type-specific node-level attention for each type; 2) ***Information Updating***, which leverages a type-level attentive mechanism to re-scale weights of different types of neighborhood representations and then performs a CNN-based information updating operation to capture multi-granular feature interactions and update node representations; 3) ***TransTag Regularization***, which provides a TransTag function to regularize node representations with the semantics of annotation triples preserved.

4.1 Type-aware Neighbor Aggregation (C1)

In traditional GNN models, such as NGCF [29] and KGAT [18], nodes propagate and aggregate integral or partial information from heterogeneous neighbors indiscriminately, shown in Figure 3(a). We argue it may loss information due to the negligence of intrinsic differences among heterogeneous neighbors. To solve the challenge **C1**, we propose type-aware neighbor sampling and aggregation strategy, depicted in Figure 3(b). For each node type, a fixed size of neighbors are sampled and a node-level attention are utilized to aggregate and obtain type-specific neighborhood representations.

4.1.1 Heterogeneous Neighbor Sampling. Generally speaking, aggregating all neighbors directly may raise several problems. 1) Exponential growth of the neighborhood size with the number of hops makes it impractical to store and calculate in large-scale graphs. 2) Due to the various neighbor sizes, some "hot" nodes (popular items or frequently-used tags) may have plentiful of neighbors while some "cold" nodes (long-tailed items or esoteric tags) have only a few neighbors. Therefore, the "hot" nodes embeddings may be impaired by weakly correlated neighbors while "cold" nodes embeddings may not be sufficiently represented. 3) Aggregating all neighbors may lead to over-smoothing issue [14]. The dense connections between nodes make the neighborhood unduly similar, resulting in indistinguishable representations. Hence we decide to adopt sampling strategy to avoid these problems. A common

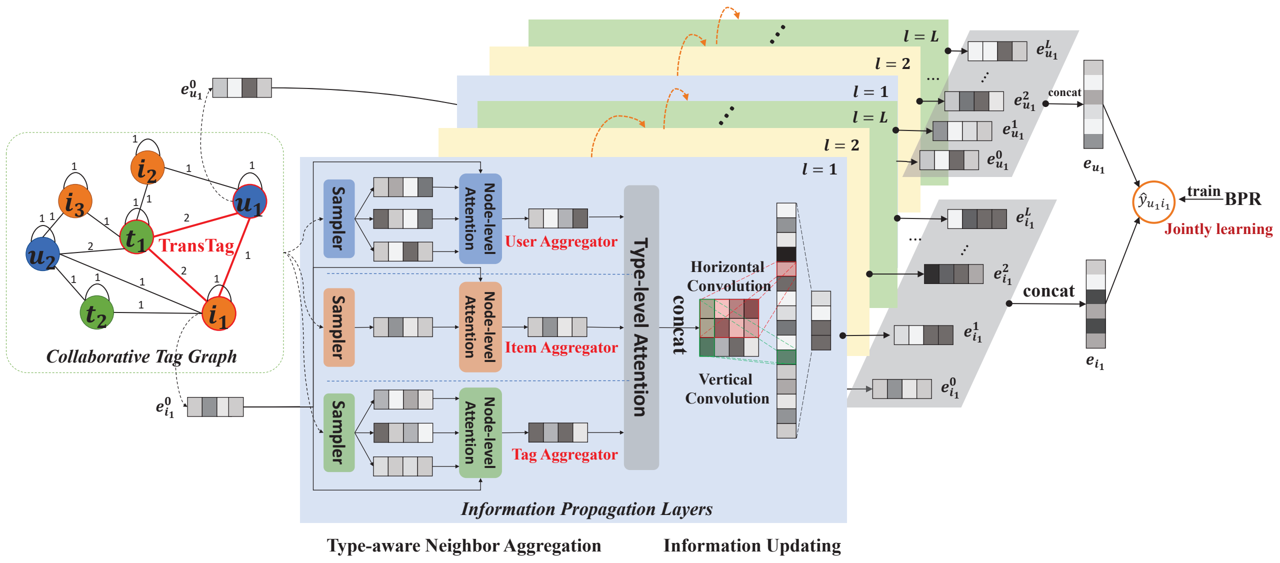


Figure 2: The architecture of TGCN.

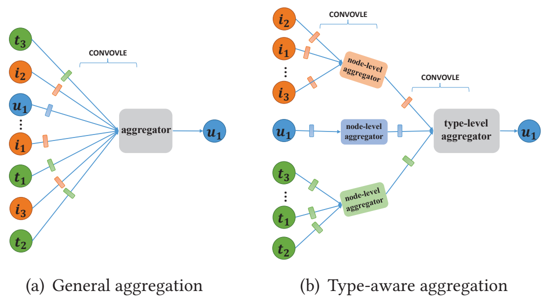


Figure 3: The process of neighbor sampling and aggregation.

approach is random sampling on neighbors, like GraphSage [7]. However, this approach is sub-optimal in the heterogeneous graph. Random sampling on heterogeneous nodes cannot guarantee that all types of neighbors will be sampled, making it difficult to preserve intrinsic features for some types with fewer nodes.

In light of this issue, we design a heterogeneous random sampling strategy. Specifically, for the k-th node type ($k \in \{1, \cdots, K\}, K = 3$), a fixed-size set of neighbors with node type k is sampled uniformly at random (with replacement) as the receptive field. We use $\mathcal{N}_v^k$ to denote the sampled fixed-size neighbors with type k for central node v, *i.e.*, $\mathcal{N}_v^k = \{j | e(v, j, w) \in \mathcal{E} \wedge \phi(j) = k\}$. Therefore, given a central node v, the sampled neighbors sets with user, item and tag type can be represented as $\mathcal{N}_v^u$, $\mathcal{N}_v^i$ and $\mathcal{N}_v^t$, respectively.

4.1.2 Neighbor Aggregation. After sampling neighbors of different node types, the next step is to aggregate neighbors of the same type and obtain type-specific neighborhood representations. In particular, the aggregated type k neighborhood representation $\mathbf{e}_{\mathcal{N}_v^k}$ is given by a weighted sum aggregator:

$$\mathbf{e}_{\mathcal{N}_v^k} = \sum_{j \in \mathcal{N}_v^k} \alpha_{v \leftarrow j} \mathbf{e}_j, \tag{1}$$

where $\alpha_{v \leftarrow j}$ is the weight assigned to neighbor j for indicating its importance. To calculate the weight, one of the most pervasive approach is to set $\alpha_{v \leftarrow j} = \frac{1}{|\mathcal{N}_v^k|}$. However, this approach assumes that all neighbors have equal importance, which is unreasonable.

To alleviate this limitation, we leverage the attention mechanism to compute weights adaptively. The motivation is that informative neighbors will contribute more to provide salient features while irrelevant neighbors will be largely neglected. Specially, we parameterize the node-level attention with a two-layer neural network, which takes both node representations and edge weight into consideration, shown in Eq.(2). The attention network is not only aware of the central and neighbor nodes, but also consider their interaction strength (edge weight), making the entire modeling process more comprehensive.

$$a_{v \leftarrow j} = \mathbf{v}_k^T \mathtt{ReLU}(\mathbf{W}_1^k [\mathbf{e}_v \| \mathbf{e}_w] + \mathbf{W}_2^k \mathbf{e}_j + \mathbf{b}^k), \tag{2}$$

where $\|$ denotes the concatenation operation, $\mathbf{e}_v, \mathbf{e}_j \in \mathbb{R}^d$ are the embeddings of central node v and neighbor j, $\mathbf{e}_w \in \mathbb{R}^b$ is the embedding of edge weight w assigned to edge $e(v, j, w)$ which is partitioned into discrete buckets in the pre-processing. Besides, $\mathbf{W}_1^k$, $\mathbf{W}_2^k$, $\mathbf{b}^k$ and $\mathbf{v}_k$ are the type-specific trainable parameters for the k-th node-level attention. The underlying intuition to employ type-aware aggregators is the intrinsic features of the different types can be learned separately with fine granularity. Finally, the attention weight $\alpha_{v \leftarrow j}$ is obtained by normalizing the above attentive score using softmax, *i.e.*, $\alpha_{v \leftarrow j} = \mathtt{softmax}(a_{v \leftarrow j})$.

4.2 Information Updating (C2)

Once different types of neighborhood are represented, the next move is to fuse these type-specific neighborhood representations (including nodes' self information) and update node information. Existing heterogeneous graph convolutional models generally leverage a pooling-based function (*e.g.*, attentive-pooling [34]) to update node information, neglecting the feature interactions that is critical in recommender systems [6, 15]. To cope with challenge **C2**, we propose an attentive CNN-based information updating method to capture multi-granular feature interactions.

4.2.1 Type-level Information Re-scale. An intuition is different nodes may have different emphasis on different types of information. Therefore, in order to distill conducive type features adaptively, we employ a type-level attention network to re-scale the weights of the type-specific neighborhood representations and emphasize more beneficial signals. By assigning the informative type information with a higher weight, the attention mechanism is capable of suggesting which types to focus on. Specifically, the attention score is calculated via a two-layer network, shown as:

$$b_k = \mathbf{p}^T \mathsf{ReLU}(\mathbf{U}\mathbf{e}_{\mathcal{N}_v^k} + \mathbf{q}), \tag{3}$$

where $\mathbf{U}$, $\mathbf{q}$ and $\mathbf{p}$ are the trainable parameters. Similarly, the attention weight can be obtained by softmax, *i.e.*, $\beta_k = \mathtt{softmax}(b_k)$.

Generally speaking, existing heterogeneous GNN-based models leverage a pooling-based function to fuse and update information crudely, *e.g.*, $\mathbf{e}_v = \sum_{k=1}^{K} \beta_k \mathbf{e}_{\mathcal{N}_v^k}$ in HetGNN [34]. However, we argue that these methods do not take into account feature interactions, which is critical to recommender systems [6, 15]. In light of this defect, we concatenate them as a 2D matrix, *i.e.*, $\mathcal{M}_v = \|_{k=1}^{K} \beta_k \mathbf{e}_{\mathcal{N}_v^k}$, and employ a CNN-based method to capture feature interactions.

4.2.2 Feature Interaction Extraction. To capture the informative multi-granular feature interactions among different types of neighborhood representations, we creatively employ CNN as type-level aggregator to process 2D information matrix $\mathcal{M}_v \in \mathbb{R}^{K\times d}$, which are widely used in image recognition [13] and natural language processing [10]. Inspired by CCPM [20] and Caser [26] in feature interactions modeling, we design two kinds of convolution with different receptive fields to capture different feature interactions. More precisely stated, "vertical filters" with shape $K \times 1$ slide over the columns of $\mathcal{M}_v$ for extracting *bit-level feature interactions* while "horizontal filters" with shape $h \times d$ capture h-order *vector-level feature interactions*.

Bit-level Feature Interactions. To extract prominent feature interactions from the perspective of embedding dimension, we employ vertical convolution filters to extract bit-level feature interactions. Vertical convolution filters cover all the three node types and slide along the embedding dimension direction, denoted as $V^t \in \mathbb{R}^{K\times 1}$, $1 \le t \le m$. Each filter V^t interacts with each column of matrix $\mathcal{M}_v$ by sliding d times from left to right, yielding convolution result c^t.

$$c^t = [c_1^t, c_2^t, \cdots, c_i^t, \cdots, c_d^t]. \tag{4}$$

The i-th convolution value c_i^t can be represented as:

$$c_i^t = \mathsf{ReLU}(\mathcal{M}_v[:, i] \odot V^t), \tag{5}$$

where $\odot$ represents the inner product operator. Particularly, the vertical convolution result is equal to the weighted sum over the K rows of $\mathcal{M}_v$ weighted by filter V^t, *i.e.*, $c^t = \sum_{k=1}^{K} V^t[k, :] * \mathcal{M}_v[k, :]$. Therefore, performing vertical convolution with m filters on information matrix $\mathcal{M}_v$ contributes to extract different bit-level feature interactions in m different subspaces.

Finally, the results of m vertical filters are concatenated into a vector o_v:

$$o_v = c^1 \| c^2 \| \cdots \| c^t \| \cdots \| c^m. \tag{6}$$

Vector-level Feature Interactions. Modeling vector-level feature interactions is the core of many recommendation models (*e.g.*, DeepFM [6]), which exploit a feed-forward neural network on the embedding vector for fully capturing implicit features. To achieve this goal, we design horizontal convolution filters to slide along the type direction on the 2D matrix $\mathcal{M}_v$ and interact with all horizontal dimensions. To be specific, n horizontal filters are used, *i.e.*, $H^t \in \mathbb{R}^{h\times d}$, where $1 \le t \le n$ and $h \in \{1, \cdots, K\}$ is the height of a filter. The i-th convolution value $\tilde{c}_i^t$ can be represented as:

$$\tilde{c}_i^t = \mathsf{ReLU}(\mathcal{M}_v[i : i + h - 1, :] \odot H^t). \tag{7}$$

Therefore, the convolutional result of filter H^t is:

$$\tilde{c}^t = [\tilde{c}_1^t, \tilde{c}_2^t, \cdots, \tilde{c}_i^t, \cdots, \tilde{c}_{K-h+1}^t]. \tag{8}$$

Similarly, the results of n horizontal filters are concatenated into a vector o_h, which is shown as:

$$o_h = \tilde{c}^1 \| \tilde{c}^2 \| \cdots \| \tilde{c}^t \| \cdots \tilde{c}^n. \tag{9}$$

Finally, the bit-level and vector-level feature interactions signals are concatenated and fed into a full-connected (FC) layer to extract high-level global interactive features, represented as:

$$\mathbf{e}_v = \mathsf{ReLU}(\mathbf{W}_f [o_v \| o_h] + \mathbf{b}_f), \tag{10}$$

where $\mathbf{W}_f$ and $\mathbf{b}_f$ are the weight matrix and bias vector.

To sum up, the major advantages of using CNN&FC to model feature interactions lie in follows: 1) Vertical convolution extracts local features on each dimension, working as a multi-head feature extractor to learn multi-aspect bit-level feature interactions. 2) Horizontal convolution interacts with every successive h neighborhood representations, capturing h-order vector-level feature interactions among different type representations. 3) FC layer recombines the multi-granular feature interaction patterns and learn high-level global interactive features.

4.2.3 High-order Propagation. We group the above two successive phases (*i.e.*, type-aware neighbor aggregation and information updating) into an *information propagation layer.* By stacking multiple information propagation layers, we can explore the higher-order connectivity inherent in the CTG and obtain expressive node representations. Specifically, we stack L layers and each node obtains L intermediate representations, namely $\{\mathbf{e}_v^1, \mathbf{e}_v^2, ..., \mathbf{e}_v^L\}$, which integrate different connectivity information.

To integrate different connectivities and make full use of the process of information propagation, we leverage the layer-wise concatenation mentioned in JK-Net [30] to generate the final representation $\mathbf{e}_v = \mathbf{e}_v^0 \| \mathbf{e}_v^1 \| \mathbf{e}_v^2 \| ... \| \mathbf{e}_v^L$. Finally, the predicted score can be obtained by the inner product of user and item representations, shown as:

$$\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i. \tag{11}$$

4.3 TransTag Regularization (C3)

As is well-known, item attributes and user interests are ***diverse***. For example, a movie has the attributes of directors, plot, theme, actors, etc. Moreover, different users chose the same movie for different reasons, which can be reflected by Figure 4. Form *John*'s perspective, he chose movie *Transformers* due to the consideration of *"science*

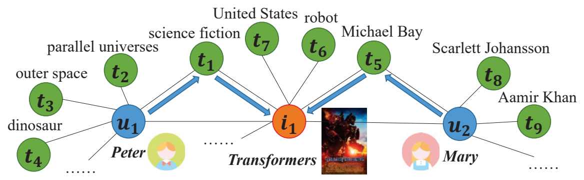


Figure 4: An example of users' substantive preference identification.

fiction" theme, which is different from *Mary* (for director *"Michael Bay"*). These ***diverse*** preferences can be ***locally connected*** via the tags, thus conducing to identify user's substantive preference over given certain items. However, these underlying semantics cannot be well modeled via graph convolution. To capture the semantics in user-tag-item triplets, identify user's substantive preference and solve challenge **C3**, we propose TransTag regularization function.

TransTag regularization function is ignited by TransE [2], a widely used method in knowledge graph embedding, which projects both entities and relations into a latent embedding space. The assumption of TransTag regularization function is that user u chose item i from the perspective of tag t, which requires the embedding of item $\mathbf{e}_i$ to be a nearest neighbor of $\mathbf{e}_u + \mathbf{e}_t$ distinctly, *i.e.*, $\mathbf{e}_u + \mathbf{e}_t \approx \mathbf{e}_i$. Hence, for a given triplet in the training assignment set $a = \langle u, t, i \rangle \in \mathcal{A}_{train}$, its energy score is formulated as follows:

$$g(u, t, i) = \|\mathbf{e}_u + \mathbf{e}_t - \mathbf{e}_i\|_2 , \tag{12}$$

where $\mathbf{e}_u, \mathbf{e}_t, \mathbf{e}_i \in \mathbb{R}^d$ are the embeddings of user, tag and item respectively.

To train TransTag regularization function, we minimize a margin-based pairwise ranking loss over $\mathcal{A}_{train}$:

$$\mathcal{L}_E = \sum_{\langle u,t,i\rangle \in \mathcal{A}_{train}, \langle u',t,i'\rangle \in \mathcal{A}'_{\langle u,t,i\rangle}} \texttt{ReLU}(\gamma + g(u, t, i) - g(u', t, i')), \tag{13}$$

where γ is a margin hyper parameter. Corrupted triplet $\langle u', t, i' \rangle$ is random sampled from $\mathcal{A}'_{\langle u,t,i\rangle}$, which is constructed by replacing either the user or item node in $\langle u, t, i \rangle$. By doing this, tag can be viewed as a relation linking user and item, revealing user's substantive preference over certain items. TransTag regularization function models the nodes on the granularity of annotation triples, working as a regularizer on the node learning procedure.

4.4 Jointly Training Details

To better perform ranking, we optimize the recommendation task with the BPR framework [22].

$$\mathcal{L}_G = \sum_{(u,i_+) \in \mathcal{P}_{train}, (u,i_-) \in \mathcal{P}'_u} - \ln \texttt{sigmoid}(\hat{y}_{ui_+} - \hat{y}_{ui_-}), \tag{14}$$

where $\mathcal{P}_{train}$ indicates the training set while $\mathcal{P}'_u$ means the unobserved item set for user u.

To effectively learn parameters for recommendation as well as preserve the regularization relationship among annotation triples, we integrate the recommendation task and the TransTag regularization in an end-to-end fashion through a jointly learning framework. Finally, the total objective function of TGCN is defined as Eq.(15).

$$\mathcal{L} = \mathcal{L}_E + \mathcal{L}_G + \lambda \|\Theta\|_2, \tag{15}$$

where λ and Θ is the regularization weight and model parameters, respectively. We optimize $\mathcal{L}_E$ and $\mathcal{L}_G$ alternatively, where mini-batch Adam [11] is adopted. Besides, dropout [24] and early stopping strategy are also applied to avoid over-fitting.

4.5 Discussion

To compare the state-of-the-art graph convolution models from a macro perspective, we perform the discussion from five technical dimensions. The comparison results are shown in Table 1.

1) For neighbor sampling, PinSage performs random sampling and HetGNN uses heterogeneous neighbor sampling based on random walk. Considering both efficiency and effectiveness, TGCN leverages heterogeneous random sampling strategy. 2) Both HGAT and TGCN employ hierarchical attention for discriminating the importance of neighbors and node types. However, TGCN takes both central and neighbor nodes as well as the interaction strength (edge weight) into consideration in Eq.(2), portraying the importance of neighbors comprehensively. 3) Among the compared models, only HGAT and HetGNN consider heterogeneous graph modeling and perform type-aware neighbor aggregation. 4) Although NGCF and KGAT model the feature interactions via element-wise product, they can only perceive coarse-grained global features. In comparison, TGCN utilizes the CNN to capture multi-granular feature interactive signals. 5) KGAT employs TransR [17] to regularize the knowledge-based relations. Similarly, TGCN leverages the TransTag to identify user's substantive preference over certain items and regularize tag-based annotation triples.

Table 1: Model comparison: 1) SP - sampling. 2) AM - attention mechanism. 3) HG - heterogeneous graph. 4) FI - feature interaction. 5) RL - regularization.

Property	PinSage	NGCF	GAT	KGAT	HGAT	HetGNN	TGCN
SP	✓	✗	✗	✗	✗	✓	✓
AM	✗	✗	✓	✓	✓	✓	✓
HG	✗	✗	✗	✗	✓	✓	✓
FI	✗	✓	✗	✓	✗	✗	✓
RL	✗	✗	✗	✓	✗	✗	✓

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Data Description and Evaluation Protocols. To evaluate the performance of the proposed TGCN, we conduct comprehensive experiments on three public real-world datasets: MovieLens, Last.Fm and Delicious, which are all released in HetRec 2011 [4].

- **MovieLens** is a movie recommendation dataset published by GroupLens research group[1]. In this dataset, a list of tag assignments for interacted movies is bound to each user.
- **Last.Fm** is an artist recommendation dataset obtained from online music system Last.Fm[2]. In this dataset, each user has a list of tag assignments to artists.
- **Delicious** dataset is gathered from Del.icio.us[3] system, which encourages users to tag bookmarks. In this dataset, each user has a list of tag assignments to web bookmarks.

[1]http://www.grouplens.org
[2]http://www.last.fm.com
[3]http://delicious.com

For a fair comparison, we adopt the same preprocessing as existing research [31, 36] to remove infrequent tags that are used less than 5 times in MovieLens and Last.Fm, and 15 times in Delicious respectively. Table 2 summarizes the statistics of these datasets. For each user, we randomly select 80% of the interacted items as training set and the remaining 20% as testing set. Assignments associated with the items in training set are used to construct the CTG.

The top-N recommendation quality is evaluated by four metrics: Precision, Recall, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [8], which are generally used in recommender systems to evaluate the performance of models [28, 29, 31].

Table 2: Datasets statistics.

Dataset	#Users	#Items	#Tags	#Assignments
MovieLens	1,651	5,381	1,586	36,728
Last.Fm	1,808	12,212	2,305	175,641
Delicious	1,843	65,877	3,508	339,744

5.1.2 Baselines and Parameter Settings. To demonstrate the effectiveness, we compare our proposed TGCN with feature-based (CFA, DSPR and DeepFM) and GNN-based (PinSage, NGCF, KGAT and HGAT) models. All the GNN-based baselines are extended to perform representation learning on CTG with tags integrated.

- **CFA**: CFA uses a sparse autoencoder to obtain latent representations of user profiles, on which user-based CF is applied for recommendation [36].
- **DSPR**: DSPR leverages MLPs with shared parameters to process tag-based features for extracting user and item representations [31].
- **DeepFM**: DeepFM combines factorization machines [21] and deep neural network for feature learning and recommendation [6].
- **PinSage**: PinSage deploys GraphSage [7] on industrial application and obtains node representations via non-spectral graph convolution [33].
- **NGCF**: NGCF explicitly encodes the collaborative signal in the form of high-order connectivities by performing embedding propagation [29].
- **KGAT**: KGAT, state-of-the-art KG-based model, performs knowledge-aware attentive graph convolution in knowledge graph for high-order relation modeling [28].
- **HGAT**: HGAT, state-of-the-art HG-based model, leverages a dual-level attention network to aggregate heterogeneous neighbors from both type and node levels [18].

In parameter settings, we optimize all models with mini-batch Adam, where the batch size is fixed at 512 and the learning rate is searched from $\{0.0001, 0.001, 0.01, 0.05\}$. The dropout ratio is tuned in $\{0.1, 0.2, \cdots, 0.9\}$ and the number of neighbors sampled is set to 25. Besides, the embedding size of nodes and attention factor is fixed to 64 and 32 respectively, and the embedding size of edge weights is 10. For TGCN, we set the depth as 3 with tower hidden dimension 64, 32 and 16 by default. The number of filters in vertical and horizontal convolution is 32 and 24 (3×8), respectively. The margin hyper parameter is set to 1 and the normalization coefficient is tuned in $\{10^{-4}, 10^{-3}, \cdots, 10^{1}\}$.

5.2 Performance Comparison

Table 3 shows the top-N recommendation performance on all three datasets, where $N \in \{10, 20\}$. It's obvious that TGCN consistently outperforms other methods over all evaluation metrics. Compared with the tag-aware models (*i.e.*, CFA and DSPR), our model has made a significant improvement. An obvious phenomenon is that, as the size of items and users increases, the recommendation performance of the feature-based methods decreases markedly compared with TGCN, which indicates that the existing tag-aware models are unsuitable for large-scale datasets (*e.g.*, Delicious). The reason is that the features (tags) for describing large-scale candidate items and users are insufficient, which is difficult to tackle the issues of sparsity, ambiguity and redundancy, limiting the performance of feature-based models fatally. However, GNN-based models leverage multi-hop neighbors to provide more contextual semantics, alleviating these problem and achieving superior results.

Among the GNN-based baselines, NGCF considers the feature interactions between central node and neighbors, achieving better performance than PinSage. Besides, HGAT leverages hierarchical attention to identify the importance of both node types and heterogeneous neighbors, obtaining best results on Movielens and Last.Fm. In comparison, KGAT obtains less satisfactory results because the special knowledge-based relation modeling and knowledge-aware attentive mechanism are less applicable to CTG, illustrating the importance of model customization in different scenarios. In contrast, TGCN utilizes type-aware aggregators to integrate heterogeneous neighbors in CTG and further employs CNN-based information updating method to capture multi-granular feature interactions, learning superior node representations.

5.3 Detailed Study of TGCN

5.3.1 Effect of type-aware neighbor aggregation. To verify the effect of type-aware neighbor sampling and aggregation strategy, we design two groups of experiment, whose results are shown in Table 4, where R is short for Recall and NG is short for NDCG. To illustrate the validity of heterogeneous neighbor sampling, we replace it with random sampling used in GraphSage [7] regardless of the node types, termed it as *TGCN-RandomSP*. Experimental results show that heterogeneous random sampling strategy performs better than random sampling. The main reasons are two folds: 1) Heterogeneous sampling ensures that all types of neighbors are obtained at each round of sampling; 2) Sampling a fixed number of neighbors for each type makes it possible to fully represent some types of features with fewer nodes.

To further demonstrate the effectiveness of type-aware neighbor aggregation, we design experiment to aggregate neighbors regardless of their node types, termed it as *TGCN-w/o HG*. Evaluation results demonstrate that ignoring node types will degrade the recommendation performance significantly. Therefore, sampling and aggregating heterogeneous neighbors differently and separately contributes to capture their intrinsic differences.

5.3.2 Effect of two-layer attention. To investigate the effect of two-layer attention, we do some ablation study by removing node-level attention and setting Eq.(2) $\alpha_{v \leftarrow j} = \frac{1}{|\mathcal{N}_v^k|}$, termed *TGCN-w/o NA*; removing type-level attention and setting Eq.(3) $\beta_k = \frac{1}{K}$, termed

Table 3: Comparison between different models. Boldface denotes the highest score and underline indicates the best result of the baselines

Dataset	Metric	Feature-based			GNN-based				TGCN	Impr.
		CFA	DSPR	DeepFM	PinSage	NGCF	KGAT	HGAT		
MovieLens	Precion@10	0.0292	0.0410	$\underline{0.0411}$	0.0321	0.0348	0.0345	0.0357	**0.0423**	+2.9%
	Precion@20	0.0235	0.0323	$\underline{0.0327}$	0.0261	0.0278	0.0272	0.0284	**0.0345**	+5.5%
	Recall@10	0.0422	0.0690	0.0698	0.0564	0.0572	0.0566	$\underline{0.0782}$	**0.0837**	+7.0%
	Recall@20	0.0623	0.0914	0.0991	0.0868	0.0937	0.0913	$\underline{0.1129}$	**0.1207**	+6.9%
	HR@10	0.1962	0.2359	$\underline{0.2612}$	0.2129	0.2264	0.2254	0.2401	**0.2672**	+2.3%
	HR@20	0.2526	0.2860	$\underline{0.3278}$	0.2880	0.2943	0.2881	0.3235	**0.3361**	+2.5%
	NDCG@10	0.0813	0.1137	$\underline{0.1204}$	0.0932	0.1005	0.1002	0.1102	**0.1234**	+2.5%
	NDCG@20	0.1049	0.1389	$\underline{0.1518}$	0.1215	0.1308	0.1272	0.1398	**0.1561**	+2.8%
Last.Fm	Precion@10	0.0633	0.0610	0.0645	0.0691	$\underline{0.0772}$	0.0709	0.0742	**0.0831**	+7.6%
	Precion@20	0.0469	0.0490	0.0501	0.0548	$\underline{0.0583}$	0.0572	0.0579	**0.0631**	+8.2%
	Recall@10	0.0973	0.0727	0.0957	0.1101	0.1036	0.1030	$\underline{0.1107}$	**0.1185**	+7.0%
	Recall@20	0.1185	0.1144	0.1400	0.1552	0.1533	0.1538	$\underline{0.1594}$	**0.1716**	+7.7%
	HR@10	0.3263	0.3363	0.3746	0.4029	0.4020	0.4022	$\underline{0.4065}$	**0.4275**	+5.2%
	HR@20	0.4175	0.4366	0.4712	0.4886	0.4913	0.4895	$\underline{0.4995}$	**0.5268**	+5.5%
	NDCG@10	0.1583	0.1491	0.1788	0.1928	0.1937	0.1868	$\underline{0.1945}$	**0.2073**	+6.6%
	NDCG@20	0.2017	0.1949	0.2233	0.2377	0.2395	0.2343	$\underline{0.2404}$	**0.2563**	+6.6%
Delicous	Precion@10	0.0051	0.0212	0.0090	0.1416	$\underline{0.1469}$	0.1288	0.1291	**0.1544**	+5.1%
	Precion@20	0.0036	0.0169	0.0071	0.1027	$\underline{0.1062}$	0.0975	0.0959	**0.1158**	+9.0%
	Recall@10	0.0069	0.0219	0.0121	0.1372	$\underline{0.1470}$	0.1211	0.1306	**0.1504**	+2.3%
	Recall@20	0.0098	0.0349	0.0175	0.1925	$\underline{0.2026}$	0.1810	0.1916	**0.2106**	+3.9%
	HR@10	0.0451	0.1069	0.0783	0.3941	$\underline{0.4171}$	0.3947	0.4026	**0.4263**	+2.1%
	HR@20	0.0635	0.1567	0.1104	0.4227	$\underline{0.4450}$	0.4302	0.4378	**0.4547**	+2.2%
	NDCG@10	0.0222	0.0510	0.0389	0.2180	$\underline{0.2232}$	0.1983	0.2023	**0.2282**	+2.2%
	NDCG@20	0.0270	0.0688	0.0487	0.2657	$\underline{0.2714}$	0.2498	0.2570	**0.2831**	+4.3%

Table 4: Effect of type-aware neighbor aggregation.

Model	MovieLens		Last.Fm		Delicious	
	R@10	NG@10	R@10	NG@10	R@10	NG@10
RandomSP	0.0696	0.1064	0.1124	0.2005	0.1457	0.2204
w/o HG	0.0703	0.1073	0.1110	0.1981	0.1419	0.2167
TGCN	**0.0837**	**0.1234**	**0.1185**	**0.2073**	**0.1504**	**0.2282**

Table 5: Effect of two-layer attention.

Model	MovieLens		Last.Fm		Delicious	
	R@10	NG@10	R@10	NG@10	R@10	NG@10
w/o NA	0.0755	0.1188	0.1163	0.1996	0.1465	0.2235
w/o CA	0.0721	0.1126	0.1154	0.1982	0.1399	0.2187
w/o A	0.0712	0.1101	0.1137	0.1953	0.1386	0.2178
TGCN	**0.0837**	**0.1234**	**0.1185**	**0.2073**	**0.1504**	**0.2282**

TGCN-w/o CA; and removing both node-level and type-level attention, termed *TGCN-w/o A*. Table 5 summarizes the experimental results. It is obvious that removing either node-level or type-level attention compromises the expressive ability of model adversely. Node-level attention takes both node representations and interaction strength into consideration when measuring the importance of neighbors. Besides, type-level attention re-scales the weights of type-specific neighborhood representations, distilling conducive type features adaptively. By combining node-level and type-level hierarchical attention, TGCN can distinguish the importance of heterogeneous neighbors in a fine-grained manner.

5.3.3 *Effect of convolution updating.* To demonstrate the effectiveness of convolution updating strategy, we perform some comparative experiments. In particular, mean pooling and MLP are two commonly used updating methods. Accordingly, we propose two variants *TGCN-pooling* and *TGCN-MLP* by replacing the CNN with mean pooling and MLP, respectively. Besides, we also investigate the effect of vertical and horizontal convolution separately, namely *TGCN-v-CNN* and *TGCN-h-CNN*. Figure 5 illustrates the Recall@10 experimental results. We can find that, compared with pooling and MLP updating functions, TGCN achieves the best results. The reason is that TGCN employs both vertical and horizontal convolutions for capturing bit-level and vector-level feature interactions, thus obtaining informative node representations.

5.3.4 *Effect of TransTag.* To verify the effect of TransTag regularization function, we disable the TransTag and term it as *TGCN-w/o RL*. Figure 6 shows the performance *w.r.t.* recall@10 of each epoch on Movielens and Last.Fm, from which we observe that TransTag is conducive to boost the recommendation performance in TRS because the translation assumption acts as a beneficial regularizer for constraining annotation triplets. By optimizing recommendation

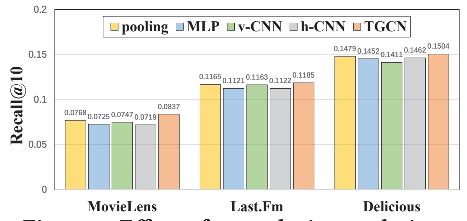
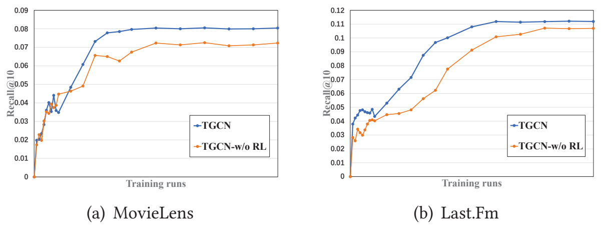


Figure 5: Effect of convolution updating.



(a) MovieLens (b) Last.Fm

Figure 6: Effect of TransTag.

Table 6: Effect of information propagation layer numbers.

Model	MovieLens		Last.Fm		Delicious	
	R@10	NG@10	R@10	NG@10	R@10	NG@10
TGCN-1	0.0706	0.1128	0.1085	0.1926	0.1431	0.2225
TGCN-2	0.0769	0.1223	0.1128	0.1963	0.1470	0.2254
TGCN-3	**0.0837**	**0.1234**	0.1185	0.2073	0.1504	0.2282
TGCN-4	0.0767	0.1204	**0.1240**	**0.2078**	**0.1564**	**0.2327**

and TransTag in a jointly learning framework, user's substantive preference over certain items can be well identified.

5.4 Parameters Sensitivity

5.4.1 Number of Layers. To explore the influence of information propagation layer numbers on the recommendation performance, we search the depth L of TGCN in the range of $\{1, 2, 3, 4\}$. The experimental results are summarized in Table 6, from which we have the following observations: 1) Increasing the layers within certain limits contributes to improve the recommendation performance substantially. The reason is that stacking more layers helps nodes to touch the distant multi-hop neighbors, enabling higher-order connectivities. 2) However, stacking too many layers may make the neighborhood features tend to be similar and reduce the performance, which is more obvious in some smaller graphs such as MovieLens dataset.

5.4.2 Number of Samples. To evaluate the impact of the number of samples, we vary the number in the range of $\{1, 5, 10, 15, 20, 25, 30\}$. Figure 7 plots the effect of samples on Movielens and the similar results on other datasets are omitted due to the space limitation. We can observe that increasing neighbor samples can improve the performance. The reason lies in the introduction of richer features brought by more neighbors. Considering both recommendation effect and training efficiency, we set the neighbors sampled to 25.

5.5 Model Complexity

In order to quantitatively analyze the space and time complexity of our proposed TGCN, we compare the model parameters and inference time with some GNN-based models. Table 7 reports the comparison results on Delicious dataset. We can observe that, compared with other models, the increase of TGCN model parameters is acceptable *w.r.t.* 5.2% on average. The reason is that the overwhelming majority of model parameters are concentrated in the node embeddings and the parameters in the graph convolution are negligible. As online recommendation services usually have high requirements on latency, the computational cost during inference is more important than that of training phase. From Table 7 we can find that TGCN achieves comparable inference time to other GNN-based models, indicating the efficiency of our model.

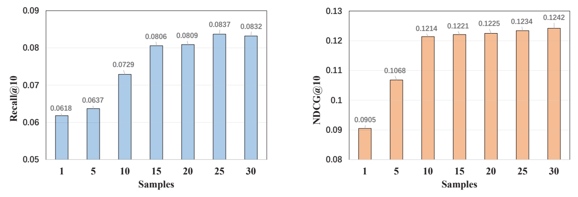


Figure 7: Effect of the number of samples on Movielens.

Table 7: Complexity Analysis on Delicious dataset

Model	Parameters ($\times 10^6$)	Relative ratio	Inference time (s)	Relative ratio
PinSage	~4.56	+5.3%	~5.3	+9.4%
NGCF	~4.57	+5.1%	~5.2	+11.5%
HGAT	~4.56	+5.3%	~5.9	-1.7%
TGCN	~4.80	-	~5.8	-

Table 8: Industrial application study.

Model	Recall@10	NDCG@10
DeepFM	0.1940	0.1772
PinSage	<u>0.3390</u>	<u>0.3090</u>
TGCN	**0.3606**	**0.3299**
Impr.	+6.4%	+6.8%

5.6 Industrial Application

To further verify the effectiveness of TGCN, we apply it in the recommender system of a mainstream App store. We sample and collect records from the App store to form an industrial dateset, which is composed of 10,456 users, 30,488 apps as well as 2,472 tags.

For performance comparison, we select two representative models that are actually applied in our production system as the baselines, *i.e.*, feature-based model DeepFM [6] and GNN-based model PinSage [33]. Table 8 shows the experimental results on our industrial dataset. We can observe that TGCN has better performance than the best baseline (PinSage) in terms of recall by 6.4% and NDCG by 6.8%. As another point, DeepFM performs much worse than PinSage, which suggests that applying feature-based models directly in the tag-aware recommendation may not lead to superior performance. It is difficult for the feature-based models to accurately describe large-scale items and users simply by relying on smaller-scale tags (features), which is similar to the results on the Delicious dataset in Table 3.

5.7 Micro-level Analysis

Leveraging the contextual semantics of multi-hop neighbors in CTG to alleviate the synonymy redundancy problem is a key advantage of GNN-based models. Therefore, we perform some micro-level analysis by investigating the embeddings of synonymous tags. We randomly select five tag pairs with similar semantics and compare their cosine similarities of the embeddings. Table 9 shows three different types of tag pairs, namely synonymous tags caused by individual diversity in writing (*e.g.*, "*5 stars*" and "*5-star*") and expression (*e.g.*, "*kids*" and "*children*"), as well as related tags (*e.g.*, "*japan*" and "*tokyo*"). We can find that, benefit from the additional contextual semantics of multi-hop neighbors, GNN-based model (TGCN) obtains more similar tag embeddings than feature-based model (DeepFM). Due to the similar neighborhood subgraphs in CTG, these tag pairs gain more approximate node embeddings, which is conducive to better depicting users and items.

Table 9: Tag embedding micro-level analysis.

Tag 1	Tag 2	TGCN	DeepFM
5 stars	*5-star*	0.621	0.242
80s	*80's*	0.384	0.109
kids	*children*	0.576	0.078
violent	*brutal*	0.683	-0.061
japan	*tokyo*	0.338	0.041

6 CONCLUSION

In this paper, we delve into the issues (*i.e.*, sparsity, ambiguity and redundancy), that restrict the development of existing feature-based tag-aware recommendation and construct CTG to alleviate these issues. To learn superior node representations, we propose a novel tag-aware recommendation model TGCN for personalized top-*N* recommendation. To cope with the heterogeneity of graph nodes, TGCN employs type-aware neighbor sampling and aggregation operation to learn the type-specific neighborhood representations. For effectively modeling multi-granular feature interactions, TGCN performs vertical and horizontal convolutions on the multi-type information matrix after attentive type-level information re-scale. Finally, we propose TransTag regularization function and perform jointly learning to identify user's substantive preference accurately. Extensive experiments demonstrate that TGCN achieves remarkable performance improvement compared with state-of-the-art models. Future work includes introducing meta-path for guiding graph convolution. Besides, we are also interested in exploring a non-uniform sampling strategy for heterogeneous nodes.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. In *SIGKDD Workshop on Deep Learning Day.*

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS.* 2787–2795.

[3] Iván Cantador, Alejandro Bellogín, and David Vallet. 2010. Content-based recommendation in social tagging systems. In *RecSys.* 237–240.

[4] Ivan Cantador, Peter L Brusilovsky, and Tsvi Kuflik. 2011. *Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011).*

[5] Chaochao Chen, Xiaolin Zheng, Yan Wang, Fuxing Hong, Deren Chen, et al. 2016. Capturing Semantic Correlation for Item Recommendation in Tagging Systems.. In *AAAI.* 108–114.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI.* 1725–1731.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS.* 1024–1034.

[8] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20, 4 (2002), 422–446.

[9] Jason J Jung. 2011. Discovering community of lingual practice for matching multilingual tags from folksonomies. *Comput. J.* 55, 3 (2011), 337–346.

[10] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP.* 1746–1751.

[11] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR.*

[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR.*

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS.* 1097–1105.

[14] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI.* 1–8.

[15] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD.* 1754–1763.

[16] Nan Liang, Hai-Tao Zheng, Jin-Yuan Chen, Arun Sangaiah, and Cong-Zhi Zhao. 2018. TRSDL: Tag-Aware Recommender System Based on Deep Learning–Intelligent Computing Systems. *Applied Sciences* 8, 5 (2018), 799.

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI.* 2181–2187.

[18] Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP.* 4823–4832.

[19] Haibo Liu. 2017. Resource recommendation via user tagging behavior analysis. *Cluster Computing* (2017), 1–10.

[20] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. In *CIKM.* 1743–1746.

[21] Steffen Rendle. 2010. Factorization machines. In *ICDM.* 995–1000.

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI.* 452–461.

[23] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys.* 259–266.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[25] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *DLRS.* 11–16.

[26] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM.* 565–573.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR.*

[28] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD.* 950–958.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR.* 165–174.

[30] 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.*

[31] Zhenghua Xu, Cheng Chen, Thomas Lukasiewicz, Yishu Miao, and Xiangwu Meng. 2016. Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling. In *CIKM.* 1921–1924.

[32] Zhenghua Xu, Thomas Lukasiewicz, Cheng Chen, Yishu Miao, and Xiangwu Meng. 2017. Tag-aware personalized recommendation using a hybrid deep model. In *IJCAI.* 3196–3202.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD.* 974–983.

[34] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD.* 793–803.

[35] Zi-Ke Zhang, Chuang Liu, Yi-Cheng Zhang, and Tao Zhou. 2010. Solving the cold-start problem in recommender systems with social tags. *EPL* 92, 2 (2010), 28002.

[36] Yi Zuo, Jiulin Zeng, Maoguo Gong, and Licheng Jiao. 2016. Tag-aware recommender systems based on deep neural networks. *Neurocomputing* 204 (2016), 51–60.