Item Tagging for Information Retrieval: A Tripartite Graph Neural Network based Approach

Kelong Mao
Tsinghua University
Huawei Noah's Ark Lab
mkl18@mails.tsinghua.edu.cn

Xi Xiao
Tsinghua University
Pengcheng Lab
xiaox@sz.tsinghua.edu.cn

Jieming Zhu*
Huawei Noah's Ark Lab
jamie.zhu@huawei.com

Biao Lu
Huawei Noah's Ark Lab
lubiao4@huawei.com

Ruiming Tang
Huawei Noah's Ark Lab
tangruiming@huawei.com

Xiuqiang He
Huawei Noah's Ark Lab
heixiuqiang1@huawei.com

ABSTRACT

Tagging has been recognized as a successful practice to boost relevance matching for information retrieval (IR), especially when items lack rich textual descriptions. A lot of research has been done for either multi-label text categorization or image annotation. However, there is a lack of published work that targets at item tagging specifically for IR. Directly applying a traditional multi-label classification model for item tagging is sub-optimal, due to the ignorance of unique characteristics in IR. In this work, we propose to formulate item tagging as a link prediction problem between item nodes and tag nodes. To enrich the representation of items, we leverage the query logs available in IR tasks, and construct a query-item-tag tripartite graph. This formulation results in a TagGNN model that utilizes heterogeneous graph neural networks with multiple types of nodes and edges. Different from previous research, we also optimize both full tag prediction and partial tag completion cases in a unified framework via a primary-dual loss mechanism. Experimental results on both open and industrial datasets show that our TagGNN approach outperforms the state-of-the-art multi-label classification approaches.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Document representation**.

KEYWORDS

Information retrieval; item tagging; graph neural networks

ACM Reference Format:
Kelong Mao, Xi Xiao, Jieming Zhu, Biao Lu, Ruiming Tang, and Xiuqiang He. 2020. Item Tagging for Information Retrieval: A Tripartite Graph Neural Network based Approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Information retrieval (IR) is a well-established research area that deals with our daily information needs, such as Web search, App search, e-commence product search, image retrieval, music finding, and so on. Although text-based Web search has been widely studied in the literature, IR in vertical domains faces some unique challenges. Different from Web search that mostly deals with full-text documents, textual descriptions of items in some other domains are not sufficiently rich or concise to convey their semantic information. For illustration, we take app search as an example. Figure 1 presents an example app (i.e., Pokemon Go) from Google Play. It consists of multiple types of information including app name, category, developer, screenshot images, and a short description. The description, however, comprises a promotion news only. Such short and noisy item descriptions increases the difficulty for retrieving relevant items.

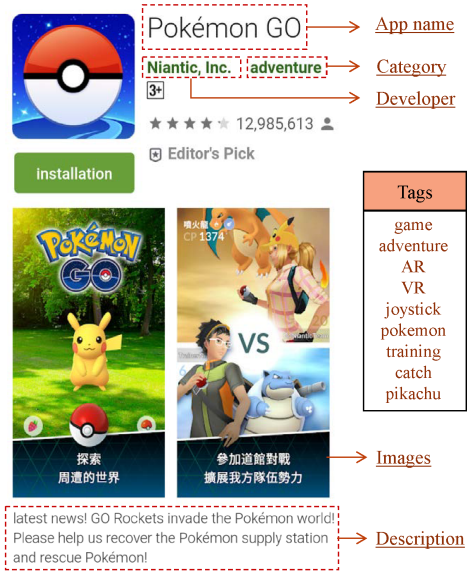


Figure 1: An Illustrative Item Example

In such scenarios, tagging plays a critical role in helping describe and enrich the semantics of items. Tags are often characterized as keywords to describe the key information of items such as category, functionality, style, related entities, target audience, etc. Tagging has been recognized as a successful practice to boost the retrieval performance, especially for those items that lack concise textual

*Corresponding author

Accepted by SIGIR, 2020, 10 pages
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

descriptions [20]. For instance, the app item in Figure 1 has a set of tags including "game", "AR" (Augmented Reality), "pikachu", etc. These tags make it easier to retrieval the app when a user searches the query "pikachu game" or "AR game", but this cannot be done from the textual description only. The collection of tags can not only boost relevance matching, but also be used for query reformulation and item recommendation [14]. In addition, displaying tags and clickable hyperlinks along with their associated items can help users navigate and explore item collections of interest.

For many industrial IR applications, item tagging serves as a key building block for better item organization and retrieval. For user generated content, tags are provided by users themselves for their posts (e.g., tweets hashtags in Twitter, question tags in StackOverflow). In contrast, for platform generated content (i.e., items), such as apps, ads and news, tags and their integration to search may be not visible to users. Item tagging becomes a regular task of operation teams [1]. However, manual tagging is a time-consuming process and might result in unmanageable efforts when the item corpus is too large. To replace or supplement the manual tagging process, a large body of research has been done toward automatic item tagging. Typical examples include app tagging [3], news tagging [21, 23], blog posts tagging [17, 27], questions tagging [22, 28], image annotation [4, 35].

Potential methods for item tagging can be broadly categorized into two types: *keyphrase extraction* [8] and *multi-label classification* [32]. Keyphrase extraction methods (e.g., TF-IDF [19], TextRank [18], PositionRank [5]) have been widely used for textual documents or websites to identify keywords from original content that best describe the subject of a document. These methods mostly follow a two phase procedure (i.e., candidate extraction → ranking). They work well for long documents but are inappropriate for items without detailed textual descriptions, because tags might not appear in the item description. As such, item tagging is often formulated as a multi-label classification problem [32], that is, assigning relevant tags to items from a collection of predefined ones. Multi-label classification models have been widely studied in the literature, and many of them are successfully applied to text categorization [2, 15, 25]. However, directly applying a traditional multi-label classification model for item tagging is sub-optimal, especially in information retrieval tasks.

In this work, inspired by the recent success of graph neural networks (GNN) [29], we propose to cast item tagging as a link prediction problem between item nodes and tag nodes, and present a GNN-based model for item tagging (namely TagGNN). In contrast to previous research, our work aims to address the following limitations.

- Most traditional multi-label classification models cannot fully exploit the correlations among tags (i.e., labels). Instead, our formulation enables tag embedding via node representation, which better captures the correlations among similar tags. It also enriches the representation of item and tag nodes, since semantically similar information can be aggregated from neighbour nodes via message passing. Intuitively, items and tags are matched not only by themselves, but also by neighbour items and neighbour tags.
- Item descriptions are usually short and noisy, making it difficult to extract semantic information from textual descriptions for classification. To alleviate this issue, we propose to not only utilize the textual descriptions, but also leverage the query logs available to enrich the representation of items. We construct a query-item-tag tripartite graph, where query-item edges indicate the interactions (e.g., clicks or downloads) in the query log and item-tag edges represent the annotation relationships. This tripartite graph is unique for IR and leads to heterogeneous GNN modeling with multiple types of nodes and edges. Our TagGNN model naturally fuses item-tag (w.r.t. TagGNN-IT) and query-item (w.r.t. TagGNN-QI) graphs.
- In practice, some new items have no existing tags and need to make full tag prediction. Some old items have partial incomplete tags (e.g., manually labelled), which only need tag completion and refinement. Both cases are desired in IR tasks. While existing work focuses on either one [15] or the other [34], we optimize both cases in a unified framework. To achieve this, we join a primary loss and a dual loss during training to avoid training-testing exposure bias.

We also emphasize that, while some work that leverages GNNs for text categorization exists [9, 10, 30], we are not aware of any published work about GNN-based item tagging that is formulated as a link prediction problem. To evaluate the effectiveness of our TagGNN approach, we conduct comprehensive experiments on two large datasets, including an open dataset of ad tagging for sponsored product search (KDDCup-2012) and a private industrial app tagging dataset for app search (Huawei-Dataset). The experimental results show that our TagGNN approach achieves consistent improvements in precision over 9 baseline models in both "without tags" and "partial tags" settings. Ablation studies and parameter analyses have also been conducted to validate our model design choices.

In summary, our work makes the following main contributions:

- Our work formulates item tagging as a link prediction problem over the query-item-tag graph and present a unique tripartite-graph neural network based approach.
- We target at both full tag prediction and partial tag completion, and present a primary-dual losses to optimize both cases in a unified learning framework.
- Our experimental results show significant improvements over both text-based and graph-based competing methods.

The remainder of this paper is organized as follows. Section 2 describes our TagGNN approach. Section 3 reports on the experimental results. We review the related work in Section 4 and finally conclude the paper in Section 5.

2 TAGGNN APPROACH

In this section, we first introduce the motivation of our model design and present an overview of TagGNN. Then, we describe the details of our model, including three parts: TagGNN-IT, TagGNN-QI, and their integration TagGNN. Finally, we show the training and inference strategies for tag prediction.

2.1 Motivation and Overview

2.1.1 Motivation. Nowadays, there is a trend to apply GNNs to enhancing text categorization tasks [9, 10, 30]. Inspired by these studies, we explore the use of GNNs for item tagging in IR. Different from textual categorization, our work aims to address the following unique challenges.

Firstly, item tagging problems usually have a large tag space (more than thousands). It is desired to capture the rich semantic relationships among tags. Taking Figure 1 as an example, Pokemon has two strongly correlated tags, i.e., AR (Augmented Reality) and VR (Virtual Reality). Such tag correlations are indicative of the strong co-existence or non-existence for related tags. Existing GNN methods mostly model text categorization as a node classification problem, since the number of categories is usually small (~tens). This, however, ignores the dependency of category labels.

Secondly, query information is readily available in IR tasks. While items lack concise textual descriptions, it is desired to join external information from query logs. For example, when a user search "chat" and download the app "Facebook", it potentially implies that the app is functionally related to "chat". Thus, tags like "chat" and "social" may be good candidates. The frequency of query-item interactions reveal the strength of such semantic correlations. How to effectively utilize the large amount of query information is an essential problem to build an accurate tagging system.

Thirdly, while existing item nodes mostly have edge connections in the graph, there are many new items everyday in the platform. These items have no links to either tag nodes or query nodes. This imposes a unique challenge for GNNs to deal with both full tag prediction and partial tag completion cases.

2.1.2 Overview. To address the above three challenges, we present TagGNN, a GNN-based item tagging approach. Figure 2 provides an overview of TagGNN. Suppose that we have got the related queries of the items from the IR system, and we also know the items' corresponding tags. Then we build an undirected tripartite graph to link query, item and tag together. The graph has three types of nodes, i.e., query, item and tag. Note that the item node can be unilateral or complete isolated if we do not know any related queries or existing tags (or both) of the item. Then, we employ TagGNN tailored for item tagging to propagate all of the information in the graph to get better item and tag representation. Finally, we compute the similarity between the item and all tags and choose K tags with the highest similarities as the our top$K$ prediction. In the following, we introduce TagGNN-IT, TagGNN-QI and TagGNN detail by detail.

2.2 TagGNN-IT

To fully exploit the interactions between items and tags, as well as correlations among tags, we treat the item tagging problem from the view of graph, modeling the multi-label classification as the link prediction problem in the graph. Specifically, we first build an undirected bipartite graph which has two types of nodes, i.e., item nodes and tag nodes. There will be an edge between the item node and the tag node if the item has the tag. Then TagGNN-IT learns new and powerful node representations in the graph for item tagging.

2.2.1 Node Representation. We choose the item titles and the tag names as the initial features of item nodes and tag nodes respectively. Without loss of generality, suppose that the node contains a string of words $(w_1, w_2, ..., w_n)$ extracted from content, title, name, description or others, we can use any models that can deal with the sequence, such as RNN and CNN, to get the initial representation h of the node. In particular, since the tag sets are fixed, we add an extra id embedding, which is a one-hot vector, for each tag node. For simplicity, here we just average all of word embeddings as the initial node representation:

$$h = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} w_i, & \text{if node is the item node} \\ \frac{1}{n}\sum_{i=1}^{n} w_i + id, & \text{if node is the tag node} \end{cases} \tag{1}$$

where id is the one-hot id embedding for the tag node.

Note that traditional multi-label text classification approaches only use the id (one-hot) embeddings of tags, which does not explicitly consider the correlations between tags. We set the tags as nodes in the graph and fuse the semantic information of tags into the initial representations, not only better modeling the correlations between tags, but also improving the generalization ability of the model.

2.2.2 TagGNN-IT Propagation. In the item-tag bipartite graph, the item node updates its representation by aggregating its neighbour tag nodes. Inspired by GAT [24], we let the tag node first compute the similarity between every neighbour tag node with their semantic (node) representations in a common embedding space. Formally, for the item node v and its neighbour tag node w, the similarity computation equation is:

$$\alpha_{vw} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_w\right]\right)\right)}{\sum_{k\in\mathcal{N}_v}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_k\right]\right)\right)}, \tag{2}$$

where $W \in \mathbb{R}^D$ is a transformation matrix to transform both the item node and tag node into a common embedding space, $\mathbf{a}$ is a global context vector to determine the similarity between the two nodes.

Then, based on their semantic similarity, the item node aggregates messages from all of neighbour tag nodes:

$$h_m = \sigma\Big(\sum_{w\in\mathcal{N}_v} \alpha_{vw} W h_w\Big), \tag{3}$$

where h_m is the incoming aggregated message from neighbour tag nodes, σ is the activation function (e.g., ReLU).

With the help of this attention mechanism, the item node can put more reasonable weights to its tag neighbours so that it can distinguish which tags are important, while which tags may be not informative and should be ignored. In this way, the item can more benefit from the representative tags and less affected by noisy tags.

Finally, we fuse the message and the original item representation into a new item embedding space to get the new item representation. One such propagation is named one layer, and we stack N such layers to capture higher-order neighbors' information. However, GNN often faces the over-smoothing problem as the number of layers gets deeper. To mitigate this and obtain more comprehensive representations, we adopt a gated skip-connection mechanism. The

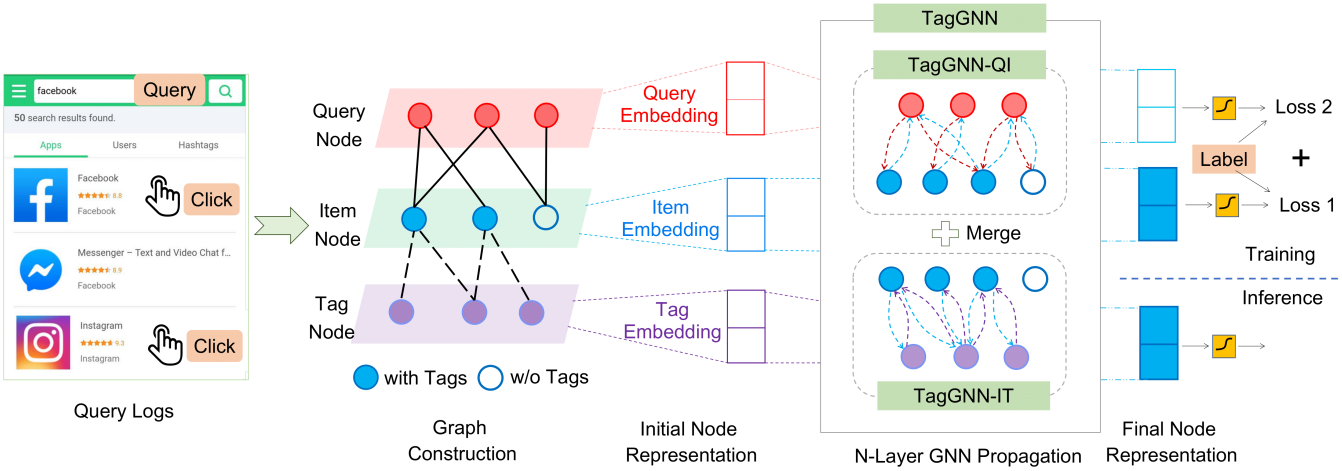


Figure 2: Overview of TagGNN

update equations are:

$$\hat{h}_v = \sigma(W_{item}(h_v + h_m)), \tag{4}$$

$$z = \text{sigmoid}(U_1\hat{h}_v + U_2 h_v + b), \tag{5}$$

$$h_v^{new} = z \odot \hat{h}_v + (1 - z) \odot h_v \tag{6}$$

where W_{item} is the transformation matrix to the new **item** embedding space, U_1, U_2, b are trainable parameters. z controls the proportion of the original representation and the new representation to get the final new representation.

The tag nodes follow the similar propagation operations except that they have their own transformation matrix W_{tag} when fusing the aggregated message from item nodes (equation 4).

2.2.3 Loss. We deem the item tagging problem as a link prediction problem in the graph, i.e., to predict which tag nodes should be linked to the target item node, which can leverage enriched tag representations to improve the performance.Specifically, we compute dot-product similarity between the item representation h_i and tag representation h_t, and compute its binary cross-entropy loss $\mathcal{L}_{LP}$ with the ground truth (0 or 1 represents link or not link):

$$\mathcal{L}_{LP}(h_i, h_t, y) = BCE(y,\ h_i \cdot h_t) \tag{7}$$

where BCE is binary cross-entropy loss, y is the ground truth for the edge between the item node and the tag node.

2.3 TagGNN-QI

To effectively leverage the query information which has not been exploited in previous literature, we design another model named TagGNN-QI also from the graph view. Specifically, we build a query-item bipartite graph from the interactions of query logs and items. There will be an (weighted) edge between the query node and the item node if they are interacted. The query-item edge can represent different meanings depending on different real scenarios. For example, in the App Store scenario (app tagging), the query-item edge may represent the click or download behavior for the app under the query, and the edge weight can be the click times or downloads.

2.3.1 Edge Representation. In TagGNN-QI, both node features and edge features are used. Similar to TagGNN-IT, we use the query contents and item titles as the initial features of the query and item nodes, and also average all the word embeddings as the initial node representations. Here we focus on edge features. As the edge may contain useful information, we also encode the initial edge representation for TagGNN-QI. Specifically, if the edge originally has a feature vector, we just keep it. If the edge weight is a scalar, we can use the weight to enhance the message passing through this edge by simply multiply the message with the weight scalar. Besides, if the weight range is very large, we can use some feature scaling strategies like min-max normalization or standardization to rescale. We can also perform feature discretization, e.g., binning [33] , to get the initial edge representation. If the edge not has weight, we just set all edge weights to 1.

2.3.2 TagGNN-QI Propagation. We change the similarity computation so as to utilize the information contained in the edge. Formally, if the edge representation e_{vw} is a vector, the similarity is:

$$\alpha_{vw} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_w \| e_{vw}\right]\right)\right)}{\sum_{k\in\mathcal{N}_v}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_k \| e_{vk}\right]\right)\right)}, \tag{8}$$

While if the edge representation e_{vw} is a scalar, the similarity is:

$$\alpha_{vw} = e_{vw} \times \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_w\right]\right)\right)}{\sum_{k\in\mathcal{N}_v}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[Wh_v \| Wh_k\right]\right)\right)}, \tag{9}$$

The notations and other operations are consistent with TagGNN-IT described in 2.2.2.

2.3.3 Loss. Since TagGNN-QI does not have tag nodes, we model it as a regular node classification form. We use a multi-layer perceptron (MLP) to transform the item representation h_i to a N dimensional vector d (where N is the number of tags) and we compute

the mean binary cross-entropy loss $\mathcal{L}_{NC}$ with the ground truth:

$$d = W_{nc} h_i + q, \tag{10}$$

$$\mathcal{L}_{NC}(h_i, y) = \frac{1}{N} \sum_{t=1}^{N} BCE(d_t, y_t) \tag{11}$$

where W_{nc} and q are trainable parameters of MLP, y_t is the ground truth between the item and the t-th tag, d_t is the t-th dimension of d.

2.4 TagGNN

To solve all the three limitations simultaneously, we integrate TagGNN-IT and TagGNN-QI to a unified model named TagGNN, which inherits both their advantages. Specifically, we merge the former two bipartite graphs to one tripartite graph which has three types of nodes, i.e., query nodes, item nodes and tag nodes. The edges are the same as in the original graphs, and the initial representations of nodes and edges are also the same as before. We perform message passing in this unified tripartite graph following the same propagation strategies described in TagGNN-IT and TagGNN-QI. Therefore, the item node will simultaneously get the messages from both query nodes and tag nodes to update its representation. TagGNN also deal with the item tagging as a link prediction problem, So, its loss form is same as TagGNN-IT described in 2.2.3 (i.e., $\mathcal{L}_{LP}$).

2.5 Training and Inference

In this part, we introduce how the three models are trained and used for item tagging.

2.5.1 Training. In the graph, there may exist isolated test item nodes which we are unaware of any query or tag information about them, and we stipulate that their representations will not be updated by the GNN propagation. As the majority of training item nodes are not isolated, there will be a training-testing exposure bias, seriously reducing the the prediction precision of isolated test item nodes.

To empower the model with the ability to handle this "cold start" problem, we add a dual loss in addition to the primary loss during training. Specifically, the primary loss $\mathcal{L}_1$ is computed with the new learned item representation and the new learned tag representation. While the dual loss $\mathcal{L}_2$ is computed with the initial item representation and the new learned tag representation. Finally we optimize the model by reducing these two losses together with stochastic gradient descent algorithms. Formally, for TagGNN-IT and TagGNN:

$$\mathcal{L}_1 = \mathcal{L}_{LP}(h^{new}_{item}, h^{new}_{tag}, y), \tag{12}$$

$$\mathcal{L}_2 = \mathcal{L}_{LP}(h_{item}, h^{new}_{tag}, y), \tag{13}$$

for TagGNN-QI:

$$\mathcal{L}_1 = \mathcal{L}_{NC}(h^{new}_{item}, y), \tag{14}$$

$$\mathcal{L}_2 = \mathcal{L}_{NC}(h_{item}, y), \tag{15}$$

The final optimization objective is:

$$\mathcal{L} = \mathcal{L}_1 + \gamma \mathcal{L}_2 \tag{16}$$

where γ is the hyper-parameter to adjust the proportion of $\mathcal{L}_1$ and $\mathcal{L}_2$.

2.5.2 Inference. When inference, since the model has been optimized to be able to deal with isolated item nodes, we do not distinguish whether the item node is isolated or not. For TagGNN-IT and TagGNN, we compute similarities between the item representation (after propagation) and all of tags representations (after propagation), and choose K tags with the highest similarities as the result. For TagGNN-QI, we transform the item representation (after propagation) to a N dimensional vector and choose the K tags corresponding to the K largest dimensions of the vector as the result.

3 EXPERIMENT

In this section, we conduct experiments on two datasets about advertisement tagging and application tagging, aiming to answer the following questions:

- **Q1**: How does TagGNN perform compared with the state-of-the-art item tagging related approaches on our tasks?
- **Q2**: Does the dual loss $\mathcal{L}_2$ really improve the performance of TagGNN? What is the impact of tag name embeddings? Does heterogeneity of TagGNN take effect?
- **Q3**: How do different designs (e.g., the number of TagGNN layers, the types of GNN) influence the performance of TagGNN?

3.1 Dataset

We perform experiments on the following two real-world datasets.

KDDCup-2012: This public dataset is originally provided by KDD Cup 2012 track2 competition for CTR prediction. Its training instances derived from session logs of the Tencent proprietary search engine, soso.com. From the dataset, we can get the **advertisements** (items), **queries** that trigger the advertisements, and the **keywords** (tags) of the advertisements. We preprocess this dataset for advertisement tagging. Specifically, we process the dataset to satisfy the following three limitations:

- In Query-Ad graph, every advertisement node link at least 20 query nodes, and every query node link at least 20 advertisement nodes.
- In Ad-Keyword graph, every advertisement node link at least 5 keyword nodes, and every keyword nodes link at least 15 advertisement nodes.
- Each word in the vocabulary should appears at least 5 times.

Huawei-Dataset: It is a industrial dataset derived from a business company's App Store. To make it non-representative of the online app search traffic, we randomly sample a subset from original data, but cover both apps without tags and apps with partial tags, and use one week query logs. A query is related to the app when the user clicks or downloads the app searching with this query. The query-app edges have weights, which represents the downloads of the app under the query. We use this dataset for app tagging.

The statistics of the two datasets are presented in Table1.

3.2 Experimental Setup

We consider two types of tagging tasks. The first is **full tag prediction**, which means that we do not know any existing tags of

Table 1: Dataset Statistics. "Avg. Queries" and "Avg. Tags" represent the average number of queries and tags associated to an item, respectively.

Dataset	#Query	#Item	#Tag	#Vocab	Avg. Queries	Avg. Tags
KDDCup-2012	92380	18861	9140	6620	89.4	13.8
Huawei-Dataset	47305	34166	2636	18601	5.8	3.6

the item and we should predict all of its tags. The second is **tag completion**, which means that we have known some tags of the item and we want to predict its remaining tags. For the second task, in our experiment, we randomly choose two tags of each item to predict and set its remaining tags as known tags.

For KDDCup-2012 dataset, we randomly choose 14861, 2000, 2000 advertisements for training, validation and test respectively. In the validation and test parts, 1000 advertisements are used for full tag prediction and another 1000 advertisements are used for tag completion. For Huawei-Dataset, we randomly choose 28166, 3000, 3000 apps for training, validation and test respectively. Similarly, in the validation and test set, 1500 apps are for full tag prediction and another 1500 apps are for tag completion.

The embedding size of the node and the word are both set to 200. The number of TagGNN layer is set to 2. TagGNN is trained with Adam optimizer, with 0.003 learning rate. Besides, we use standardization to normalize edge weights (we leave feature discretization in the future work). We apply 0.5 feature dropout rate to alleviate overfitting. We stop training the model when the validation error plateaus. We use **Precision@K**, which is a common metric for multi-label classification task, as our evaluation metric.

3.3 Baselines

In order to verify the validity of TagGNN, we compare it with the following baselines[1]:

- **FastText-I**: FastText [6] is a simple and efficient text classification approach which averages the word/n-grams embeddings as the document embedding, then feeds the document embedding into a linear classifier. We use it to do multi-label text classification with item titles.
- **FastText-QI**: The only difference with FastText-I is that we concatenate the item title with its top-10 queries' contents as the new initial features.
- **Transformer-I**: This baseline follows the multi-label classification model [26] that applies the most commonly used Transformers as the text encoder. We use item title as input.
- **Transformer-QI**: The only difference with Transformer-I is that we concatenate the item titles with its top-10 queries' contents as the new initial features.
- **XmlCNN-QI**: XmlCNN [15] is a multi-label classification model that follows TextCNN [12] to use CNNs as the text encoder. We use the concatenation of item title and query content as input. We set kernel sizes to {2,3,4} and use 100 kernels for each kernel size.
- **TextRNN-QI**: TextRNN [16] is a frequently-used text classification method which employ RNN with multi-task learning to encode the text. We use it to do multi-text classification with query contents and item titles. We use one-layer bidirectional RNN and the hidden embedding size is set to 200.
- **SimRank-QI**: SimRank [11] is a popular graph-based approach that exploits the node-to-node relationships based on the topology of the graph. We propagate tags in the query-item bipartite graph based on SimRank to predict new tags for items.
- **ML-GCN-I**: ML-GCN [4] is a recently published work that learns the label correlations via GCNs for image-based multi-label classification. We extend it to text-based classification and use FastText (performed best in experiments) as the textual encoder. We set τ to 0.1 and 0.3 for KDDCup-2012 and Huawei-Dataset respectively to build the needed label graph. Other settings are consistent with the original paper.
- **ML-GCN-QI**: The only difference with ML-GCN-I is that we change the main model to **TagGNN-QI**.

[1]The embedding size is uniformly set to 200 for all baselines if not specified. Part of baselines are experimented with https://github.com/Tencent/NeuralNLP-NeuralClassifier.

3.4 Performance Comparison (Q1)

The comparative results are summarized in Table 2. In the following, we discuss the results of two tasks, i.e., full tag prediction and tag completion respectively.

3.4.1 Results of Full Tag Prediction. We have the following observations about the results of full tag prediction task:

- Our final TagGNN substantially outperforms all the other baselines on both two datasets, verifying the effectiveness of our model to solve the full tag prediction task. In particular, TagGNN improves the strongest baseline TagGNN-QI (also ours) by 6.8% and 5.3% in P@1 and P@5 on KDDCup-2012 dataset. We attribute such notable improvements to the novel and powerful design of TagGNN that can benefit from both explicit and implicit interactions and representation fusions among queries, items and tags.
- Query information is very useful and can be easily utilized to solve the full tag prediction task. It is obvious that the precision gains a huge improvement (7% to 38.5% for KDDCup-2012, and 13.7% to 30.7% for Huawei-Dataset) for all baselines after fusing the query information, which strongly proves the importance of queries. Note that the gains of ML-GNN-QI compared with ML-GNN-I are also mostly originated from TagGNN-QI since their major difference lies in the main models of ML-GNN. Thus, comparatively speaking, TagGNNs get the biggest percentages of boost from the query information, demonstrating that TagGNNs can utilize queries better than other baselines.
- Graph-based SimRank-QI and ML-GCN-I are inferior to text-based FastText-QI and Transformer-I respectively. This phenomenon shows that not all graph-based or graph&text-based methods can beat the traditional text-based methods. How to use all the information in the form of graph is the real key for graph based methods, not the graph form itself. This also proves that TagGNN can take advantage of graph information more effectively.
- Unexpectedly, ML-GCN-QI performs worse than TagGNN-QI. Since the main model of ML-GCN-QI we used is just TagGNN-QI, it demonstrates that the label (tag) embedding

Table 2: Performance comparison of different models. "Without Tags" indicates that items have no tags before prediction, and "Partial Tags" means that items have incomplete tags and need tag completion. TagGNN-IT, TagGNN-QI and TagGNN show our approaches.

	Model	Features	KDDCup-2012						Huawei-Dataset					
			Without Tags			Partial Tags			Without Tags			Partial Tags		
			P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
Text based	FastText-I	Item Text	0.405	0.352	0.331	0.158	0.134	0.105	0.529	0.386	0.286	0.392	0.265	0.197
	FastText-QI	Item & Query Text	0.581	0.510	0.470	0.286	0.190	0.143	0.688	0.492	0.354	0.515	0.340	0.246
	Transformer-I	Item Text	0.373	0.332	0.311	0.175	0.124	0.098	0.471	0.338	0.249	0.358	0.244	0.185
	Transformer-QI	Item & Query Text	0.443	0.393	0.363	0.161	0.124	0.097	0.608	0.432	0.313	0.477	0.314	0.227
	XmlCNN-QI	Item & Query Text	0.371	0.327	0.302	0.112	0.083	0.064	0.515	0.356	0.259	0.341	0.225	0.163
	TextRNN-QI	Item & Query Text	0.484	0.424	0.387	0.169	0.117	0.089	0.615	0.428	0.308	0.379	0.255	0.186
Graph based	SimRank-QI	Query-Item Graph	0.559	0.510	0.479	0.171	0.144	0.125	0.577	0.421	0.299	0.499	0.342	0.243
Graph & Text based	ML-GCN-I	Tag-Tag Graph	0.365	0.311	0.296	0.191	0.148	0.113	0.414	0.342	0.251	0.414	0.276	0.200
	ML-GCN-QI	Query-Item & Tag-Tag Graph	0.742	0.672	0.625	0.385	0.273	0.193	0.721	0.519	0.371	0.612	0.388	0.272
	TagGNN-IT	Item-Tag Graph	0.438	0.326	0.280	0.342	0.250	0.187	0.539	0.362	0.264	0.444	0.286	0.209
	TagGNN-QI	Query-Item Graph	0.755	0.688	0.643	0.403	0.295	0.214	0.730	0.520	0.379	0.618	0.395	0.276
	TagGNN	Query-Item-Tag Graph	**0.823**	**0.741**	**0.683**	**0.449**	**0.330**	**0.236**	**0.743**	**0.534**	**0.381**	**0.644**	**0.416**	**0.288**

strategy proposed in ML-GCN is not effective on item tagging task, and further proves the effectiveness of the way that TagGNN leveraging the tags.

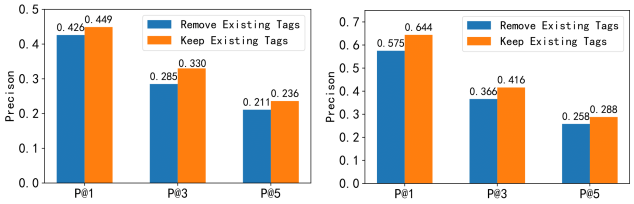
3.4.2 Results of Tag Completion. We have the following observations about the results of tag completion task:

- TagGNN still achieves the best performance across the two datasets on tag completion task, demonstrating the comprehensive superiority of TagGNN than other baselines. Specifically, it can beat the strongest baseline TagGNN-QI (also ours) by 4.6% and 2.6% in P@1 on KDDCup-2012 dataset and Huawei-Dataset, and hugely surpasses all the text-based approaches, which is a relative good performance.
- When queries are not available , TagGNN-IT outperforms all the other baselines, i.e., FastText-I, Transformer-I and ML-GCN-I on the two datasets. More notably, on KDDCup-2012 dataset, even if the other baselines using queries, TagGNN-IT can still outperforms them in most cases. It may be because that the number of tags in KDDCup-2012 is larger than Huawei-Dataset (as shown in Tabel 1), which boosts the TagGNN to better release its potency. Such an excellent performance of TagGNN-IT also verifies that the design of our TagGNN framework has strong ability to leverage existing tags so as to improve the performance of tag completion task.

In addition, some readers may wonder why the results of full tag prediction seem to be better than the results of tag completion as shown in Table 2? Here is an illustration:

These two tasks have different numbers of ground truth tags. Note that there are only 2 ground truth tags for tag completion task. So, compared with the larger ground truth set of full tag prediction task, it is more difficult to hit the ground truth tags in tag completion task, leading to an illusion that the tag completion's precision is lower than the full tag prediction's.

To further demonstrate that TagGNN can really leverage the existing tags to improve the performance of tag completion, we remove all the existing tags of items in the test set, and retrain the model to test its performance. Results are shown in Figure 3. It is obvious that the performance gets worse after removing the existing tags, which verifies our illustration.



(a) KDDCup-2012 (b) Huawei-Dataset

Figure 3: Results of removing/keeping existing tags for the tag completion task.

3.5 Ablation Study (Q2)

In this section, we study how three particular components of TagGNN, i.e., $\mathcal{L}_2$ (see 2.5), tag name embeddings (see 2.2.1) and heterogeneity affect the performance and answer **Q2**.

3.5.1 Dual Loss $\mathcal{L}_2$. As described in 2.5, we add the dual loss $\mathcal{L}_2$ to deal with the cold start problem of isolated items. Here, we perform a ablation study to verify the validity of this strategy by removing $\mathcal{L}_2$ when training. Table 3 shows the experimental results of TagGNNs training with $\mathcal{L}_2$ and without $\mathcal{L}_2$, and we have the following findings:

- Without $\mathcal{L}_2$, performances of TagGNN-IT for full tag prediction on both two datasets drop sharply, indicating that the model nearly loses the ability to handle the cold start problem. Such performance degradation is due to the fact that When the graph has no query nodes, all training item nodes still have neighbour tag nodes. However, the test item nodes for full tag prediction will have no neighbours to aggregate, leading to a huge gap between training and inference.
- Although $\mathcal{L}_2$ may slightly hurt the precision of TagGNN-IT on tag completion task, it is trivial compared with the huge improvement on full tag prediction task. On the whole, our

Table 3: Performance comparison of ablation study. "TagGNN w/o $\mathcal{L}_2$ & TNE" represents TagGNN without both $\mathcal{L}_2$ and tag name embeddings, "TagGNN w/o $\mathcal{L}_2$" represents TagGNN only without $\mathcal{L}_2$, "TagGNN-homogeneous" represents TagGNN using homogeneous update function. Similar notations for others.

Model	KDDCup-2012						Huawei-Dataset					
	Without Tags			Partial Tags			Without Tags			Partial Tags		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
TagGNN-IT	0.438	**0.326**	**0.280**	0.342	0.250	0.187	**0.539**	**0.362**	**0.264**	0.444	0.286	0.209
TagGNN-IT w/o $\mathcal{L}_2$	0.016	0.011	0.010	**0.360**	0.274	**0.200**	0.291	0.199	0.149	**0.447**	**0.294**	0.207
TagGNN-IT w/o $\mathcal{L}_2$ & TNE	0.012	0.010	0.009	0.332	**0.258**	0.192	0.289	0.200	0.148	**0.447**	0.293	**0.210**
TagGNN-IT-homogeneous	**0.439**	0.320	0.274	0.333	0.241	0.183	0.529	0.352	0.261	0.428	0.277	0.206
TagGNN-QI	**0.755**	**0.688**	**0.643**	**0.403**	**0.295**	**0.214**	**0.723**	0.520	**0.379**	**0.618**	**0.395**	**0.276**
TagGNN-QI w/o $\mathcal{L}_2$	0.746	0.679	0.639	0.402	0.288	0.211	**0.723**	0.517	0.378	0.611	0.393	**0.276**
TagGNN-QI w/o $\mathcal{L}_2$ & TNE	0.747	0.675	0.635	0.401	0.286	0.209	0.715	0.516	0.371	0.606	0.392	0.272
TagGNN-QI-homogeneous	0.745	0.675	0.632	0.391	0.280	0.207	0.722	**0.524**	0.369	0.608	0.381	0.268
TagGNN	**0.823**	**0.741**	**0.683**	**0.449**	**0.330**	**0.236**	**0.743**	**0.534**	**0.381**	**0.644**	**0.416**	**0.288**
TagGNN w/o $\mathcal{L}_2$	0.791	0.719	0.664	0.442	0.316	0.235	0.721	0.521	0.377	0.637	0.409	0.282
TagGNN w/o $\mathcal{L}_2$ & TNE	0.789	0.715	0.661	0.426	0.306	0.227	0.711	0.519	0.361	0.624	0.391	0.275
TagGNN-homogeneous	0.807	0.724	0.673	0.417	0.315	0.227	0.732	0.527	0.375	0.629	0.404	0.282

Table 4: Performance comparison of TagGNN with different number of propagation layers.

Model	KDDCup-2012						Huawei-Dataset					
	Without Tags			Partial Tags			Without Tags			Partial Tags		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
TagGNN-1	0.786	0.697	0.649	0.412	0.302	0.218	0.676	0.474	0.338	0.553	0.351	0.245
TagGNN-2	**0.823**	**0.741**	**0.683**	**0.449**	**0.330**	0.236	**0.743**	**0.534**	**0.381**	**0.644**	**0.416**	**0.288**
TagGNN-3	0.815	0.735	0.674	0.432	0.321	**0.238**	0.732	0.515	0.364	0.641	0.411	0.284
TagGNN-4	0.811	0.728	0.670	0.428	0.315	0.234	0.728	0.507	0.361	0.633	0.409	0.283

Table 5: Performance comparison of different types of GNN.

Model	KDDCup-2012						Huawei-Dataset					
	Without Tags			Partial Tags			Without Tags			Partial Tags		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
GCN	0.771	0.683	0.635	0.441	0.313	0.227	0.717	0.507	0.359	0.594	0.386	0.269
GraphSAGE	0.793	0.712	0.663	0.441	0.305	0.221	0.675	0.465	0.331	0.592	0.372	0.258
GAT	0.806	0.725	0.671	0.424	0.306	0.223	0.722	0.515	0.364	0.623	0.394	0.273
TagGNN	**0.823**	**0.741**	**0.683**	**0.449**	**0.330**	**0.236**	**0.743**	**0.534**	**0.381**	**0.644**	**0.416**	**0.288**

proposed dual loss $\mathcal{L}_2$ is really an effective way to handle the isolated nodes.

- Queries are quite informative and powerful to greatly alleviate the gap mentioned above. From the results of "TagGNN-QI w/o $\mathcal{L}_2$" and "TagGNN w/o $\mathcal{L}_2$", we find that when the query information is available, $\mathcal{L}_2$ may be cannot bring very notable promotion as before. But it is still a valid auxiliary to improve accuracy.
- Moreover, jointly considered Table 2 and Table 3, we find that even without query information and $\mathcal{L}_2$, "TagGNN-IT w/o $\mathcal{L}_2$" is still much better than traditional multi-label text classification methods on tag completion task. This demonstrates that the mode of TagGNN-IT can more effectively utilize the existing tag information to help solve tag completion task.
- We note that the heterogeneous GNN based HGAT model [9] cannot be directly applied to item tagging. But considering that it models text categorization as node classification, we can take "TagGNN-QI w/o $\mathcal{L}_2$" as the approximate implementation of HGAT on item tagging. The results show that TagGNN is much better than HGAT on both datasets.

3.5.2 Tag Name Emeddings. We introduced in 2.2.1 that the initial representation of the tag node is the combination of its tag name embedding and tag id embedding. Note that the tag id embedding is always available since it is a one-hot embedding which is only related to the total number of tags. However, the tag name may be not visible during training the model in some situations (e.g., the company outsources the project of item tagging to others but it do not want to disclose the exact names of tags). Thus, here we remove the tag name embeddings from the initial node representation and see how it influence the performance of TagGNN. The experimental results are reported in Table 3.

On the whole, the results show that the tag name embeddings (TNE) just bring slight improvement. But we believe the potential of TNE is much more than that. we are also considering how to

better use the semantic information of tag names, such as exploring more fine-grained word-level interactions among items and tags. We leave it in our future work.

3.5.3 homogeneity and heterogeneity of TagGNN. Considering that the query-item-tag tripartite graph is heterogeneous, we also design TagGNN to be heterogeneous, as embodied in its update function (equation 4). To demonstrate the effectiveness of this heterogeneous design, we change the update function to be homogeneous, i.e., not distinguish the node types, and test the performance of TagGNN-QI, TagGNN-IT and TagGNN. We report the experimental results in Table 3.

From the results, we can see that heterogeneous TagGNNs are nearly consistently better than homogeneous ones, demonstrating that setting different transformation matrices for different types of nodes is reasonable and valid, which can bring steady improvement.

3.6 Design Choices of TagGNN (Q3)

In this part, we research how different designs influence the performance from two perspectives.

3.6.1 ***Effect of Layer Numbers.*** To explore how the number of propagation layers affects the performance, we vary the number of model layers. Specially, we conduct experiments with the layer numbers in range of {1, 2, 3, 4}. Table 4 summarizes the experimental results, wherein TagGNN-X indicates the model with X layers. From the results, we have the following observations:

- TagGNN-1 is obviously worse than TagGNN-2,3,4, indicating that only one propagation layer is not enough to reach an excellent performance. It is reasonable since one-layer GNN propagation can only capture the first-order neighbors' information. Hence, semantic relationships between query and query, item and item, tag and tag are not explicitly used, resulting in unsatisfactory performance. So it is necessary to stack at least two propagation layers.
- Stacking too much (larger than 3) layers will not bring additional promotion. Compared with TagGNN-2, only TagGNN-3 got a little gain (0.2%) in P@5 (Paritial Tags of KDDCup-2012), verifying that two layers are enough for TagGNN. Too many layers may lead to redundancy that hurts performance.

3.6.2 ***Effect of Types of GNN.*** To verify the superiority of the propagation design of TagGNN, we replace the TagGNN with some other popular GNN models, e.g., GCN, GraphSAGE and GAT. For GraphSAGE, we choose its "mean" strategy. All corresponding settings are consistent with TagGNN. We show the experimental results in Table 5.

The results shows that our TagGNN is clearly superior to all these representative GNNs. Specifically, GCN and GraphSAGE beat each other on two datasets but are worse than GAT. As for GAT, it is modestly inferior to TagGNN. It may be because that TagGNN can leverage additional edge information and has better representation fusion between two layers, which makes TagGNN more effective for item tagging.

3.7 Expert Evaluation

Before deploying the tagging model for production use, we need to perform a manual A/B testing by our operation team. Specifically, we randomly sample 540 apps from the test set of Huawei-Dataset, half for full tag prediction and half for tag completion. The sampling is performed uniformly to keep the proportion of each app category (e.g., game, study) consistent with the whole app corpus. In the full tag prediction setting, we predict top-5 tags for each app, while in the tag completion setting, we predict top-k tags to assure that each item has at least five tags. For example, if an item has 3 existing tags (3.6 on average), we set k=2. We generate two groups of app-tag samples predicted using both TagGNN and our production baseline model. This leads to a total of 4050 app-tag pairs. We randomly split the test samples and distribute them to four domain experts from our operation team. They assess the test samples one by one to check whether a tag is appropriate for an app. Finally, the expert evaluation results show that TagGNN achieves 81.1% accuracy for full tag prediction, and 88.2% accuracy for tag completion. Meanwhile, TagGNN achieves a 22.8% relative improvement over the production baseline. The improvement is significant for production deployment.

4 RELATED WORK

4.1 Multi-Label Classification

Multi-label classification [32] is a widely-studied research topic, spanning multiple tasks such as text tagging [15, 26] and image annotation [4]. Recent research efforts have been devoted to optimizing the content representation learning or exploring label dependencies for improvement. More specifically, Liu et al. [15] and Chang et al. [26] study the application of CNNs and transformers to enhance text-based multi-label classification, respectively. Chen et al. [4] investigate the use of GNNs to capture correlations among labels. All these studies assume rich contents. In contrast, we have to leverage external information (e.g., query logs) to enrich items. We also empirically compare TagGNN with them in Table 2.

4.2 Graph Neural Networks

Graph neural networks (GNNs) [29] has become a trending research topic. The research of GNNs successfully extends traditional convolutional neural networks to graph-structured data, leading to abundant applications such as text categorization [30], recommendation [31], and link prediction [7]. Our work is inspired by these successful studies, and has been extended for item tagging. We empirically compare TagGNN with three representative GNN models, i.e., GCN [13], GraphSAGE [7], and GAT [24].

4.3 GNN-based Text Categorization

As a promising technique, GNNs have been recently adopted to boost text categorization tasks. In particular, Yao et al. [30] propose the first use of graph convolution networks for text classification. But this work models each document as a graph node and cannot handle new documents that are not present in the graph during training. Later work [9, 10] makes some extensions to tackle this issue. Especially, Hu et al. [9] construct a topic-document-entity graph and model it using heterogeneous GNNs. This work is mostly closest to ours. However, the differences lie in that: 1) We model item tagging as a link prediction problem, instead of the node classification formulation in [9], which enables both full tag prediction and tag completion. 2) Our query and tag nodes, which naturally

exist in IR tasks, provide multi-source information to enrich item representation, but topic and entity nodes are all intermediate information extracted from documents using preprocessing tools.

5 CONCLUSION

In this paper, we present TagGNN, a tripartite graph neural network model for item tagging. Our model builds on the heterogeneous GNN techniques, but differs from other previous studies in three unique aspects: 1) Instead of node classification, TagGNN formulates item tagging as a novel link prediction problem. 2) TagGNN leverages query logs to enrich item representation and forms a query-item-tag tripartite graph that is unique for IR. 3) TagGNN is capable of making both full tag prediction and partial tag completion in a unified way. Experimental results on two large datasets validate the superiority of our TagGNN approach over existing methods. In addition, we perform an expert evaluation from our operation team and obtain quite positive results for production use.

6 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of Guangdong Province (2018A030313422), the National Natural Science Foundation of China (61972219, 61773229), the National Key Research and Development Program of China (2018YFB1800600, 2018YFB1800204), the R&D Program of Shenzhen (JCYJ20190813174403598, JCYJ20190813165003837), and Overseas Cooperation Research Fund of Graduate School at Shenzhen, Tsinghua University (HW2018002), and the research fund of PCL Future Regional Network Facilities for Large-scale Experiments and Applications (PCL2018KP001).

REFERENCES

[1] 2018. Metadata and the Tagging Process at The New York Times. https://iptc.org/news/metadata-and-the-tagging-process-at-the-new-york-times

[2] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-Scale Multi-Label Text Classification on EU Legislation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*. 6314–6322.

[3] Ning Chen, Steven C. H. Hoi, Shaohua Li, and Xiaokui Xiao. 2016. Mobile App Tagging. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*. 63–72.

[4] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-Label Image Recognition With Graph Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5177–5186.

[5] Corina Florescu and Cornelia Caragea. 2017. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

[6] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, (EACL)*.

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Annual Conference on Neural Information Processing Systems (NIPS)*. 1024–1034.

[8] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. 1262–1273.

[9] Linmei Hu, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, (EMNLP-IJCNLP)*. 4820–4829.

[10] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text Level Graph Neural Network for Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[11] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 538–543.

[12] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

[14] Jianguo Li, Yong Tang, and Jiemin Chen. 2016. Leveraging tagging and rating for recommendation: RMF meets weighted diffusion on tripartite graphs. *CoRR* abs/1611.00812 (2016).

[15] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 115–124.

[16] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101* (2016).

[17] Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2014. Tagging Your Tweets: A Probabilistic Modeling of Hashtag Annotation in Twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. 999–1008.

[18] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*. 404–411.

[19] Eirini Papagiannopoulou and Grigorios Tsoumakas. 2020. A review of keyphrase extraction. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 10, 2 (2020).

[20] Janani Balaji Saeid Belkasim Rajshekhar Sunderraman Sanghoon Lee, Mohamed Masoud and Seung-Jin Moon. 2016. A Survey of Tag-based Information Retrieval. *International Journal of Multimedia Information Retrieval* 6 (2016).

[21] Bichen Shi, Gevorg Poghosyan, Georgiana Ifrim, and Neil Hurley. 2018. Hashtagger+: Efficient High-Coverage Social Tagging of Streaming News. *IEEE Trans. Knowl. Data Eng.* 30, 1 (2018), 43–58.

[22] Bo Sun, Yunzong Zhu, Yongkang Xiao, Rong Xiao, and Yungang Wei. 2019. Automatic Question Tagging with Deep Neural Networks. *IEEE Transactions on Learning Technologies* 12, 1 (2019), 29–43.

[23] Shijie Tang, Yuan Yao, Suwei Zhang, Feng Xu, Tianxiao Gu, Hanghang Tong, Xiaohui Yan, and Jian Lu. 2019. An Integral Tag Recommendation Model for Textual Content. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 5109–5116.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[25] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint Embedding of Words and Labels for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2321–2331.

[26] Kai Zhong Yiming Yang Inderjit Dhillon Wei-Cheng Chang, Hsiang-Fu Yu. 2019. X-BERT: eXtreme Multi-label Text Classification with using Bidirectional Encoder Representations from Transformers. In *arXiv preprint arXiv:1905.02331*.

[27] Jason Weston, Sumit Chopra, and Keith Adams. 2014. #TagSpace: Semantic Embeddings from Hashtags. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1822–1827.

[28] Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Improving Recommendation of Tail Tags for Questions in Community Question Answering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*. 3066–3072.

[29] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019).

[30] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 7370–7377.

[31] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*. 974–983.

[32] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Trans. Knowl. Data Eng.* 26, 8 (2014), 1819–1837.

[33] Alice Zheng and Amanda Casari. 2018. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.".

[34] Jingbo Zhou, Shan Gou, Renjun Hu, Dongxiang Zhang, Jin Xu, Airong Jiang, Ying Li, and Hui Xiong. 2019. A Collaborative Learning Framework to Tag Refinement for Points of Interest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1752–1761.

[35] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. 2017. Learning Spatial Regularization with Image-Level Supervisions for Multi-label Image Classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2027–2036. https://doi.org/10.1109/CVPR.2017.219