Multi-Relational Graph based Heterogeneous Multi-Task Learning in Community Question Answering

Zizheng Lin[1], Haowen Ke[1], Ngo-Yin Wong[1], Jiaxin Bai[1], Yangqiu Song[1], Huan Zhao[2], Junpeng Ye[3]
[1]Department of Computer Science and Engineering, HKUST, Hong Kong, China
[2]4Paradigm Inc., Beijing, China
[3]Tencent Technology (SZ) Co., Ltd., Shenzhen, China
{zlinai,hkeaa,nywongac,jbai,yqsong}@cse.ust.hk,zhaohuan@4paradigm.com,jayjpye@tencent.com

ABSTRACT

Various data mining tasks have been proposed to study Community Question Answering (CQA) platforms like Stack Overflow. The relatedness between some of these tasks provides useful learning signals to each other via Multi-Task Learning (MTL). However, due to the high heterogeneity of these tasks, few existing works manage to jointly solve them in a unified framework. To tackle this challenge, we develop a multi-relational graph based MTL model called Heterogeneous Multi-Task Graph Isomorphism Network (HMTGIN) which efficiently solves heterogeneous CQA tasks. In each training forward pass, HMTGIN embeds the input CQA forum graph by an extension of Graph Isomorphism Network and skip connections. The embeddings are then shared across all task-specific output layers to compute respective losses. Moreover, two cross-task constraints based on the domain knowledge about tasks' relationships are used to regularize the joint learning. In the evaluation, the embeddings are shared among different task-specific output layers to make corresponding predictions. To the best of our knowledge, HMTGIN is the first MTL model capable of tackling CQA tasks from the aspect of multi-relational graphs. To evaluate HMTGIN's effectiveness, we build a novel large-scale multi-relational graph CQA dataset with over two million nodes from Stack Overflow. Extensive experiments show that: (1) HMTGIN is superior to all baselines on five tasks; (2) The proposed MTL strategy and cross-task constraints have substantial advantages.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Community Question Answering, Heterogeneous Multi-Task Learning, Multi-Relational Graph, Cross-task Constraint

ACM Reference Format:
Zizheng Lin[1], Haowen Ke[1], Ngo-Yin Wong[1], Jiaxin Bai[1], Yangqiu Song[1], Huan Zhao[2], Junpeng Ye[3]. 2021. Multi-Relational Graph based Heterogeneous Multi-Task Learning in Community Question Answering. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482279

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482279

1 INTRODUCTION

Community Question Answering (CQA) forums like Stack Overflow[1] help millions of users to seek solutions or share their knowledge. Various CQA tasks like duplicate question detection and answer recommendation have been extensively studied [31]. Due to rich interconnections between components of a CQA platform, some CQA tasks are related to each other. For example, tag recommendation can be solved using a link prediction model and the semantic representation of keywords reflected by this task can also help evaluate a similar link prediction model in duplicate question detection. Moreover, duplicate question detection would also help personalize answer recommendation as they both evaluate text similarities, although the latter should be formulated as a ranking problem. Thus, it is natural to use Multi-Task Learning (MTL) [5] to jointly solve different CQA tasks to obtain better overall performance.

However, most existing MTL frameworks only consider similar tasks [25, 48] like considering classification and regression of the same learning features [47], or assuming that the features are different but label spaces are the same across tasks [16]. Regarding heterogeneous tasks (i.e., tasks with very different properties such as contrasting objective functions and learning features), these MTL frameworks cannot be effective. In the case of tackling CQA tasks, there are two major challenges for heterogeneous MTL (e.g., jointly solve link prediction, classification, and ranking over different types of nodes).

First, it is non-trivial to share features for heterogeneous CQA data in the semantic space. Specifically, different entities, like users and answers barely share similar semantic space, as they are usually connected through complicated paths in different types of relations. It may be not sufficient for the algorithm to learn the feature dependency, which is crucial for effectively sharing features across tasks, directly from training feature representations. Thus, a multi-relational graph based model should be considered to build relationships among entities in a CQA platform.

Second, there are explicit relationships among output labels in the label space. However, to our best knowledge, existing MTL algorithms do not explicitly model the label relations. For example, in CQA, a more reputable user's answer is more likely to be ranked higher, but the relation of preferences cannot be explicitly reflected by the corresponding representations of users and answers. Thus,

[1]https://stackoverflow.com/

besides data-driven learning, explicit label-based constraints can be imposed to regularize the representation learning across different CQA tasks.

To address the aforementioned challenges, we propose a novel MTL model named Heterogeneous Multi-Task Graph Isomorphism Network (HMTGIN) that efficiently learns multiple heterogeneous CQA tasks on any given CQA forum graph, despite the possibly substantial heterogeneity of the tasks and graph. HMTGIN adopts the hard parameter sharing mechanism [25] to improve efficiency and reduce memory consumption. Inside HMTGIN, we design the Multi-Relational Graph Isomorphism Network (MRGIN), a multi-relational variant of the Graph Isomorphism Network (GIN) [41], and employ it with skip connections [11] to learn the node embeddings shared across tasks. Besides learning from data, HMTGIN also imposes cross-task constraints on tasks' relationships to capture the feature and label dependency across heterogeneous tasks.

Despite some existing MTL works [45, 46] or graph mining [1] on CQA, to the best of our knowledge, no previous MTL algorithm tackles CQA from the aspect of multi-relational graphs.

To evaluate the effectiveness of the proposed approach, we build a novel large-scale multi-relational graph CQA dataset with over two million nodes from Stack Overflow. We then define five different CQA tasks which can be categorized into link prediction, ranking, and classification problems. Extensive experiments are conducted to compare HMTGIN's performance with corresponding baselines in each task and to examine the effect of the proposed MTL mechanism and cross-task constraints.

Our main contributions are as follows:

• We propose a novel MTL model termed HMTGIN that efficiently mines any given CQA forum graph, where the graph and tasks can have immense heterogeneity. To the best of our knowledge, HMTGIN is the first MTL model that tackles CQA tasks from the perspective of multi-relational graphs.

• We construct a novel million-scale multi-relational graph CQA dataset from Stack Overflow.

• We perform extensive experiments of five tasks, where HMTGIN is shown to be superior to all baselines. Further empirical analysis demonstrates the considerable improvements of our MTL strategy and cross-task constraints.

Our dataset and code are publicly available at https://github.com/HKUST-KnowComp/HMTGIN.

2 RELATED WORK

We discuss the related work in four-fold.

2.1 Community Question Answering (CQA)

CQA platforms enable people to seek and share knowledge effectively. For instance, Stack Overflow is a prominent CQA platform about programming, where many developers actively learn from others or share their expertise. Various CQA tasks have been extensively studied [31], like post recommendation [40] and duplicate post detection [33]. In this paper, we study five CQA tasks of the following three types: link prediction, ranking, and classification. Details of these tasks are in Section 3.1. Despite some existing works about MTL for CQA [45, 46], to the best of our knowledge, no previous MTL algorithm can solve CQA tasks from the aspect of multi-relational graphs, and the tasks considered in previous MTL works are relatively homogeneous. In contrast, our approach is the first multi-relational graph based MTL model tackling highly heterogeneous CQA tasks.

2.2 Multi-Task Learning (MTL)

MTL aims to strengthen a model's generalizability by jointly learning multiple related tasks [5, 25, 48]. Successful applications in many domains like Natural Language Processing [21, 26, 30] have verified MTL's effectiveness. Recent MTL models usually employ deep neural networks [21, 26]. Most of them adopt either hard parameter sharing where each task shares the same hidden layers while keeping its output layer, or soft parameter sharing where every task has its model [25]. However, most existing MTL models only undertake tasks with small heterogeneity [25, 48], like considering regression and classification of identical learning features [47], or assuming that features are different but label spaces are the same [16]. Moreover, many existing MTL algorithms are purely data-driven [20, 25, 48], which usually prevents them from capturing the label dependency across tasks due to the large heterogeneity of multi-tasks. In contrast, our MTL model efficiently performs highly heterogeneous learning tasks, where two cross-task constraints are imposed on tasks' relationships to regularize the joint learning.

2.3 Graph Neural Networks

Graph Neural Networks (GNNs) typically perform graph representation learning by recursive neighborhood aggregation [38]. Many GNN variants [19, 42] led to significant advancement in various graph mining tasks like link prediction. Additionally, [41] conducted a systematic study on GNNs' representational power via the connections between GNNs and Weisfeiler-Lehman (WL) graph isomorphism test [37]. Under this framework, [41] proposes Graph Isomorphism Network (GIN) which provably accomplishes the utmost discriminative power among all kinds of GNNs. Different from the above GNNs which only consider homogeneous graphs, several variants [6, 14, 27, 36, 39, 44, 49] have been designed for multi-relational graphs. Although GNNs have demonstrated superb learning abilities, few works adopt GNNs in MTL models [34, 35]. Hence, we devise an extension of GIN termed Multi-Relational GIN as one of the main components of our MTL model to facilitate multi-relational graph representation learning. Besides GIN, our framework can be easily extended to incorporate other GNN architectures to further boost performance.

2.4 Constraint Learning

Constraint learning aims to improve a model's performance by incorporating domain knowledge as constraints. One popular constraint learning framework is the cannot-link and must-link modeling, where constraints are typically based on ground-truth labels [2]. Another powerful scheme is the Constrained Conditional Models (CCM) [7] where learning is separated from the knowledge-aware inference. Furthermore, Posterior Regularization (PR) [10] embodies knowledge by a joint learning and inference method. However, few works apply constraints on tasks' relationships to enhance MTL. Thus, in our MTL model, we impose cross-task constraints on tasks' relationships, which is shown to be effective.

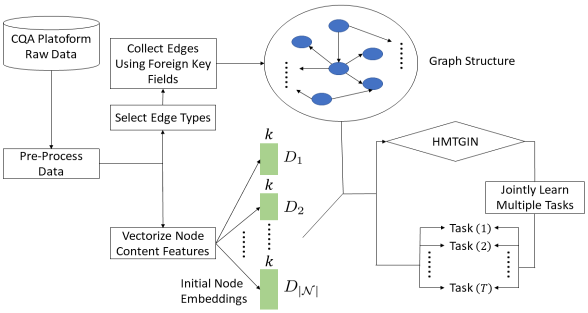


Figure 1: Overview of the generic framework. k is the dimensions of initial node embeddings, and $\mathcal{N}$ is the set of all node types, and $D_i, (i = 1, 2, ..., |\mathcal{N}|)$ denote cardinalities of respective node types, and T is the number of tasks.

3 PRELIMINARIES

In this section, we describe the tasks and frequently used notations. Figure 1 depicts the generic framework for building the multi-relational graph from a CQA raw dataset, and how HMTGIN performs MTL on the built dataset.

3.1 Tasks Description

We divide all tasks into the following three types: link prediction, ranking, and classification.

3.1.1 Link Prediction. Link prediction predicts whether an edge exists between a given node pair, where some of the existing edges are hidden from the input graph. Regarding CQA, link prediction can benefit various applications like recommendations. Here are the two link prediction tasks: (1) Tag recommendation: given a question-tag pair, predict whether the tag belongs to that question; (2) Duplicate question detection: given a question-question, predict whether there is a 'Duplicate' type of edge. To obtain negative examples, we randomly sample twice as many node pairs as the existing links for each task, where no target link exists between any of the sampled node pairs.

3.1.2 Ranking. A ranking task obtains sorts an input item list for a user to quickly identify the desired item(s). Since many CQA platform questions have many associated answers, generating a personalized ranking of the answers can greatly save users' time. Hence, we define a task called answer recommendation as follows: given a question with at least eight answers, provide a ranking of all its answers such that the accepted answer has a high rank. Each sample contains a question index, a list its associated answers' indices, and the list positional index of the accepted answer as the label.

3.1.3 Classification. A classification task categorizes a certain node attribute. Since some attributes like questions' *score* attribute might reveal crucial node properties, categorization of them may help identify important nodes. Here are the two link prediction tasks: (1) Answer score classification: classify an answer score into one of the integers in $[0, 3]$, where a higher value means higher *score* attribute; (2) User reputation classification: classify a user's reputation into one of the integers in $[0, 4]$, where a higher value indicates higher *reputation* attribute. During the data pre-processing, labels for each target attribute are generated in advance by dividing the values into different intervals, and the corresponding attribute is masked in the dataset.

3.2 Notations

Following notations are used throughout the remaining paper: the directed and labeled input multi-relational graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{N}$, and edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where each node $v \in \mathcal{V}$ has one particular node type $\phi(v) \in \mathcal{N}$, and each edge $e \in \mathcal{E}$ has one particular edge type $\psi(e) \in \mathcal{R}$, and $\mathcal{R}$ does not include the self-loop edge type. Additionally, we use Multi-Layer Perceptron (MLP) or *MLP* for a feedforward neural network with *zero or more layers*. Moreover, we use σ for activation functions like RELU. In addition, we use *BN* for Batch Normalization (BN) [15]. Furthermore, we use f for sigmoid function. We also use $\circ$ for constructing the network by stacking different layers.

4 METHODOLOGY

We first present MRGIN. Then we explain all task-specific output layers. Thereafter, we explain the cross-task constraints on tasks' relationships. Finally, we describe the whole MTL algorithm.

4.1 Multi-Relational Graph Isomorphism Network (MRGIN)

We propose MRGIN which, together with skip connections [11], embeds the input multi-relational graph. MRGIN is inspired by GIN [41] that has been theoretically shown to have the maximum discriminative power among all types of GNNs. Specifically, let $d^{(l')}$ be the dimension of (l')-th MRGIN layer's node representation. The representation of a node $v_i \in \mathcal{V}$ in $(l+1)$-th MRGIN layer, denoted as $\mathbf{h_i}^{(l+1)} \in \mathbb{R}^{d^{l+1}}$, is computed as:

$$\mathbf{h_i}^{(l+1)} = \sigma^{(l+1)} \circ BN^{(l+1)} \circ MLP^{(l+1)} \circ \sigma^{(l+1)}\Big(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \mathbf{W_r}^{(l+1)} \mathbf{h_j}^{(l)} + (1 + \epsilon^{(l+1)}) \mathbf{W_0}^{(l+1)} \mathbf{h_i}^{(l)}\Big), \quad (1)$$

where $\mathcal{N}_i^r$ is the set of neighbor indices of node v_i under edge type $r \in \mathcal{R}$, and $\mathbf{h_j}^{(l)} \in \mathbb{R}^{d^{(l)}}$ as well as $\mathbf{h_i}^{(l)} \in \mathbb{R}^{d^{(l)}}$ are the representations of nodes v_j and v_i respectively in l-th MRGIN layer, and both $\mathbf{W_r}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ and $\mathbf{W_0}^{(l+1)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$ are type-specific transformation matrices, and $\epsilon^{(l+1)}$ is a scalar which is either trainable or fixed. Updating one layer is to concurrently evaluate Equation (1) for all nodes.

Briefly speaking, MRGIN aggregates transformed neighboring nodes' representations by sum pooling. Moreover, every node's feature vector is scaled by $1 + \epsilon$ before being aggregated into its representation in the next layer. The vector produced by the summation is then processed via a non-linear activation function, followed by an MLP, a BN, and finally another non-linear activation function. One crucial distinction between MRGIN and GIN is that the transformation matrices in the aggregation of MRGIN depend on

edges' orientations and types, which enables MRGIN to exploit the input graph's structural and relational information.

4.2 Task-Specific Output Layers (TSOLs)

We design TSOLs for task types in Section 3.1 as follows:

4.2.1 Link Prediction. We associate each link prediction task t with a diagonal matrix $\mathbf{D_t} \in \mathbb{R}^{k \times k}$ initialized by a standard normal distribution, where k is the dimension of the input node embeddings for TSOLs. Given a node pair (v_i, v_j) for a task t, we follow the DisMult algorithm [43] to calculate the link prediction score S_{ij} as follows:

$$S_{ij} = \mathbf{h_i'^T D_t h_j'}, \tag{2}$$

where $\mathbf{h_i'}$ and $\mathbf{h_j'} \in \mathbb{R}^k$ is the input node embeddings for TSOLs of v_i and $v_j \in \mathcal{V}$ respectively. Suppose the type of the target link in task t is r_t, define a set of node pairs as follows:

$$\mathcal{H}_t = \{\, (v_i, v_j) \mid v_i \in \mathcal{V}, v_j \in \mathcal{V},\ \exists\, e \in E \text{ from } v_i \text{ to } v_j \text{ such that } \psi(e) = r_t \,\}. \tag{3}$$

Let $\mathcal{H}_t'$ be the set of node pairs obtained by the negative sampling procedure mentioned in Section 3.1 for task t. Then task t's loss L_t is:

$$L_t = -\frac{1}{|\mathcal{H}_t| + |\mathcal{H}_t'|} \sum_{(v_i,v_j) \in \mathcal{H}_t \cup \mathcal{H}_t'} [w \cdot y_{ij} \cdot \log(f(S_{ij})) + (1 - y_{ij}) \cdot \log(1 - f(S_{ij}))], \tag{4}$$

where w is the weight of positive samples (i.e., node pairs in H_t), and $y_{ij} = 1$ if $(v_i, v_j) \in \mathcal{H}_t$ and $y_{ij} = 0$ if $(v_i, v_j) \in \mathcal{H}_t'$.

4.2.2 Ranking. Regarding ranking task t, the parameters of its TSOL include $\mathbf{w_t} \in \mathbb{R}^{2k}$ and $b_t \in \mathbb{R}$. Denote the i-th input sample as $(Q_i, \mathcal{A}(Q_i))$ where Q_i is a question index and $\mathcal{A}(Q_i)$ is a list containing the indices of the question's answers. Denote the sample's label as y_i. Motivated by RankNet [4], we compute the loss term of i-th sample, denoted as L_t^i, as follows: (1) Retrieve the input node embeddings for TSOLs by Q_i and $\mathcal{A}(Q_i)$; (2) Concatenate the question embedding with each of the answer embeddings. Denote the concatenated embeddings as $\mathbf{M_t^i} \in \mathbb{R}^{2k \times |\mathcal{A}(Q_i)|}$; (3) Compute the ranking score of answer in the j-th position of $\mathcal{A}(Q_i)$, denoted as R_j, by

$$R_j = \sigma(\mathbf{w_t^T M_t^i}[:, j] + b_t), \tag{5}$$

where $\mathbf{M_t^i}[:, j]$ is the j-th column of $\mathbf{M_t^i}$ matrix; (4) Calculate the loss term as

$$L_t^i = -\frac{1}{|\mathcal{A}(Q_i)|} \sum_{j=1, j \neq y_i}^{|\mathcal{A}(Q_i)|} [\log(f(R_{y_i} - R_j))]. \tag{6}$$

The loss of this task denoted as L_t is the average of all input samples' loss terms. Since Equation (6) is essentially a pairwise ranking loss, minimizing L_t amounts to maximizing the difference between the ranking score of the accepted answer and those of the other answers in each input sample, which results in recommendations where the ranks of the accepted answers tend to be high.

4.2.3 Classification. Regarding a classification task t, we use a MLP with one layer as its TSOL. The weight and bias of this MLP are $\mathbf{W_t} \in \mathbb{R}^{K_t \times k}$ and $\mathbf{b_t} \in \mathbb{R}^{K_t}$ respectively, where K_t is the number of possible classes in task t. Given an input sample, which is a node embedding vector $\mathbf{h_i'} \in \mathbb{R}^k$, the TSOL first computes the logit vector, denoted as $\mathbf{x_i} \in \mathbb{R}^{K_t}$, by a linear transformation:

$$\mathbf{x_i} = \mathbf{W_t h_i'} + \mathbf{b_t}. \tag{7}$$

Then the loss of task t, denoted as L_t, is calculated by softmax cross-entropy:

$$L_t = -\frac{1}{N'} \sum_{i=1}^{N'} [\log(\frac{\exp(\mathbf{x}[y_i])}{\sum_{j=1}^{K_t} \exp(\mathbf{x}[j])})], \tag{8}$$

where N' is the number of input samples and y_i is the class label of the i-th sample.

4.3 Cross-task Constraints

Besides data-driven learning, our model can be equipped with domain knowledge about tasks' relationships via incorporating several cross-task constraints into the objective function to regularize the joint learning. In this paper, we design two cross-task constraints as a demonstration of our constraint learning framework. Apart from these constraints, our framework can easily incorporate more cross-task constraints exploiting the domain knowledge about tasks' relationships by simply adding extra constraint loss functions into the objective.

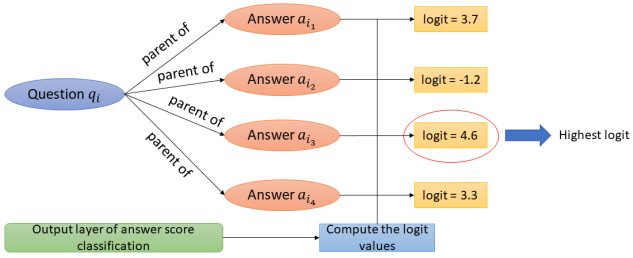
Each of the two cross-task constraints relies on one of the following two assumptions based on the domain knowledge about CQA. Given two answers A_1 and A_2 of a question:

(1) **Assumption 1**: if A_1's *score* attribute is higher than that of A_2, then A_1 is more likely to be accepted;
(2) **Assumption 2**: if A_1's owner's *reputation* attribute is higher than that of A_2, then A_1 is more likely to be accepted.

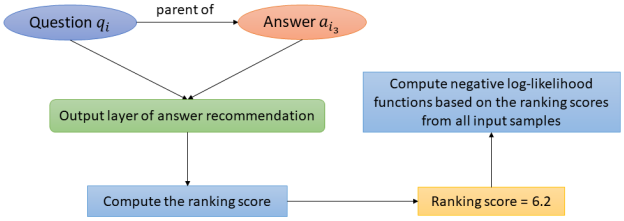
Assumption 1 is reasonable as an answer with higher *score* attribute usually has higher quality. Similarly, Assumption 2 is also legitimate since a user with higher *reputation* attribute typically has higher capability, which implies that the user's answer tends to be better. Hence, we propose the following two cross-task constraints.

The constraint of assumption 1 encourages answer(s) predicted to have the highest score attribute in each sample to rank highly in the recommendation list. Figure (2a)-(2b) illustrate an example of computing this constraint loss. The general idea is: (1) For each input sample (i.e., a question with all of its answers), use the output layer of **answer score classification** to compute all answers' logits; (2) Use the output layer of **answer recommendation** to get the ranking score(s) of answer(s) with the highest logit; (3) Compute negative log-likelihood functions based on these ranking scores from all input samples, which are then averaged to get the overall constraint loss function C_1.

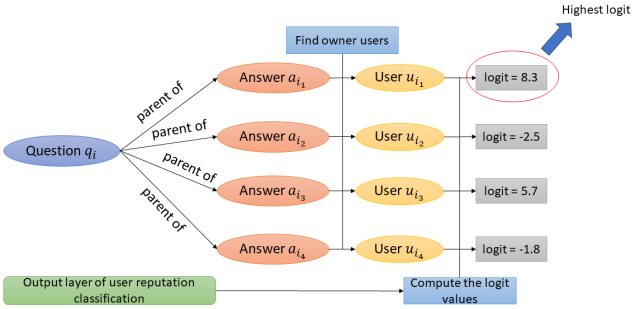
Specifically, we store a sample list $\mathcal{I}_1$ before training, where each sample is a question index with a list of the indices of the question's answers. Let the i-th sample be $(Q_i, \mathcal{A}(Q_i))$ where Q_i is the question index and $\mathcal{A}(Q_i)$ is the answer indices list. This sample's loss term is calculated as follows: (1) Retrieve the corresponding node embeddings by Q_i and $\mathcal{A}(Q_i)$; (2) Calculate the logit values of the



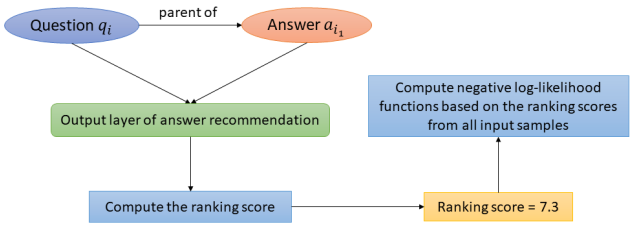
(a) Constraint 1 step (1): compute logits via answer classification output layer and select the answer with highest logit.



(b) Constraint 1 step (2), (3): compute the ranking score(s) of answer(s) with the highest logit, then compute negative log-likelihoods which are averaged to get the overall constraint loss.



(c) Constraint 2 step (1), (2): find owner users and compute their logits via user classification output layer.



(d) Constraint 2 step (3), (4): compute the ranking score(s) of answers whose owner user(s) has the highest logit, then compute negative log-likelihoods which are averaged to get the overall constraint loss.

Figure 2: Examples of computing the proposed two cross-task constraints' loss functions. Figures (a)-(b) illustrate the 1st constraint, while figures (c)-(d) illustrate the 2nd constraint.

answers associated with $\mathcal{A}(Q_i)$ by Equation (7), where the weight and bias come from the TSOL of answer score classification, and the answer embeddings are those retrieved in Step (1); (3) Create a list $\mathcal{I}_1^{'}$ containing the indices of answer(s) with the highest logit value in Step (2); (4) Concatenate the retrieved question embedding with the retrieved embedding of every answer whose index is in $\mathcal{I}_1^{'}$. Denote the concatenated embedding(s) as $\mathbf{M_1^i} \in \mathbb{R}^{2k \times |\mathcal{I}_1^{'}|}$; (5) Obtain the ranking score of every answer whose index is in $\mathcal{I}_1^{'}$ via Equation (5), where the weight, bias, and activation function are from the TSOL of answer recommendation, and the concatenated embeddings are those in $\mathbf{M_1^i}$; (6) Create a list $\mathcal{R}_1^{'}$ containing the ranking scores in Step (5), and then compute the loss term C_1^i as:

$$C_1^i = -\frac{1}{|\mathcal{R}_1^{'}|} \sum_{j=1}^{|\mathcal{R}_1^{'}|} [\log(f(\mathcal{R}_1^{'}[j]))]. \tag{9}$$

The first constraint loss C_1 is the average of loss terms of all samples in $\mathcal{I}_1$. Since minimizing Equation (9) amounts to maximizing the ranking scores of answers corresponding to $\mathcal{I}_1^{'}$, reducing C_1 will encourage answer(s) which is predicted to have the highest value of *score* attribute in each sample to rank highly, which enhances the consistency between the answer score classification and answer recommendation tasks.

The constraint of assumption 2 encourages the answer(s) whose user is predicted to have the highest reputation in each sample to have higher rank in the recommendation list. Figure (2c)-(2d) illustrate an example of computing this constraint loss. The general idea is: (1) For each input sample (i.e., a question with all of its answers), find the answers' owner users; (2) Use the output layer of **user classification** to compute the logits of all these users; (3) Use the output layer of **answer recommendation** to get the ranking score(s) of answer(s) whose owner(s) has the highest logit; (4) Compute negative log-likelihood functions based on these ranking scores from all samples as in Equation 10 (the notations are similar to those in Equation 9 for the first constraint), which are then averaged to get the overall constraint loss C_2.

$$C_2^i = -\frac{1}{|\mathcal{R}_2^{'}|} \sum_{j=1}^{|\mathcal{R}_2^{'}|} [\log(f(\mathcal{R}_2^{'}[j]))]. \tag{10}$$

Because minimizing Equation (10) is equivalent to maximizing the ranking scores of answers corresponding to $\mathcal{I}_2^{'}$, decreasing C_2 will provide the answer(s) whose user is predicted to have the highest reputation in each sample with more chance of having a high rank in the recommendation list, which strengthens the consistency between the user reputation classification and answer recommendation tasks.

Admittedly, this approach of designing constraints is somewhat ad-hoc. Nonetheless, these cross-task constraints, which reflect human's domain knowledge of the tasks, not only are lucid and easy to implement but also significantly boost our model's performance as verified in the experiments.

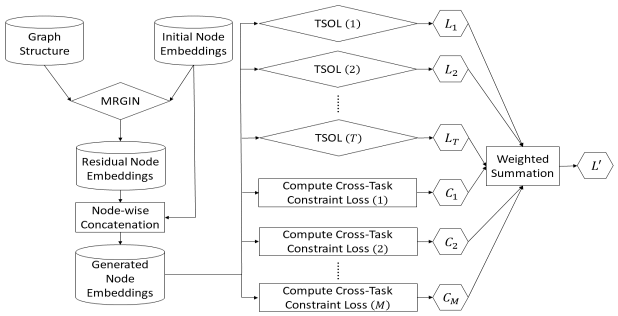


Figure 3: Generic overview of HMTGIN. *TSOL* (i) denotes the task-specific output layer of i-th task.

4.4 Heterogeneous Multi-Task Graph Isomorphism Network (HMTGIN)

Finally, we illustrate our MTL algorithm called HMTGIN. Suppose the input multi-relational CQA graph consists of graph structure and initial node embeddings, and there are T tasks and M cross-task constraints. The training process of HMTGIN is as follows: (1) The MRGIN model defined in Section 4.1 learns new node embeddings by taking in the initial node embeddings and then performing neighborhood aggregations based on the graph structure. The new node embeddings are termed *residual node embeddings*; (2) Skip-connection technique is adopted by performing node-wise concatenation between the initial node embeddings and the residual node embeddings. We call these concatenated embeddings as *generated node embeddings*; (3) The generated node embeddings are shared by every TSOL to calculate corresponding loss $L_t (t \in \{1, 2, ..., T\})$ as described in Section 4.2; (4) Cross-task constraint losses $C_1, C_2, ..., C_M$ are computed using respective generated node embeddings as described in Section 4.3; (5) Compute the total loss L' as:

$$L' = \frac{1}{T} \sum_{t=1}^{T} \alpha_t L_t + \sum_{i=1}^{M} \beta_i C_i, \tag{11}$$

where $\alpha_t (t \in \{1, 2, ..., T\})$ and $\beta_i (i \in \{1, 2, ..., M\})$ are pre-defined constant real numbers. (6) Update HMTGIN's learnable parameters by back-propagation with respect to the total lossL'; (7) Repeat Step (1) - (6) until the termination condition is met. In evaluation, the generated node embeddings after the last training epoch are shared across different TSOLs to make corresponding predictions. A generic overview of our model is shown in Figure 3 .

5 EXPERIMENTS

In this section, we present our experiments.

5.1 Dataset

We obtained the Stack Overflow raw dataset from the Stack Exchange Data Dump[2]. We then performed data cleaning steps like removing attributes with too many missing values. Afterward, we constructed a million-scale multi-relational graph consisting of questions with at least eight answers, all other types of nodes that

[2]https://archive.org/details/stackexchange

Table 1: Cardinalities of all node types.

Type	Cardinality
Questions	108,113
Answers	1,212,308
Users	773,517
Tags	55,663

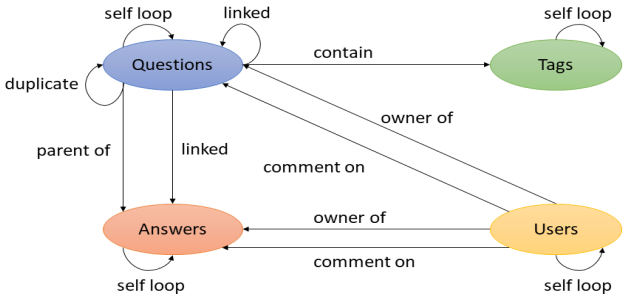


Figure 4: Schema of the Stack Overflow dataset.

are either directly or indirectly connected to these questions, and associated edges. The graph has 4 node types and 22 edge types including self loop edge types for all node types and all reverse edge types[3]. Thus, in our experiments, we do not compare with ordinary GCN and GIN as doing so will lose a lot of edges. Each node type's cardinality is included in Table 1, where the total number of nodes is over two million. Figure 4 depicts the graph's schema[4].

Every node has several attributes (e.g., 'Body' attribute of Questions). An attribute would be ignored if it appears in less than 10% of the corresponding type of nodes. Regarding any date-time attribute (e.g., 'CreationDate' of Questions), it will be ignored if it is missing in any of the corresponding types of nodes. We performed several text pre-processing operations like expanding contractions on every textual attribute (e.g., 'Body' attribute of Answers). Thereafter, we constructed the initial feature vector for an attribute A as follows: (1) If A is textual, we numericalize it by averaging all its words' pre-trained GloVe [24] embeddings. A vector filled with 0 is used for missing values; (2) If A is categorical (e.g., 'PostTypeId' attribute of Posts of other types), then we convert it into a one-hot vector. The value of A will be set to -1 before conversion if its value is missing; (3) If A is numerical (e.g., 'Views' attribute of Users), then we normalize its value into the range $[0, 1]$. The value of A will be set to 0 before conversion if its value is missing; (4) If A is a date-time attribute, we encode its different aspects of temporal information (e.g., what weekday it is) into a numeric vector. The initial feature vector of every node is then the concatenation of all its attributes' initial feature vectors. We applied SVD to reduce the dimension of all nodes' initial feature vectors to 16 for efficiency. These compressed vectors serve as the initial node embeddings. All

[3]Any two edges which have either different source node types or different destination node types are considered to be of different edge types (e.g., 'owner_of' edge type between Users and Questions are considered to be different from that between Users and Answers).

[4]Reverse edge types are omitted for conciseness.

Table 2: Baselines and evaluation metrics for all tasks.

Tasks	Baselines	Metrics
Tag Recommendation	WDL, NCF, TransE COMPGCN, KBGAT	Accuracy, F1
Duplicate Question Detection	CDSSM, MaLSTM, TransE COMPGCN, KBGAT	Accuracy, F1
Answer Recommendation	WDL, NCF COMPGCN	HR@3, NDCG@3
Answer Score Classification	Text-CNN, BiLSTM COMPGCN, BERT	Accuracy, Macro F1
User Reputation Classification	Text-CNN, BiLSTM COMPGCN, BERT	Accuracy, Macro F1

experiments are only performed on the constructed Stack Overflow dataset as our approach can be directly applied to other CQA platforms with trivial modifications.

5.2 Baselines

We compare our model with the following algorithms: (1) **WDL** [8]; (2) **NCF** [12]; (3) **CDSSM** [29]; (4) **MaLSTM** [22]; (5) **Text-CNN** [17] ; (6) **BiLSTM** [28]; (7) **TransE** [3]; (8) **COMPGCN** [32]; (9) **KBGAT** [23]; (10) **BERT (Large)** [9]. We do not include any of the existing MTL models for CQA in the experiment since none of them can jointly learn the highly heterogeneous tasks targeted in this paper.

Baseline models for the ranking task are extended to incorporate the loss function of RankNet [4] to boost performance. Besides the above baselines, to verify the effectiveness of MRGIN, we also compare HMTGIN with one of its variants termed Heterogeneous Multi-Task Graph Convolutional Networks (**HMTGCN**) which is the same as HMTGIN except that it replaces the MRGIN component of HMTGIN with RGCNs. The baseline algorithms and evaluation metrics for all tasks are included in Table 2, where F1 means F1 score, HR@3 means Hit Ratio in top-3 list, NDCG@3 means Normalized Discounted Cumulative Gain in top-3 list, and Macro F1 means macro F1 score for multi-class classification[5]. The word embeddings are fixed during training for CDSSM, MaLSTM, Text-CNN, BiLSTM, and BERT.

5.3 Settings

We split each task's dataset into training, development, and test sets with an $8:1:1$ ratio. The best hyper-parameter configuration of every model is determined by evaluating the trained models on corresponding development set(s), where the configuration with the best average score over all relevant evaluation metrics is chosen. Multi-task models are compared by their average scores over all respective evaluation metrics on the development sets of all corresponding tasks. Afterward, the trained model with the chosen configuration is examined on the test set(s) to get its test scores for all relevant evaluation metrics. Every instance of a baseline model is trained and evaluated on only one task, whereas every instance of an MTL model is trained and evaluated on all tasks.

[5]We omit micro F1 score as it always has the same value as accuracy does in this setting.

Table 3: Performance comparison in the TR task.

Models \ Metrics	Accuracy	F1
WDL	0.870 ± 0.002	0.786 ± 0.001
NCF	0.871 ± 0.001	0.786 ± 0.001
TransE	0.879 ± 0.000	0.795 ± 0.000
COMPGCN	0.895 ± 0.002	0.810 ± 0.001
KBGAT	0.882 ± 0.001	0.803 ± 0.001
HMTGCN	0.911 ± 0.002	0.866 ± 0.002
HMTGIN	**0.921** ± 0.000	**0.878** ± 0.000

The hyper-parameters we have tuned for HMTGIN as follows, where the values used in the best configuration of HMTGIN are shown in bold.

(1) number of hidden layer: **0**, 1, 2, 3;
(2) hidden dimension(s) : 8, **16**, 32, 64;
(3) whether the ϵ coefficient in Equation 1 is trainable or not: yes, **no**;
(4) dropout rate in all MRGIN layer: **0**, 0.1, 0.2, 0.3;
(5) number of mlp layer in each MRGIN layer: 0, **1**, 2, 3, 4.

We train each HMTGIN instance for 135 epochs, where the checkpoint is saved whenever the average score increases. The coefficients of both cross-task constraints are 1. The coefficients associated with the losses of both duplication question detection and answer score classification are 7, whereas the others are 1. The model is trained using Adam [18] algorithm in full batch. The σ activation function in Equation 1 and Equation 5 is Leaky ReLU. The value of the ϵ coefficient in Equation 1 is 0. The learning rate is reduced by 0.5 in every 50 epochs.

5.4 Performance Comparison

We abbreviate task names as follows: '**TR**' means tag recommendation; '**DQD**' denotes duplicate question detection; '**AR**' means answer recommendation; '**ASC**' denotes answer score classification; '**URC**' means user reputation classification. Regarding each task, we train and test the corresponding model instances with their respective best hyper-parameters under three different random seeds. The mean test scores and the standard deviations of all models are shown in Table 3, 4, 5, 6, and 7 respectively [6]. The best mean score in each task for every corresponding evaluation metric is marked in bold.

In summary, HMTGIN outperforms all baselines in all tasks, where the improvements are up to 16.9%, which distinctly demonstrates its effectiveness.

Regarding link prediction, the improvements in accuracy are up to 12%, and the improvements in F1 score are up to 13.3%. Regarding ranking, the improvements are up to 11.3% and 12.9% in HR@3 and NDCG@3 respectively. Such considerable improvements indicate that the proposed cross-task constraints are significant. Regarding classification, all HMTGIN's scores in URC task substantially exceed those of the corresponding best baselines (the improvements over

[6]The numbers to the left of '±' symbols are the mean scores and the numbers to the right of '±' symbols are the corresponding standard deviations

Table 4: Performance comparison in the DQD task.

Models \ Metrics	Accuracy	F1
CDSSM	0.795 ± 0.004	0.656 ± 0.003
MaLSTM	0.794 ± 0.003	0.601 ± 0.002
TransE	0.683 ± 0.001	0.552 ± 0.001
COMPGCN	0.702 ± 0.003	0.583 ± 0.002
KBGAT	0.721 ± 0.002	0.591 ± 0.002
HMTGCN	0.773 ± 0.001	0.645 ± 0.002
HMTGIN	**0.803** ± 0.001	**0.685** ± 0.001

Table 5: Performance comparison in the AR task.

Models \ Metrics	HR@3	NDCG@3
WDL	0.624 ± 0.001	0.49 ± 0.001
NCF	0.631 ± 0.001	0.498 ± 0.001
COMPGCN	0.697 ± 0.001	0.583 ± 0.002
HMTGCN	0.717 ± 0.002	0.589 ± 0.003
HMTGIN	**0.737** ± 0.000	**0.619** ± 0.000

Table 6: Performance comparison in the ASC task.

Models \ Metrics	Accuracy	Macro F1
Text-CNN	0.421 ± 0.001	0.423 ± 0.002
BiLSTM	0.435 ± 0.004	0.438 ± 0.005
COMPGCN	0.331 ± 0.002	0.342 ± 0.001
BERT	0.442 ± 0.001	0.438 ± 0.001
HMTGCN	0.434 ± 0.002	0.429 ± 0.002
HMTGIN	**0.444** ± 0.001	**0.439** ± 0.002

Table 7: Performance comparison in the URC task.

Models \ Metrics	Accuracy	Macro F1
Text-CNN	0.395 ± 0.003	0.393 ± 0.003
BiLSTM	0.405 ± 0.001	0.396 ± 0.003
COMPGCN	0.314 ± 0.001	0.302 ± 0.001
BERT	0.459 ± 0.001	0.436 ± 0.001
HMTGCN	0.423± 0.002	0.417 ± 0.001
HMTGIN	**0.483** ± 0.002	**0.470** ± 0.001

accuracy and macro F1 are 2.4% and 3.4% respectively), whereas in ASC task, HMTGIN only slightly outperforms the corresponding best baselines. This might result from that the best baseline is already powerful enough for ASC task, which leads to a small room for further improvement. By comparing the performance of HMTGIN and HMTGCN, we can see that HMTGIN consistently outperforms HMTGCN in all tasks, where the improvements are

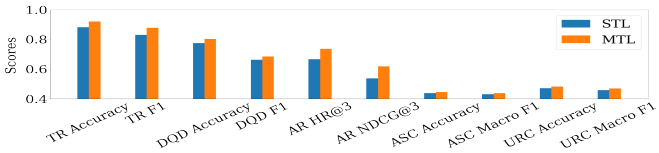


Figure 5: Comparison between STL and MTL settings of HMTGIN.

up to 6%, and in two out of the five tasks, the improvements are at least 3%. This result shows that GIN is more powerful than GCN.

5.5 Ablation Studies

We conduct the following ablation studies: (1) comparing Single Task Learning (STL) settings with MTL settings; (2) examining the influence of the cross-task constraints. (3) parameter sensitivity study on the number of MLP layers in each MRGIN layer. Every numerical result is the average over three random seeds.

5.5.1 STL Settings VS MTL Settings. The STL scores together with the MTL ones are shown in Figure 5, and the x-axis contains the task names and respective metrics.

Generally speaking, the MTL instance outperforms all of its STL counterparts for all evaluation metrics (i.e., there is no negative transfer), where the improvements are up to 8%, which justifies the effectiveness of our MTL strategy. In particular, the MTL instance surpasses the STL instances by a large margin in TR (3.8% and 4.7% in accuracy and F1 score respectively), DQD (2.8% and 2.2% in accuracy and F1 score respectively), and AR (7.1% and 8% in HR@3 and NDCG@3 respectively). By contrast, the performance gain in the ASC task (0.7% in both accuracy and macro F1 score) and URC task (1.2% and 1.1% in accuracy and macro F1 score respectively) is not very noticeable. Such a distinction indicates that learning signals from the MTL mechanism are more crucial for the first three tasks than the last two tasks.

5.5.2 Role of Cross-task Constraints. We conduct experiments on the following settings of cross-task constraints to examine their significance, where the results are included in Figure 6: (1) *No constraint* means removing both constraints described in Section 4.3; (2) *Constraint 1 only* denotes retaining only constraint 1, and similarly for *Constraint 2 only*; (3) *Both 0.5* denotes keeping both constraints, and setting both their coefficients as 0.5, and similarly for *Both 1*[7] and *Both 2*. We only show and discuss the F1 scores for link prediction tasks, NDCG@3 score for ranking task, and macro F1 scores for classification task here due to space limitation, and the pattern of the scores which are not included here is almost the same. The x-axis is similar to that of Figure 5, where the 'Average' includes the average value over all the five shown scores of the instance under each of the above five settings.

In general, *Both 1* instance achieves the best average score, which shows that the proposed cross-task constraints are beneficial to the overall performance. Moreover, keeping both constraints almost always improves performance in the tasks related to the constraints (i.e., AR, ASC, and URC), except in a few cases where there are still

[7] Configuration of the HMTGIN instance in Section 5.4 corresponds to this setting.

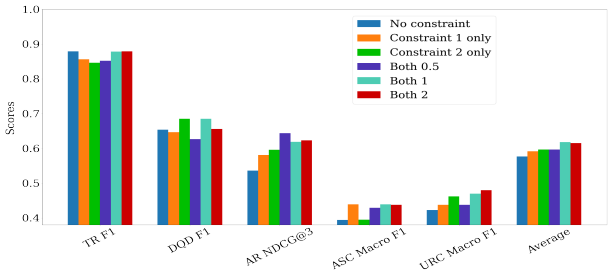


Figure 6: Comparison between different settings of HMTGIN's cross-task constraints.

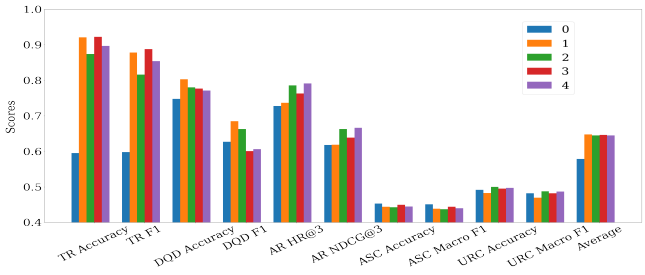


Figure 7: Comparison between different numbers of the MLP layer in each MRGIN layer.

some settings that retain both constraints and have the best scores for the corresponding evaluation metrics. For instance, *Both 0.5*, *Both 1*, and *Both 2* instances all outperform the remaining settings in both evaluation metrics for AR, where the improvements for NDCG@3 range from 2.3% to 25% respectively. These enhancements are attributed to the cross-task constraints which strengthen the consistency between related tasks, as illustrated in Section 4.3. Specifically, *No constraint* and *Constraint 1 only* instances consistently perform worse than the setting *Constraint 2 only* in URC (the average difference for macro F1 is 3.2%), which empirically justifies the analyses in Section 4.3 as the constraint 2 is established on the relationship between URC and AR. In addition, *Constraint 1 only* instance surpasses both *Constraint 2 only* and *No constraint* instances in ASC (the average improvements in macro F1 is 8.9%), which is also consistent with the explanation in Section 4.3 since the constraint 1 reinforces the coherence between AR and ASC. Another interesting observation is that the average score of *Both 0.5* instance is quite close to those of *Constraint 1 only*, *Constraint 2 only* and *No Constraint* instances, where the improvements range from 0.1% to 2.1%. This might be because the small weights of both coefficients weaken the regularization power of the constraints. However, the average score of the *Constraint 2 only* instance is worse than that of the *Constraint 1 only* instance, although the gap is tiny (0.3%). This is probably because the high weights of both coefficients somewhat impede the optimization of losses. Accordingly, choosing appropriate values of both constraints' coefficients is critical.

5.5.3 Parameter sensitivity study on the number of MLP layers in each MRGIN layer. The performance with different number of MLP layer in each MRGIN layer is shown in Figure 7, where the *0* is the instance with no MLP module , the *1* is the instance with 1 MLP layer, and so on[8].

To summarize, the presence of MLP layer greatly boosts the performance (the average improvements are up to 6.9%), which can be attributed to the universal approximation theorem [13] ensuring that MLP can learn the composition of the multiset functions that allows GIN to have the maximum discriminative power among all GNN variants [41]. Furthermore, the instance with one MLP layer has the best average score (i.e., 64.8%), even though the other instances with MLP module are only slightly weaker than it (the largest margin in the average score is only 0.3%). This indicates that altering the number of MLP layer in each MRGIN layer of HMTGIN have limited effect when there is at least one MLP layer in each MRGIN layer. Moreover, increasing the number of MLP layer does not exhibit any generally monotonic trend in the average scores (the changes from zero MLP layer to one MLP layer, from one to two MLP layers, from two MLP layers to three MLP layers, and from three MLP layers to four MLP layers are 6.9%, −0.3%, 0.1% and −0.1% respectively). The performance gain offered by the MLP module is most distinct in TR, where the improvements from the instance without MLP layer to instances with MLP layer are up to 32.7% and 29.0% for accuracy and F1 respectively. Such considerable improvements might be since the MLP layer is highly effective in capturing the tags' semantics. By contrast, the instances with MLP module tend to perform worse compared with those without MLP module in classification tasks, especially in ASC, where the instance without MLP layer outperforms all the instances with MLP layer in both evaluation metrics (the score differences range from 0.3% to 1.0% and 0.7% to 1.4% for accuracy and macro F1 respectively). Such results suggest that MLP is not very helpful for distinguishing the data characteristics in MTL.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a multi-relational graph based MTL model named HMTGIN which efficiently tackles CQA with considerable task and graph heterogeneity. To the best of our knowledge, HMTGIN is the first MTL model solving CQA tasks from the angle of multi-relational graphs. To evaluate the effectiveness of HMTGIN, we build a novel million-scale multi-relational graph CQA dataset. Experiments of five tasks demonstrate that HMTGIN surpasses all baselines in all tasks. Further ablation studies manifest the substantial role of the proposed cross-task constraints and MTL strategy. An interesting future direction would be investigating how to develop a more principled way to incorporate the domain knowledge by imposing cross-task constraints on multi-relational graph based MTL algorithms for CQA tasks.

7 ACKNOWLEDGEMENT

The authors of this paper were supported by the NSFC Fund (U20B2053) from the NSFC of China, the RIF (R6020-19 and R6021-20) and the GRF (16211520) from RGC of Hong Kong, the MHKJFS (MHP/001/19) from ITC of Hong Kong, with special thanks to the WeChat-HKUST WHAT Lab on Artificial Intelligence Technology.

[8]The *1* instance corresponds to the configuration in Section 5.4.

REFERENCES

[1] Leman Akoglu, Duen Horng Chau, U Kang, Danai Koutra, and Christos Faloutsos. 2012. Opavion: Mining and visualization in large graphs. In *SIGMOD*. ACM, 717–720.

[2] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *SIGKDD*. ACM, 59–68.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[4] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *ICML*. ACM, 89–96.

[5] Rich Caruana. 1997. Multitask Learning. *Machine Learning* 28, 1 (1997), 41–75.

[6] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *SIGKDD*. ACM, 1358–1368.

[7] Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2012. Structured learning with constrained conditional models. *Machine learning* 88, 3 (2012), 399–431.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *DLRS@RecSys*. ACM, 7–10.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. ACL, 4171–4186.

[10] Kuzman Ganchev, João GraÃ, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* 11, 67 (2010), 2001–2049.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. IEEE, 770–778.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. ACM, 173–182.

[13] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991), 251–257.

[14] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. ACM, 2704–2710.

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. PMLR, 448–456.

[16] Xin Jin, Fuzhen Zhuang, Sinno Jialin Pan, Changying Du, Ping Luo, and Qing He. 2015. Heterogeneous Multi-task Semantic Feature Learning for Classification. In *CIKM*. ACM, 1847–1850.

[17] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. ACL, 1746–1751.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*. OpenReview.net.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[20] Hui Li, Yanlin Wang, Ziyu Lyu, and Jieming Shi. 2020. Multi-task Learning for Recommendation over Heterogeneous Information Network. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[21] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *ACL*. ACL, 1–10.

[22] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI*. AAAI Press, 2786–2792.

[23] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *ACL*. ACL, 4710–4723.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. ACL, 1532–1543.

[25] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR* abs/1706.05098 (2017).

[26] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *AAAI*. AAAI Press, 4822–4829.

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

[28] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[29] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*. ACM, 101–110.

[30] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*. ACL, 231–235.

[31] Ivan Srba and Maria Bielikova. 2016. A comprehensive survey and classification of approaches for community question answering. *ACM Transactions on the Web* 10, 3 (2016), 1–63.

[32] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[33] Liting Wang, Li Zhang, and Jing Jiang. 2020. Duplicate question detection with deep learning in stack overflow. *IEEE Access* 8 (2020), 25964–25975.

[34] Menghan Wang, Yujie Lin, Guli Lin, Keping Yang, and Xiao-ming Wu. 2020. M2GRL: A Multi-task Multi-view Graph Representation Learning Framework for Web-scale Recommender Systems. In *SIGKDD*. ACM, New York, NY, USA.

[35] Wenlin Wang, Hongteng Xu, Zhe Gan, Bai Li, Guoyin Wang, Liqun Chen, Qian Yang, Wenqi Wang, and Lawrence Carin. 2020. Graph-Driven Generative Models for Heterogeneous Multi-Task Learning. In *AAAI*. AAAI Press, 979–988.

[36] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2022–2032.

[37] Boris Weisfeiler and Andrei A Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia* 2, 9 (1968), 12–16.

[38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020), 1–21.

[39] Wenyi Xiao, Huan Zhao, Haojie Pan, Yangqiu Song, Vincent W Zheng, and Qiang Yang. 2019. Beyond personalization: Social content recommendation for creator equality and consumer satisfaction. In *SIGKDD*. ACM, 235–245.

[40] Fei Xu, Zongcheng Ji, and Bin Wang. 2012. Dual role model for question recommendation in community question answering. In *SIGIR*. ACM, 771–780.

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*. OpenReview.net.

[42] 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*. PMLR, 5449–5458.

[43] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*. OpenReview.net.

[44] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *CoRR* abs/2004.00216 (2020).

[45] Min Yang, Lei Chen, Xiaojun Chen, Qingyao Wu, Wei Zhou, and Ying Shen. 2019. Knowledge-enhanced Hierarchical Attention for Community Question Answering with Multi-task and Adaptive Learning.. In *IJCAI*. ijcai.org, 5349–5355.

[46] Min Yang, Wenting Tu, Qiang Qu, Wei Zhou, Qiao Liu, and Jia Zhu. 2019. Advanced community question answering by leveraging external knowledge and multi-task learning. *Knowledge-Based Systems* 171 (2019), 106–119.

[47] Xiaolin Yang, Seyoung Kim, and Eric P. Xing. 2009. Heterogeneous multitask learning with joint sparsity constraints. In *NIPS*. Curran Associates, Inc., 2151–2159.

[48] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. *CoRR* abs/1707.08114 (2017).

[49] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *SIGKDD*. ACM, 635–644.