Noname manuscript No.
(will be inserted by the editor)

An Induced Multi-Relational Framework for Answer Selection in Community Question Answer Forums

Kanika Narang · Chaoqi Yang · Adit Krishnan ·
Junting Wang · Hari Sundaram · Carolyn Sutter

the date of receipt and acceptance should be inserted later

Abstract Individuals often visit Community Question Answer (CQA) forums seeking answers to nuanced questions that are hard to address with a textual search. In this paper, we consider the question of identifying the best candidate answer to a question on these forums. We propose a novel approach to leverage the explicit and implicit connections across different questions and answers on the forum via induced relational graph convolutional networks (IR-GCN). We make three contributions. First, we introduce a modular framework that separates the construction of content graphs from the label selection mechanism. We use equivalence relations across content to induce graphs comprising cliques and enable two complementary label selection mechanisms; label contrast, and label sharing, via graph convolutional operators. Second, we show that encoding label contrast in this manner creates discriminative magnification, enhancing the separation between contrasting nodes in the latent representation space. Third, we show a surprising result—applying familiar boosting techniques across our content graphs outperforms the popular stacking, fusion, or aggregation methods for neural architectures. We show strong results over state-of-the-art neural baselines with extensive experiments across 50 StackExchange[1] communities.

1 Introduction

Individuals often visit Community Question Answer (CQA) forums, like StackExchange, to seek answers to nuanced questions that are not readily answered via web-search engines. Unlike other familiar Learning-to-Rank problems in the IR community [3,4], CQA platforms can identify and leverage past questions asked by similar users and relevant answers to those questions. However, there is only limited work in the context of identification of "best answers" among user-generated content that exploit these implicit and explicit connections.

A well-studied approach is to identify salient features for each question-answer tuple (q, a) in an inductive supervised classification setting [2,17,32]. In this vein, neural text

K. Narang, C. Yang, A. Krishnan, J. Wang, H. Sundaram, C. Sutter
University of Illinois at Urbana-Chamaign
Urbana, IL, USA

[1] https://stackexchange.com/

models exploit textual features [38,36,33] by learning effective representations of (q, a) tuples. While the neural feature models are effective, there are limitations to examining (q, a) tuples in isolation: an answer is adjudged ”best” *in relationship* to other answers to the same question. Further, considering other answers to similar questions, as well as those from similar users, can provide cues to answer quality. These relational aspects of user-generated content provide a unique dimension that is absent in textual search. Our key proposal is to build a flexible and expressive framework to incorporate the relational aspects of user-generated content for the answer selection task.

Relational aspects are best captured as graphs connecting content. Graph Convolutional Networks (GCNs) are shown to be an effective approach to incorporate attributed graph structure in tasks such as node classification [19] and link prediction [27]. While GCNs are a plausible approach, we need to overcome a fundamental implicit assumption in prior work before we can apply it to our problem. Prior work in GCNs adopt label sharing amongst nodes; label sharing implicitly assumes similarity between two nodes connected by an edge. Extensions to the basic GCN model such as signed networks [7] and multi-relation networks [39,27] do not address the fundamental challenge of modeling label contrast. For instance, in the answer selection problem, if we link together answers to the same question, they do not share acceptance labels. We label an answer as ‘accepted’ by contrast to other answers to the same question. In other words, the relational views (or graphs) could capture similarity or contrast between connected content, depending on the relation in consideration.

We develop a novel framework to model the diverse relations between content through a separate *induced* graph across (q, a) tuples. The key idea is to then use distinct label selection mechanisms depending on the semantics of the relational view. For instance, in the trivial case, the label depends only on the answer features in the *reflexive* view (i.e., no edges), or the label contrasts to other connected answers in a *contrastive* view, or the label is shared among similar answers to different questions in a *similarity* view. This generalizes to a broader principle: pick equivalence relations to induce graphs comprising cliques, and then pick an appropriate label selection mechanism (label sharing or label contrast) for the graph. We show how to develop convolutional architectures to achieve the sharing and contrast label selection mechanisms. Then, we aggregate results across different relational views through a boosting framework to identify the label for each (q, a) tuple. In summary, our contributions are as follows:

Modular, Induced Relational Framework: We introduce a modular framework that separates the construction of graphs from the label selection mechanism. While prior work in answer selection considers content in isolation [2,17], we use equivalence relations to induce graphs comprising cliques and apply label contrast or label sharing to each graph in our Induced Relational GCN (IR-GCN) framework. Our framework applies to other application semantics involving graphs [1].

Discriminative Semantics: We develop a label contrast GCN to differentiate connected vertices in a contrastive view. While prior work in graph convolution (e.g., [19,39]) emphasizes node similarity or edge similarity [7], we show that our contrast encoding creates *discriminative magnification*. We enhance the separation of contrasting nodes in the embedding space.

Boosted Architecture: We show through extensive empirical results that boosting techniques applied across relational views improves learning in our convolutional model. Much of the past work on neural architectures develop stacking, fusion, or aggregator architectures to incorporate multiple views. In contrast, boosting proves a simple and effective strategy in the multi-view setting.

We conducted extensive experiments using our IR-GCN framework with excellent experimental results on the popular CQA forum—StackExchange. For our analysis, we collect data from 50 communities—the ten largest communities from each of the five StackExchange[2] categories. We achieved an improvement of over 4% in accuracy and 2.5% in MRR, on average, over state-of-the-art baselines. We also show that our model is more robust to label sparsity compared to multi-relational GCNs. [3]

We organize the rest of this paper as follows. In section 2, we formulate our problem statement and then discuss induced relational views for the answer selection problem in section 3. We then detail the modeling of these views in our convolution framework in section 4 and introduce our gradient boosting aggregation approach in section 5. In section 6, we describe our experiments, related work in section 7 and then conclude in section 8.

2 Problem Formulation

In Community Question Answer (CQA) forums, an individual asking a question seeks to identify the most relevant candidate answer to his question. On StackExchange CQA forums, users annotate their preferred answer as "accepted".

Let $\mathcal{Q}$ denote the set of questions in the community and for each $q \in \mathcal{Q}$, we denote $\mathcal{A}_q$ to be the associated set of answers. Each question $q \in \mathcal{Q}$, and each answer $a \in \mathcal{A}_q$ have authors $u_q, u_a \in \mathcal{U}$ respectively. Without loss of generality, each question q, each answer $a \in \mathcal{A}_q$, user $u_q, u_a \in \mathcal{U}$ have an associated set of features.

Our unit of analysis is a question-answer tuple $(q, a), q \in \mathcal{Q}, a \in \mathcal{A}_q$, and we associate each (q, a) tuple with a label $y_{q,a} \in \{-1, +1\}$, where '+1' implies acceptance and '-1' implies rejection. The goal of this paper is *to develop a framework to identify the accepted answer to a question posted on a CQA forum.*

3 Induced Relational Views

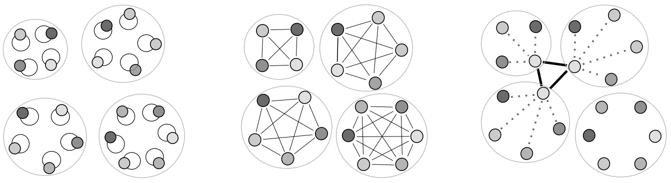
In this section, we discuss the notion of induced relational views, which is central to our induced relational GCN framework developed in Section 4. First, in Section 3.1, we introduce potential strategies for selecting the accepted answer to a question. We show how each strategy induces a graph G on the set of all question-answer (q, a) tuples. Next, in Section 3.2, we show how each of these sample strategies is an instance of an equivalence relation; our framework generalizes to incorporate any such relation.

3.1 Constructing Induced Views

In this section, we discuss four sample strategies that represent strategies to label an answer as 'accepted.' Each strategy $S_i \in \mathbf{S}$ *induces* a graph $G_i = (V, E_i)$ (also referred to as a relational view). In each graph G_i, a vertex $v \in V$ represents a tuple (q, a) and an edge $e \in E_i, E_i \subseteq V \times V$ connects two tuples that are related under that strategy or view. Note that each G_i has the same vertex set, while edge sets E_i depend on strategy S_i. Each strategy employs one of the three different relation types, reflexive, contrastive, and similarity to

[2] https://stackexchange.com/sites

[3] We provide Reddit (https://www.reddit.com/) results in our supplementary material using expert answers as a proxy for acceptance, to overcome the absence of explicit labels.



(a) Reflexive (b) Contrastive (c) Similarity by Contrast

Fig. 1: Reflexive(fig. 1a), Contrastive (fig. 1b) and Similarity by Contrast (fig. 1c) relations among (q, a) tuples. Reflexive assumes no dependence on other answers for prediction. Contrastive compares all answers to a question; Similarity by Contrast connects answers across questions if they differ from other answers similarly. Solid lines indicate similarity, while dotted lines indicate contrast. The contrast is only significant in three questions in the above example.

connect the tuples. We use one reflexive strategy, one contrastive, and two similarity strategies in our experiments. Figure 1 summarizes the three relations. We organize the below discussion by relation type.

3.1.1 Reflexive

A natural strategy is to examine each (q, a) tuple in isolation and then assign a label $y_{q,a} \in \{-1, +1\}$ corresponding to 'not accepted' or 'accepted.' In this case, $y_{q,a}$ depends on only the features of (q, a). This is a **Reflexive** relation, and the corresponding graph $G_r = (V, E_r)$ has a specific structure. In particular, in this graph G_r, we have only self-loops, and all edges $e \in E_r$ are of the type (v, v). That is, for each vertex $v \in V$, there are no edges (v, u) to any other vertices $u \neq v \in V$. Much of the prior work on feature driven answer selection [2,17,32,31] adopts this view.

3.1.2 Contrastive

A second strategy is to examine answers *in relation* to other answers to the same question and label one such answer as 'accepted.' Thus the second strategy *contrasts* (q, a), with other tuples in $(q, a'), q \in \mathcal{Q}; a, a' \in \mathcal{A}_q; a' \neq a$. This is a **Contrastive** relation and the corresponding graph $G_c = (V, E_c)$ has a specific structure. Specifically, we define an edge $e \in E_c$ for all (q, a) tuples for the same question $q \in \mathcal{Q}$. That is, if $v = (q_1, a_1), u = (q_2, a_2)$, $e = (u, v) \in E_c \iff q_1 = q_2$. Intuitively, the contrastive relation induces cliques connecting all answers to the same question. Introducing contrasts between vertices sharpens differences between features, an effect (described in more detail in Section 4.2) we term *Discriminative Feature Magnification*. Notice that the contrastive relation is distinct from graphs with signed edges (e.g., [7]). In our framework, the contrast is a *neighborhood* property of a vertex, whereas in [7], the negative sign is a property of an *edge*.

3.1.3 Similarity by Contrast

A third strategy is to identify *similar* (q, a) tuples *across* questions. Prior work [35] indicates that individuals on StackExchange use diverse strategies to contribute answers. Experts (with a high reputation) tend to answer harder questions, while new members (with low reputation) looking to acquire reputation tend to be the first to answer a question.

How might similarity by contrast work? Consider two individuals Alice and Bob with *similar* reputations (either high or low) on StackExchange, who contribute answers a_A and

a_B to questions q_1 and q_2 respectively. If Alice and Bob have high reputation difference with other individuals who answer questions q_1 and q_2 respectively, then it is likely that (q_1, a_A) and (q_2, a_B) will share the same label (if they are both experts, their answers might be accepted, if they are both novices, then this is less likely). However, if Alice has a high reputation difference with other peers who answer q_1, *but Bob does not have that difference* with peers who answer q_2, then it is less likely that the tuples (q_1, a_A) and (q_2, a_B) will share the label, even though the reputations of Alice and Bob are similar.

Thus the key idea of the **Similarity by Contrast** relation is that link tuples that are *similar in how they differ* with other tuples. We construct the graph $G_s = (V, E_s)$ in the following manner. An edge $e = (v, u)$ between tuples v and u exists if the similarity $s(v, u)$ between tuples v, u exceeds a threshold δ. We define the similarity function $s(\cdot, \cdot)$ to encode similarity by contrast. That is, $e = (v, u) \in E_s \iff s(v, u) \geq \delta$.

Motivated by [35], we consider two different views that correspond to the similar contrast relation. The **TrueSkill Similarity** view connects all answers authored by a user where her skill (computed via Bayesian TrueSkill [16])) differs from competitors by margin δ. We capture both cases when the user is less or more skilled than her competitors. In the **Arrival Similarity** view, we connect answers across questions based on the similarity in the relative time of their arrival (posting timestamp). Notice that two Similarity by Contrast views have different edge (E) sets since the corresponding similarity functions are different. Notice also, that the two similarity function definitions are transitive. [4]

3.2 Generalized Views

Now we present the general case of the induced view. First, notice that each of the three relation types that we consider—reflexive, contrastive, and similarity—result in a graph $G_i = (V, E_i)$ comprising a set of cliques. This is not surprising, since all three relations presented here, are equivalence relations. Second, observe the semantics of how we select the tuple with the accepted answer. Within the three relations, we used two semantically different ways to assign the 'accepted' answer label to a tuple. One way is to share the labels amongst all the vertices in the *same clique* (used in the reflexive and the similarity relations). The second is to *assign label based on contrasts with other vertices* in the same clique. We can now state the organizing principle of our approach as follows.

> A generalized *modular* framework: pick a meaningful equivalence relation on the (q, a) tuples to induce graph comprising cliques and then apply specific label semantics (label sharing or label contrast) within each clique.

Equivalence relation results in a graph with a set of disconnected cliques. Cliques have some advantages: they have well-defined graph spectra [5, p. 6]; cliques allow for *exact* graph convolution; parallelize the training as the convolution of a clique is independent of other cliques.

[4] One trivial way of establishing similarity is co-authorship i.e. connect all (q, a) tuples of a user (probably on the same topic) across different questions. Note that the accepted answer is labeled relative to the other answers. As the competing answers are different in each question, we can not trivially assume acceptance label similarity for all co-authored answers. In our experiments, co-authorship introduced a lot of noisy links in the graph leading to worse performance.

4 Induced Relational GCN

Now, we will encode the two label assignment mechanisms within a clique via a graph convolution. First, we briefly review Graph Convolution Networks (GCN) and identify some key concepts. Then, given the views G_i for the four strategies, we show how to introduce label contrasts in Section 4.2 followed by label sharing in Section 4.3.

4.1 Graph Convolution

Graph Convolution models adapt the convolution operations on regular grids (like images) to irregular graph-structured data $G = (V, E)$, learning low-dimensional vertex representations. If for example, we associate a scalar with each vertex $v \in V$, where $|V| = N$, then we can describe the convolution operation on a graph by the product of signal $x \in \mathbb{R}^N$ (feature vectors) with a learned filter g_θ in the fourier domain. Thus,

$$g_\theta * x = U\, g_\theta\, U^T x, \tag{1}$$

where, Λ and U are the eigenvalues and eigenvector of the normalized graph Laplacian, $L = I_N - D^{-1/2} A D^{1/2}$, and where $L = U \Lambda U^T$. A denotes the adjacency matrix of a graph G (associated with a view) with N vertices. Equation (1) implies a filter g_θ with N free parameters, and requires expensive eigenvector decomposition of the adjacency matrix A. Deferrard et al. [6] proposed to approximate g_θ, which in general is a function of Λ, by a sum of Chebyshev polynomials $T_k(x)$ up to the k-th order. Then,

$$g_\theta * x \approx U \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda})\, U^T x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\, x, \tag{2}$$

where, $\tilde{\Lambda} = 2\Lambda/\lambda_{\max} - I_N$ are the scaled eigenvalues and $\tilde{L} = 2L/\lambda_{max} - I_N$ is the corresponding scaled Laplacian. Since $\tilde{L} = U\tilde{\Lambda}U^T$, the two equations are approximately equal.

The key result from Deferrard et al. [6] is that Equation (2) implies k-hop localization—the convolution result depends only on the k-hop neighborhood. In other words, Equation (2) is a k-hop approximation.

However, since we use equivalence relations in our framework that result in cliques, we can do an *exact* convolution operation since vertices in a clique only have one-hop (i.e., $k = 1$) neighbors (see lemma 5.2, [15]). The resulting convolution is linear in L and now has only two filter parameters, θ_0 and θ_1 shared over the whole graph.

$$g_\theta * x = \theta_0 x + \theta_1 (L - I_N)\, x \tag{3}$$

We emphasize the distinction with Kipf et al. [19] who approximate the Deferrard et al. [6] observation by restricting $k = 1$. They do so since they work on arbitrary graphs; since our relations result in views with cliques, we do not make any approximation by using $k = 1$.

4.2 Contrastive Graph Convolution

Now, we show how to perform graph convolution to encode the mechanism of contrast, where label assignments for a tuple depend on the contrast with its neighborhood.

To establish contrast, we need to compute the *difference* between the vertex's own features to its neighborhood in the clique. Thus we transform Equation (3) by setting $\theta = \theta_0$ = θ_1, which essentially restricts the filters learned by the GCN. This transformation leads to the following convolution operation:

$$g_\theta * x = \theta \left(I_N + L - I_N\right) x = \theta \left(I_N - D^{-1/2} A D^{-1/2}\right) x \tag{4}$$

Notice that Equation (4) says that for example, for any vertex u with a scalar feature value x_u, for a given clique with $n \geq 2$ vertices, the convolution operation computes a new value $\hat{x}_u$ for vertex u as follows:

$$\hat{x}_u = \theta \left(x_u - \frac{1}{n-1} \sum_{v \in \mathcal{N}_u} x_v \right). \tag{5}$$

where $\mathcal{N}_u$ is the neighborhood of vertex u. Notice that since our equivalence relations construct cliques, for all vertices u that belong to a clique of size n, $|\mathcal{N}_u| = n - 1$.

When we apply the convolution operation in Equation (4) at each layer of GCN, output for the k-th layer is:

$$\mathbf{Z}_c^k = \sigma\left(\left(I_N - D^{-1/2} A_c D^{1/2}\right) \mathbf{Z}_c^{k-1} \mathbf{W}_c^k\right) \tag{6}$$

with A_c denoting the adjacency matrix in the contrastive view. $\mathbf{Z}_c^k \in \mathbb{R}^{N \times d}$ are the learned vertex representations for each (q, a) tuple under the contrastive label assignment. N is the total number of tuples and d refers to the dimensionality of the embedding space. $\mathbf{Z}^{k-1}$ refers to the output of the previous $(k-1)$-th layer, and $\mathbf{Z}^0 = X$ where X is the input feature matrix. $\mathbf{W}_c^k$ are the filter θ parameters learnt by the GCN; $\sigma(\cdot)$ denotes the activation function (e.g. ReLU, $\tanh$).

To understand the effect of Equation (6) on a tuple, let us restrict our attention to a vertex u in a clique of size n. We can do this since the convolution result in one clique is unaffected by other cliques. When we do this, we obtain:

$$z_c^k(u) = \sigma\left(\left(z_c^{k-1}(u) - \frac{1}{n-1} \sum_{v \in \mathcal{N}_u} z_c^{k-1}(v)\right) \mathbf{W}_c^k\right). \tag{7}$$

Now consider a pair of contrasting vertices, u and v in the same clique of size n. Let us ignore the linear transform by setting $W_c^k = \mathbf{I}$ and set $\sigma(\cdot)$ to the identity function. Then we can easily verify that:

$$z_c^k(u) - z_c^k(v) = \underbrace{\left(1 + \frac{1}{n-1}\right)}_{\text{magnification}} \times \underbrace{\left(z_c^{k-1}(u) - z_c^{k-1}(v)\right)}_{\text{contrast in previous layer}}, \tag{8}$$

where, $z_c^k(u)$ denotes the output of the k-th convolution layer for the u-th vertex in the contrastive view. As a result, each convolutional layer magnifies the feature contrast between the vertices that belong to the same clique. Thus, the contrasting vertices move further apart. We term this as *Discriminative Feature Magnification* and Equation (8) implies that we should see higher magnification effect for smaller cliques.

4.3 Encoding Similarity Convolution

We next discuss how to encode the mechanism of sharing labels in a GCN. While label sharing applies to our similarity by contrast relation (two strategies: Arrival similarity; TrueSkill similarity, see Section 3.1), it is also trivially applicable to the reflexive relation, where the label of the tuple only depends on itself. First, we discuss the case of similarity by contrast.

4.3.1 Encoding Similarity by Contrast

To encode label sharing for the two similarity by contrast cases, we transform Equation (3) with the assumption $\theta = \theta_0 = -\theta_1$. Thus

$$g_\theta * x = \theta \left(I_N + D^{-1/2} A D^{-1/2} \right) x, \tag{9}$$

Similar to the Equation (4) analysis, convolution operation in Equation (9) computes a new value $\hat{x}_u$ for vertex u as follows:

$$\hat{x}_u = \theta \left(x_u + \frac{1}{n-1} \sum_{v \in \mathcal{N}_u} x_v \right) = \theta \left(\frac{n-2}{n-1} x_u + \frac{n}{n-1} \mu_x \right). \tag{10}$$

That is, in the mechanism where we share labels in a clique, the convolution pushes the values of each vertex in the clique to the average feature value, $\mu_x = \frac{1}{n} \sum_{v \in \mathcal{N}_u \cup u} x_v$, in the clique.

When we apply the convolution operation in Equation (9) at each layer of GCN, output for the k-th layer:

$$\mathbf{Z}_s^k = \sigma \left(\left(I_N + D^{-1/2} A_s D^{1/2} \right) \mathbf{Z}_s^{k-1} \mathbf{W}_s^k \right) \tag{11}$$

with A_s denoting the adjacency matrix in the similar views.

We analyze the similarity GCN in a maner akin to Equation (7) and we can easily verify that:

$$z_s^k(u) - z_s^k(v) = \underbrace{\left(1 - \frac{1}{n-1} \right)}_{\text{reduction}} \times \underbrace{\left(z_s^{k-1}(u) - z_s^{k-1}(v) \right)}_{\text{contrast in previous layer}}, \tag{12}$$

where, $z_s^k(i)$ denotes the output of the k-th convolution layer for the i-th vertex in the similar view. As a result, each convolutional layer reduces the feature contrast between the vertices that belong to the same clique. Thus, the similar vertices move closer.

The proposed label sharing encoding applies to both similarity by contrast strategies (TrueSkill; Arrival). We refer to the corresponding vertex representations as $\mathbf{Z}_{ts}^k$ (TrueSkill), $\mathbf{Z}_{as}^k$ (Arrival).

4.3.2 Reflexive Convolution

We encode the reflexive relation with self-loops in the graph resulting in an identity adjacency matrix. This relation is the trivial label sharing case, with an independent assignment of vertex labels. Thus, the output of the k-th convolutional layer for the reflexive view, $\mathbf{Z}_r^k$ reduces to:

$$\mathbf{Z}_r^k = \sigma \left(I_N \mathbf{Z}_r^{k-1} \mathbf{W}_r^k \right) \tag{13}$$

Hence, the reflexive convolution operation is equivalent to a feedforward neural network with multiple layers and activation $\sigma(\cdot)$.

Each strategy $S_i \in \mathbf{S}$ belongs to one of the three relation types—reflexive, contrastive and similarity, where $\mathbf{R}$ denotes the set of strategies of that relation type. $\mathcal{R} = \bigcup \mathbf{R}$ denotes the set of all relation types. $\mathbf{Z}_i^K \in \mathbb{R}^{NXd}$ represents the d dimensional vertex embeddings for strategy S_i at the K-th layer. For each strategy S_i, we obtain a scalar score by multiplying $\mathbf{Z}_i^K$ with transform parameters $\widetilde{W_i} \in \mathbb{R}^{d \times 1}$. The sum of these scores gives the combined prediction score, $\mathbf{H_R} \in \mathbb{R}^{NX1}$, for that relation type.

$$\mathbf{H_R} = \sum_{S_i \in \mathbf{R}} \mathbf{Z}_i^K \widetilde{W}_i^T \tag{14}$$

In this section, we proposed Graph Convolutional architectures to compute vertex representations of each (q, a) tuple under the four strategies. In particular, we showed how to encode two different label assignment mechanisms—label sharing and determine label based on contrast—within a clique. The architecture that encodes label assignment based on contrast is a novel contribution, distinct from the formulations presented by Kipf et al. [19] and its extensions [7,27]. Prior convolutional architectures implicitly encode the label sharing mechanism (eq. (9)); however, label sharing is unsuitable for contrastive relationships across vertices. Hence our architecture fills this gap in prior work.

5 Aggregating Induced Views

In the previous sections, we introduced four strategies to identify the accepted answer to a question. Each strategy induces a graph or relational view between (q, a) tuples. Each relational view is expected to capture semantically diverse neighborhoods of vertices. The convolution operator aggregates the neighborhood information under each view. The key question that follows is, *how do we combine these diverse views in a unified learning framework?* Past work has considered multiple solutions:

- **Neighborhood Aggregation**: In this approach, we represent vertices by aggregating feature representations of it's neighbors across all views [14,27].
- **Stacking**: Multiple convolution layers stacked end-to-end (each potentially handling a different view) [37].
- **Fusion**: Follows a multi-modal fusion approach [9], where views are considered distinct data modalities.
- **Shared Latent Structure**: Attempts to transfer knowledge across relational views (modalities) with constraints on the representations (e.g. [39] aligns embeddings across views).

Ensemble methods introduced in [27] work on multi-relational edges in knowledge graphs. None of these approaches are directly suitable for our induced relationships. Our relational views utilize different label assignment semantics (label sharing within a clique vs. determine label based on contrast within a clique). In our label contrast semantics, we must achieve feature discrimination and label inversion between contrasting vertices, as opposed to label homogeneity and feature sharing in the label sharing case. Thus, aggregating relationships by pooling, concatenation, or addition of vertex representations fail to capture semantic heterogeneity of the induced views. Further, data induced relations are uncurated and inherently noisy. Directly aggregating the learned representations via Stacking or Fusion can lead to noise propagation. We also expect views of the same relation type to be correlated.

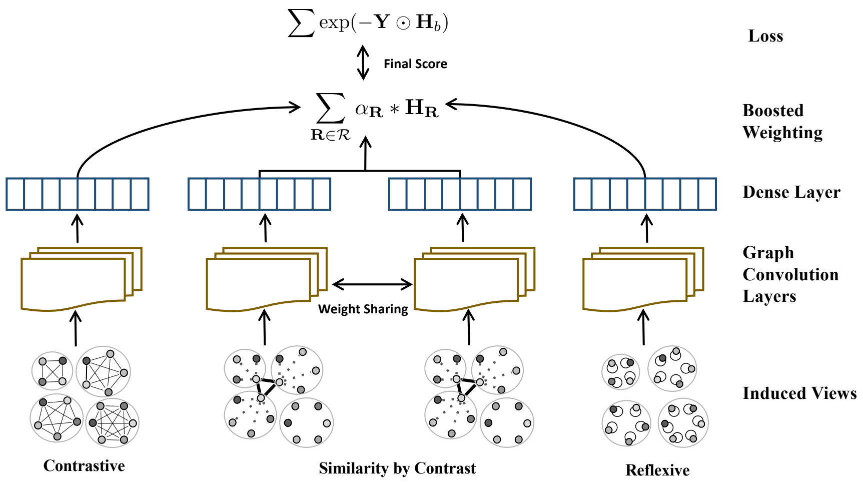


Fig. 2: Schematic diagram of our proposed IR-GCN model.

We thus propose the following approach to aggregate information across relation types and between views of a relation type.

Cross-relation Aggregation: We expect distinct relation types to perform well on different subsets of the set of (q, a) tuples. We empirically verify this with the Jaccard overlap between the set of misclassified vertices under each relational view of a relation type on our dataset. Given $\mathbf{M}_A$ and $\mathbf{M}_B$, the sets of (q, a) tuples misclassified by GCNs A and B respectively, the jaccard overlap is,

$$\mathcal{J}_{A,B} = \frac{\mathbf{M}_A \cap \mathbf{M}_B}{\mathbf{M}_A \cup \mathbf{M}_B}$$

The $\mathcal{J}_{A,B}$ values are as follows for the relational pairings: (Contrastive, TrueSkill Similarity) = 0.42, (Contrastive, Reflexive) = 0.44 and (Reflexive, TrueSkill Similarity) = 0.48. Relatively low values of the overlap metric indicate uncorrelated errors across the relations.

Gradient boosting techniques are known to improve performance when individual classifiers, including neural networks [28], are diverse yet accurate. A natural solution then is to apply boosting to the set of relation types and bridge the weaknesses of each learner. We employ Adaboost [12] to combine relation level scores, $\mathbf{H}_{\mathbf{R}}$ (eq. (14)) in a weighted manner to compute the final boosted score, $\mathbf{H}_b \in \mathbb{R}^{N \times 1}$ representing all relation types (Line 12, algorithm 1). $\mathbf{Y} \in \mathbb{R}^{NX1}$ denotes the acceptance label of all tuples. Note that an entry in $(\mathbf{Y} \odot \mathbf{H}_{\mathbf{R}}) > 0$ when the accepted label of the corresponding (q, a) tuple and sign of the prediction score, $sign(\mathbf{H}_{\mathbf{R}})$, of relation type $\mathbf{R}$ match and < 0 otherwise. Thus, the weights $\alpha_{\mathbf{R}}$ adapt to the fraction of correctly classified tuples to the misclassified tuples by the relation $\mathbf{R}$ (Line 9, algorithm 1). The precise score computation is described in algorithm 1. We use the polarity of each entry in the boosted score, $sign(\mathbf{H}_b) \in \{-1, 1\}$, to predict the class label of the corresponding (q, a) tuple. The final score is also used to create a ranked list among all the candidate answers, $a \in \mathcal{A}(q)$ for each question, $q \in \mathcal{Q}$. $L_{(q,a)}$ represents the position of candidate answer a in the ranked list for question q.

Intra-relation Aggregation: Gradient boosting methods can effectively aggregate relation level representations, but are not optimal within a relationship type (since it cannot capture shared commonalities between different views of a relation type). For instance, we should

Algorithm 1 IR-GCN Boosted Score Computation

```
function FORWARD($\mathbf{X}, \mathbf{Y}, \{A_i\}_{S_i \in \mathbf{S}}$)
    $\mathbf{H}_b \leftarrow \mathbf{0}$
    for $\mathbf{R} \in \mathcal{R}$ do
        $\{\mathbf{Z}_i^K\}_{S_i \in \mathbf{R}} \leftarrow Conv(\mathbf{X}, \{A_i\}_{S_i \in \mathbf{R}})$
                                                        ▷ Equation 6, 11, 13
        $\mathbf{H}_\mathbf{R} = \sum_{S_i \in \mathbf{R}} \mathbf{Z}_i^K \times \widetilde{\mathbf{W}}_i$      ▷ Equation 14
        $\mathbf{e}_\mathbf{R} \leftarrow \exp(-\mathbf{Y} \odot \mathbf{H}_b)$
                                                        ▷ $\odot \rightarrow$ *Hadamard Product*
        $\alpha_\mathbf{R} \leftarrow \frac{1}{2} \ln \frac{\sum \mathbf{e}_\mathbf{R} \odot \mathbb{1}((\mathbf{Y} \odot \mathbf{H}_\mathbf{R}) > 0)}{\sum \mathbf{e}_\mathbf{R} \odot \mathbb{1}((\mathbf{Y} \odot \mathbf{H}_\mathbf{R}) < 0)}$
                                                        ▷ $\sum \rightarrow$ *reduce-sum*
                                                        ▷ $\mathbb{1}(.) \rightarrow$ *element-wise Indicator function*
        $\mathbf{H}_b \leftarrow \mathbf{H}_b + \alpha_\mathbf{R} * \mathbf{H}_\mathbf{R}$      ▷ Update boosted GCN
    end for
    return $\mathbf{H}_b$, $\{\mathbf{H}_R\}_{\mathbf{R} \in \mathcal{R}}$, $\{\mathbf{Z}_i^K\}_{S_i \in \mathbf{S}}$
                                                        ▷ Boosted scores, Relation level scores,
                                                        ▷ Each GCN vertex representations
end function
```

facilitate information sharing between the TrueSkill similarity and Arrival similarity views. Thus, if an answer is authored by a user with a higher skill rating and answered significantly earlier than other answers, its probability to be accepted should be mutually enhanced by both signals. Empirically, we also found True Skill and Arrival Similarity GCNs to commit similar mistakes ($\mathcal{J}_{TS,AS}$ = 0.66). Thus, intra-relation learning (within a single relation type like Similarity by Contrast) can benefit from sharing the structure of their latent spaces, i.e., weight parameters of GCN.

Weight Sharing: For multiple views representing a relation type (e.g., TrueSkill and Arrival Similarity), we train a separate GCN for each view but share the layer-wise linear-transforms $\mathbf{W}_i^k$ to capture similarities in the learned latent spaces. Weight sharing is motivated by a similar idea explored to capture local and global views in [39]. Although sharing the same weight parameters, each GCN can still learn distinct vertex representations as each view convolves over a different neighborhood and employ random dropout during training. We thus propose to use an alignment loss term to minimize prediction difference between views of a single relation type[24]. The loss attempts to align the learned vertex representations at the *last layer* K (the loss term aligns pairs of final vertex representations, $||\mathbf{Z}_i^K - \mathbf{Z}_{i'}^K|| \;\forall\; S_i, S_i' \in \mathbf{R}$). In principle, multiple GCNs augment performance of the relation type by sharing prior knowledge through multiple Adjacency matrices ($\mathbf{A}_i \;\forall\; S_i \in \mathbf{R}$).

Training Algorithm: Algorithm 2 describes the training algorithm for our IR-GCN model. For each epoch, we first compute the aggregated prediction score $\mathbf{H}_b$ of our boosted model, as described in algorithm 1. We use a supervised exponential loss $\mathcal{L}_b$ for training with elastic-net regularization (L1 loss - $\mathcal{L}_1(.)$ and L2 loss - $\mathcal{L}_2(.)$) on the graph convolutional weight matrices $\mathbf{W}_\mathbf{i}^k \;\forall\; S_i \in \mathbf{S}$ for each view. Note that we employ weight sharing between all views of the same relation type so that only one set of weight matrices is learned per relation. The exponential loss, $\mathcal{L}_\mathbf{R}$, for each relation type is added alternatingly to the boosted loss. We apply an *exponential annealing schedule*, $\lambda(t)$, i.e. a function of the training epochs (t), to the loss function of each relation. As training progress and the boosted model learns to distribute vertices among the relations optimally, an increase in $\lambda(t)$ en-

Algorithm 2 IR-GCN Training

```
Input: Input Feature Matrix $X$, Acceptance labels for each tuple, $\mathbf{Y}$, Adjacency matrix of each view $\{A_i\}_{S_i \in \mathbf{S}}$
Output: Trained Model i.e. Weight parameters $W_i^1 \dots W_i^k, S_i \in \mathbf{S}, \forall k \in [1, K]$ and transform parameters $\widetilde{W}_i, S_i \in \mathbf{S}$
for $t \leftarrow 1$ to num-epochs do
    $\mathbf{H}_b, \{\mathbf{H}_R\}_{\mathbf{R} \in \mathcal{R}}, \{\mathbf{Z}_i^K\}_{S_i \in \mathbf{S}} \leftarrow$ FORWARD$(X, Y, \{A_i\}_{S_i \in \mathbf{S}})$
                                                                ▷ Algorithm 1
    for $\mathbf{R} \in \mathcal{R}$ do
        $\mathcal{L}_b \leftarrow \sum \exp(-\mathbf{Y} \odot \mathbf{H}_b) + \gamma_1 \mathcal{L}_1(.) + \gamma_2 \mathcal{L}_2(.)$
                                                                ▷ $\sum \rightarrow$ reduce-sum
                                                                ▷ $\odot \rightarrow$ Hadamard Product
        $\mathcal{L}_\mathbf{R} \leftarrow 0$
        for $S_i \in \mathbf{R}$ do
            $\mathcal{L}_i \leftarrow \sum \exp(-\mathbf{Y} \odot \mathbf{H}_\mathbf{R})$
            $\mathcal{L}_\mathbf{R} \leftarrow \mathcal{L}_\mathbf{R} + \mathcal{L}_i + \frac{1}{2} \sum_{S'_i \neq S_i} ||\mathbf{Z}_i^K - \mathbf{Z}_{i'}^K||$
        end for
        $\mathcal{L}_b \leftarrow \mathcal{L}_b + \lambda(t) \mathcal{L}_\mathbf{R}$
        $W_i^k \leftarrow W_i^k + \eta_{\text{ADAM}} \frac{\partial \mathcal{L}_b}{\partial W_i^k}$     ▷ $\forall k \in [1, K], \forall S_i \in \mathbf{R}$
        $\widetilde{W}_i \leftarrow \widetilde{W}_i + \eta_{\text{ADAM}} \frac{\partial \mathcal{L}_b}{\partial \widetilde{W}_i}$     ▷ $\forall S_i \in \mathbf{S}$
    end for
end for
```

sures more emphasis is provided to the individual convolutional networks of each relation. Figure 2 illustrates the overall architecture of our IR-GCN model.

6 Experiments

In this section, we first describe our dataset, followed by our experimental setup; comparative baselines, evaluation metrics, and implementation details. We then present results across several experiments to evaluate the performance of our model on merging semantically diverse induced-relations.

	Technology			**Culture/Recreation**			**Life/Arts**			**Science**			**Professional/Business**		
	ServerFault	AskUbuntu	Unix	English	Games	Travel	SciFi	Home	Academia	Physics	Maths	Statistics	Workplace	Aviation	Writing
$\|Q\|$	61,873	41,192	9,207	30,616	12,946	6,782	14,974	8,022	6,442	23,932	18,464	13,773	8,118	4,663	2,932
$\|\mathcal{A}\|$	181,974	119,248	33,980	110,235	45,243	20,766	49,651	23,956	23,837	65,800	53,772	36,022	33,220	14,137	12,009
$\|U\|$	140,676	200,208	84,026	74,592	14,038	23,304	33,754	30,698	19,088	52,505	28,181	54,581	19,713	7,519	6,918
$\mu(\|\mathcal{A}_q\|)$	2.94	2.89	3.69	3.6	3.49	3.06	3.31	2.99	3.7	2.75	2.91	2.62	4.09	3.03	4.10

Table 1: Dataset statistics for the top three Stack Exchange communities from five different categories. $|Q|$: number of questions; $|\mathcal{A}|$: number of answers; $|U|$: number of users; $\mu(|\mathcal{A}_q|)$: mean number of answers per question. Professional/Business communities have slightly more answers per question on average than others. Technology communities are the largest in terms of number of question out of the five categories.

6.1 Dataset

We evaluate our approach on multiple communities catering to different topics from a popular online Community Question Answer (CQA) platform, *StackExchange*[5]. The platform divides the communities into five different categories, i.e. Technology ($\mathbf{T}$), Culture/Recreation ($\mathbf{C}$), Life/Arts ($\mathbf{L}$), Science ($\mathbf{S}$) and Professional ($\mathbf{P}$). For our analysis, we collect data from the ten largest communities from each of the five categories until March 2019, resulting in a total of 50 StackExchange communities. In StackExchange, each questioner can mark a candidate answer as an ”accepted” answer. We only consider questions with an accepted answer. Table 1 shows the final dataset statistics.

For each (q, a) tuple, we compute the following basic features:
Activity features : View count of the question, number of comments for both question and answer, the difference between posting time of question and answer, arrival rank of answer (we assign rank 1 to the first posted answer) [32].
Text features : Paragraph and word count of question and answer body and question title, presence of code snippet in question and answer (useful for programming based forums)
User features : Word count in user profile’s Aboutme section for both users; one posting the question and other posting the answer.

Time-dependent features like upvotes/downvotes of the answer and user features like reputation or badges used in earlier studies on StackExchange [2] are problematic for two reasons. First, we only know the aggregate values, not how these values change with time. Second, since these values typically increase over time, it is unclear if an accepted answer received the votes *prior* to or *after* an answer was accepted. Thus, we do not use such time-dependent features for our model and the baselines in our experiments.

6.2 Experimental Setup

6.2.1 Baselines

We compare against state-of-the-art feature-based baselines for answer selection and competing aggregation approaches to fuse diverse relational views of the dataset [39,27].
Random Forest (RF) [2,32] model trains on the feature set mentioned earlier for each dataset. This model is shown to be the most effective feature-based model for Answer Selection.
Feed-Forward network (FF) [17] is used as a deep learning baseline to learn non-linear transformations of the feature vectors for each (q, a) tuple. This model is equivalent to our Reflexive GCN model in isolation.
Dual GCN (DGCN) [39] trains a separate GCN for each view. In addition to the supervised loss computed using training labels, they introduce a regularizer to minimize mean squared error (MSE) between vertex representations of two views, thus aligning the learned latent spaces. The regularizer loss is similar to our intra-relation aggregation approach but assumes label and feature sharing across *all* the views.
Relational GCN (RGCN) [27] combines the output representations of previous layer of each view to compute an aggregated input to the current layer.

We also report results for each view individually: Contrastive (C-GCN), Arrival Similarity (AS-GCN), TrueSkill Similarity (TS-GCN), and Reflexive (R-GCN) with our proposed

[5] https://stackexchange.com/

IR-GCN model. We do not compare with other structure-based approaches to compute vertex representations [23,13,30] as GCN is shown to outperform them [19]. We also compare with common aggregation strategies to merge neural representations discussed earlier in section 5 later.

6.2.2 Evaluation Metric

We randomly select 20% of the questions, $\mathbf{T}_q \subset \mathcal{Q}$ to be in the test set. Then, subsequently all (q,a) tuples such that $q \in \mathbf{T}_q$ comprise the set of test tuples or vertices, $\mathbf{T}$. The rest of the vertices, along with their label information, is used for training the model. We evaluate our model on two metrics, Accuracy and Mean Reciprocal Rank (MRR). Accuracy metric is widely used in vertex classification literature while MRR is popular for ranking problems like answer selection. Formally,

$$Acc = \frac{1}{|\mathbf{T}|} \sum_{(q,a) \in \mathbf{T}} \mathbb{1}\left(y_{(q,a)} \cdot h_b((q,a)) > 0\right)$$

with $\cdot$ as the product and $\mathbb{1}$ as the indicator function. The product is positive if the accepted label and predicted label match and negative otherwise.

$$MRR = \frac{1}{|\mathbf{T}_q|} \sum_{q \in \mathbf{T}_q} \frac{1}{\sum_{a' \in \mathcal{A}(q)} \mathbb{1}\left(L_{(q,a)} < L_{(q,a')}\right)}$$

where $L_{(q,a)}$ is the position of accepted answer a in the ranked list for question q [34].

Method	Technology		Culture/Recreation		Life/Arts		Science		Professional/Business	
	Acc(%)	MRR	Acc(%)	MRR	Acc(%)	MRR	Acc(%)	MRR	Acc(%)	MRR
RF [2,32]	66.78±0.023	0.683±0.043	72.50±0.018	0.626±0.050	72.71±0.049	0.628±0.089	68.09±0.024	0.692±0.049	74.72±0.044	0.595±0.081
FF [17]	67.31±0.027	0.786±0.022	72.22±0.020	0.782±0.023*	73.58±0.049	0.780±0.034	67.87±0.024	0.800±0.028	74.63±0.040	0.760±0.049
DGCN [39]	70.70±0.022	0.782±0.017	75.22±0.017	0.772±0.028	76.73±0.034	0.784±0.038	71.45±0.023*	0.792±0.035	76.86±0.031	0.751±0.046
RGCN [27]	54.40±0.045	0.673±0.045	60.39±0.016	0.646±0.042	59.97±0.043	0.655±0.054	58.65±0.054	0.683±0.042	63.02±0.038	0.657±0.061
AS-GCN	67.76±0.032	0.775±0.015	73.05±0.021	0.763±0.025	73.79±0.048	0.777±0.042	66.93±0.045	0.788±0.028	74.99±0.045	0.742±0.047
TS-GCN	66.87±0.032	0.779±0.018	72.16±0.023	0.764±0.023	72.02±0.061	0.766±0.048	65.90±0.042	0.790±0.031	74.17±0.046	0.747±0.044
C-GCN	71.64±0.022*	0.790±0.015*	76.18±0.017*	0.781±0.024	77.37±0.034*	0.788±0.040*	70.81±0.042	0.800±0.032*	77.57±0.038*	0.768±0.034*
IR-GCN	**73.96±0.023**	**0.794±0.014**	**78.61±0.018**	**0.791±0.025**	**79.21±0.032**	**0.800±0.037**	**74.98±0.021**	**0.809±0.028**	**80.17±0.026**	**0.785±0.032**

[*] DGCN stands for DualGCN, RGCN stands for RelationalGCN, and IR-GCN stands for Induced Relational GCN.

Table 2: Accuracy and MRR values for StackExchange with state-of-the-art baselines. Our model outperforms by at least 4% in Accuracy and 2.5% in MRR. Contrastive GCN performs best among individual views. The model with $*$ symbol has the second-best performance among all other models. Our model shows statistical significance at level 0.01 overall second best model on single tail paired t-test.

6.2.3 Implementation Details

We implemented our model and the baselines in Pytorch. We use ADAM optimizer [18] for training with 50% dropout to avoid overfitting. We use four hidden layers in each GCN with hidden dimensions 50, 10, 10, 5, respectively, and ReLU activation. The coefficients of $\mathcal{L}_1$ and $\mathcal{L}_2$ regularizers are set to $\gamma_1 = 0.05$ and $\gamma_2 = 0.01$ respectively. For TrueSkill Similarity, we use margin $\delta = 4$ to create links, while for Arrival similarity, we use $\delta = 0.95$. We implement a mini-batch training for large graphs where each batch contains a set of questions and their associated answers. This is equivalent to training on the whole graph as we have disconnected cliques. All code and data will be released upon publication.

6.3 Performance Analysis

Table 2 shows impressive gains over state-of-the-art baselines for all five categories. We report mean results for each category obtained after 5-fold cross-validation on each of the communities. Our induced-relational GCN model beats best performing baseline by 4-5% on average in accuracy. The improvement in MRR values is around 2.5-3% across all categories. Note that MRR is based only on the rank of the accepted answer, while accuracy is based on correct labeling of *both* accepted and non-accepted answers.

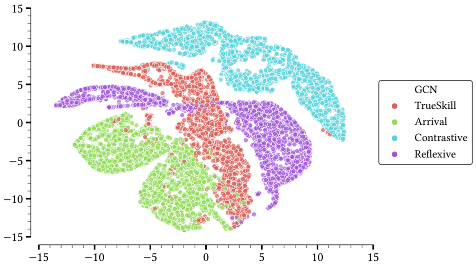


Fig. 3: t-stochastic neighbor embedding (t-SNE) [22] distributions of the learned vertex representations by our model for Chemistry StackExchange. Each view learns a distinct vertex representation. Best viewed in color.

Among individual views, Contrastive GCN performs best on all the communities. It even beats the best performing baseline DualGCN that uses all the relational views. Note that the contrastive view compares between the candidate answers to a question and uses our proposed contrastive modification to the convolution operation. Arrival Similarity follows Contrastive and then Reflexive. The superior performance of the Arrival Similarity view shows that early answers tend to get accepted and vice versa. It indicates that users primarily use CQA forums for quick answers to their queries. Also, recall that Reflexive predicts each vertex's label independent of other answers to the same question. Thus, the competitive performance of the Reflexive strategy indicates that vertex's features itself are well predictive of the label. TrueSkill Similarity performs at par or slightly worse than Reflexive. Figure 3 presents t-SNE distributions [22] of the learned vertex representations ($\mathbf{Z}_i^K$) of our model applied to Chemistry StackExchange from Science category. Note that each view, including two views under Similarity by Contrast relation, learns a distinct vertex representation. Hence, all views are essential and contribute to our final performance.

Out of the baseline graph ensemble approaches, DualGCN performs significantly better than RelationalGCN by an average of around 26% for all categories. Recall that in the RelationalGCN model, the convolution output of each view is linearly combined to compute the final output. Linear combination works well for knowledge graphs as each view can be thought of as a feature, and then it accumulates information from each feature. DualGCN is similar to our approach and trains different GCN for each view and later merges their results. However, it enforces similarity in vertex representations learned by each view. This restriction is not suitable for our induced-relationships as they are semantically different (contrastive captures contrast in features vs. similarity enforces label sharing).

6.4 Ablation Study on Relation Types

We present the results of an ablation study with a different combination of relation types (Contrastive, Similarity, and Reflexive) used for the IR-GCN model in Table 3. We conducted this study on the biggest community from each of the five categories, i.e., ServerFault (Technology), English (Culture), Science Fiction (Life), Physics (Science), Workplace (Business). Similarity by Contrast relation (TrueSkill and Arrival) used in isolation performs the worst among all the variants. Training Contrastive and Similarity by Contrast relation

{ **Relation Type**}	**Tech**	**Culture**	**Life**	**Sci**	**Business**
C	71.23	75.90	78.71	72.99	76.85
{ TS, AS }	67.86	74.15	75.75	65.80	76.13
R	68.30	73.35	76.57	67.40	75.76
{TS, AS } + R	69.28	75.50	76.41	70.11	77.90
C + R	73.04	77.66	80.25	73.72	80.04
C + { TS, AS }	72.81	78.04	81.41	72.19	80.15
C + { TS, AS } + R	**73.87**	**78.74**	**81.60**	**74.68**	**80.56**

Table 3: 5-fold Accuracy (in %) comparison for different combination of relation types for our boosted model. Contrastive and Similarity by Contrast relations together performs similar to the final model.

together in our boosted framework performs similar to our final model. Reflexive GCN contributes the least as it does not consider any neighbors.

6.5 Aggregator Architecture Variants

We compare our gradient boosting based aggregation approach with other popular methods used in literature to merge different neural networks discussed in section 5.

Method	**Tech**	**Culture**	**Life**	**Sci**	**Business**
Stacking [37]	68.58	74.44	79.19	70.29	75.50
Fusion [9]	72.30	77.25	80.79	73.91	79.01
NeighborAgg [14,27]	69.29	74.28	77.94	68.42	78.64
IR-GCN	**73.87**	**78.74**	**81.60**	**74.78**	**80.56**

Table 4: 5-fold Accuracy (in %) comparison of different aggregator architectures. These architectures perform worse than Contrastive GCN. Fusion performs similarly but is computationally expensive.

Table 4 reports the accuracy results for these aggregator variants as compared to our model. Our method outperforms all the variants with Fusion performing the best. This worse performance reaffirms that existing aggregation models are not suitable for our problem. Note that these approaches perform worse than even Contrastive GCN except Fusion. The fusion approach performs similarly to our approach but is computationally expensive as the input size for each IR-GCN in fusion is linear in the number of all views in the model.

6.6 Textual Features

Most of the current literature focuses on using textual features for Answer Selection. In this section, we compare our proposed IR-GCN model to a popular text-based model [29].

QA-LSTM/CNN [29] uses a stacked bidirectional LSTM model followed by convolution filters to learn embeddings for the question and answer text separately. They then rank answers in decreasing order of the cosine similarity between question and answer embeddings.

Textual Similarity (T-GCN) We create a *Similarity by Contrast* view that connects answers authored by a user where her answer's text is significantly similar (dissimilar) to the question's text than the other competing answers. We used cosine similarity on the learned question and answer embeddings from the QA-LSTM approach as the similarity function.

IR-GCN + T-GCN extends our proposed model to also include the Textual Similarity as the third *Similarity by Contrast* view in addition to Arrival and TrueSkill views.

Method	**Tech**	**Culture**	**Life**	**Sci**	**Business**
QA-LSTM/CNN[29]	66.49	71.70	69.42	62.91	72.55
FF [17]	68.30	73.35	76.57	67.40	75.76
T-GCN	69.25	73.77	76.39	67.79	77.08
IR-GCN	73.87	78.74	81.60	74.68	80.56
IR-GCN + T-GCN	73.89	78.00	81.07	74.49	78.86

Table 5: 5-fold Accuracy comparison of text-based baseline and textual similarity GCN with IR-GCN.

In general, the text-based baseline, QA-LSTM, performs worse than even reflexive GCN, as shown in Table 5. Note that reflexive GCN employs a feedforward model on the activity and user features used in our experiments. This is a surprising result as most of the current literature focus on textual features for the task. Our results indicate that non-textual features are useful too for the answer selection task on StackExchange communities.

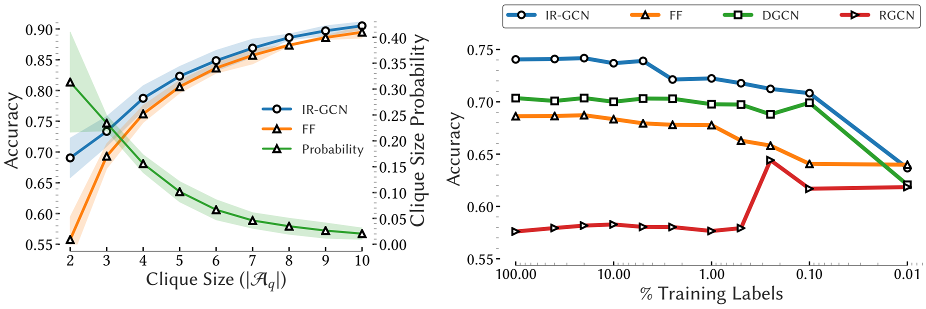
Textual Similarity GCN performs better than QA-LSTM and Reflexive GCN. Even though we use the output of QA-LSTM to construct the graph for T-GCN, the graph improves performance as it connects answers across different questions. However, adding the T-GCN view in our proposed IR-GCN model decreases the performance slightly. One possible explanation could be that similarity by contrast views based on user features (Arrival similarity and TrueSkill similarity) are not compatible with views based on textual features.

6.7 Discriminative Magnification effect

We show that due to our proposed modification to the convolution operation for contrastive view, we achieve *Discriminative Magnification effect* (eq. (6)). Note that the difference is scaled by Clique size $(1 + 1/n - 1)$, i.e. number of answers to a question, $|\mathcal{A}_q|$. Figure 4a shows the accuracy of our IR-GCN model as compared to the FeedForward model with varying clique size. Recall that the FeedForward model predict node labels independent of other nodes and is not affected by clique size. We report average results over the same five communities as above. We can observe that increase in accuracy is much more for lower clique sizes (13% improvement for $|\mathcal{A}_q| = 2$ and 4% for $|\mathcal{A}_q| = 3$ on average). The results are almost similar for larger clique sizes. In other words, our model significantly outperforms the FeedForward model for questions with fewer candidate answers. However, around 80% of the questions have very few answers(< 4), and thus this gain over FF is significant.

6.8 Label Sparsity

Graph Convolution Networks are robust to label sparsity as they exploit graph structure and are thus heavily used for semi-supervised settings. Figure 4b shows the change in accuracy for Physics StackExchange from the Science category at different training label rates. Even though our graph contains disconnected cliques, IR-GCN still preserves robustness to label sparsity. In contrast, the accuracy of the FeedForward model declines sharply with less label information. Performance of DualGCN remains relatively stable while Relational GCN's performance increases with a decrease in label rate. Relational GCN assumes each view to be of similarity relation, and thus, adding contrastive relation introduces noise in the model. However, as the training labels become extremely sparse, the training noise decreases that leads to a marked improvement in the model. In the case of a meager label rate of 0.01%, all approaches converge to the same value, which is the expectation of theoretically random selection. We obtained similar results for the other four StackExchange communities but omitted them for brevity.



(a)

(b)

Fig. 4: a) Accuracy of our IR-GCN model compared to the FF model with varying clique size (i.e., number of answers to a question, $|\mathcal{A}_q|$) for Contrastive view. We report averaged results over the largest community of all categories. Our model performs much better for smaller cliques, and the effect diminishes for larger cliques (eq. (6)). 80% of the questions have < 4 answers.
b) Change in accuracy with varying training label rates for Physics StackExchange. Our model is more robust to label sparsity than other relation ensemble approaches. RGCN works better with fewer labels as contrastive relation introduces noise in the model. At extreme sparsity, all approaches converge to the same value indicating random selection.

6.9 Limitations

We do recognize certain limitations of our work. First, we focus on equivalence relations that induce a graph comprising cliques. While cliques are useful graph objects for answer selection, equivalence relations may be too restrictive for other problems (e.g., the relation is not transitive). However, our modular framework does apply to arbitrary graphs, except that Equation (3) will no longer be an *exact* convolution but be an approximation. Second, we assume no evolution in author skills. This assumption is not true as users evolve with experience. We aim to address this in future work.

7 Related Work

Our work intersects two research areas; Answer Selection and handling multi-relational social data, primarily via Graph Convolution.

Answer Selection In CQA forums, previous answer selection literature includes feature-driven models and deep text models.

Feature-Driven Models in CQA identify and incorporate user features, content features, and thread features, e.g., in tree-based models to identify the best answer. Tian et al. [32] found that the best answer tends to be early and novel, with more details and comments. Jenders et al. [17] trained classifiers for online forums, Burel et al. [2] emphasize the Q&A thread structure.

Deep Text Models learn optimal QA text-pair representations to select the best answer [38, 36,33]. Feng et al. [10] augment CNNs with discontinuous convolution for improved representations; Wang et al. [29] use stacked biLSTMs to match question-answer semantics.

Graph Convolution is applied in spatial and spectral domains to compute graph node representations for downstream tasks including node classification [19], link prediction [27], multi-relational tasks [25] etc. Spatial approaches employ random walks or k-hop neighborhoods to compute node representations [23,13,30] while fast localized convolutions are applied in the spectral domain[6,8]. Our work is inspired by Graph Convolution Networks (GCN) [19], which outperforms spatial convolutions and scales to large graphs. GCN extensions have been proposed for signed networks [7], inductive settings [14], multiple relations [39,27] and diffusion [26]. However, GCN variants assume label sharing, which cannot model contrastive relations in our setting.

Multi-Relational Modeling: While text and feature models treat answer content independently, we focus on integrating multi-relational aspects in the prediction. We identify a few related threads; adversarial approaches to integrate social neighbor data [20,21]; meta-learning to adapt across data modalities or tasks [11]. Different from these directions, we focus on the flexibility and simplicity of our multi-relational graph formulation for modeling user-generated content.

8 Conclusion

This paper addressed the question of identifying the accepted answer to a question in CQA forums. We developed a novel induced relational graph convolutional (IR-GCN) framework to address this question. We made three contributions. First, we introduced a novel idea of using strategies to induce different views on (q, a) tuples in CQA forums. Each view consists of cliques and encodes—reflexive, similar, contrastive—relation types. Second, we encoded label sharing and label contrast mechanisms within each clique through a GCN architecture. Our novel contrastive architecture achieves *Discriminative Magnification* between nodes. Finally, we show through extensive empirical results on StackExchange that boosting techniques improved learning in our convolutional model. Our ablation studies show that the contrastive relation is most effective individually in StackExchange.

References

1. Brugere, I., Gallagher, B., Berger-Wolf, T.Y.: Network structure inference, a survey: Motivations, methods, and applications. ACM Comput. Surv. (2018). https://doi.org/10.1145/3154524
2. Burel, G., Mulholland, P., Alani, H.: Structural normalisation methods for improving best answer identification in question answering communities. In: International Conference on World Wide Web, WWW (2016)
3. Burges, C.J., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: Advances in Neural Information Processing Systems. MIT Press (2007)
4. Burges, C.J.C.: From RankNet to LambdaRank to LambdaMART: An overview. Tech. rep., Microsoft Research (2010)
5. Chung, F.R.K.: Spectral graph theory, CBMS Regional Conference Series in Mathematics, vol. 92. American Mathematical Society (1997)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems (2016)
7. Derr, T., Ma, Y., Tang, J.: Signed graph convolutional network. CoRR **abs/1808.06354** (2018)
8. Duvenaud, D.K., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in Neural Information Processing Systems (2015)
9. Farnadi, G., Tang, J., De Cock, M., Moens, M.F.: User profiling through deep multimodal fusion. In: International Conference on Web Search and Data Mining. WSDM '18, ACM (2018)
10. Feng, M., Xiang, B., Glass, M.R., Wang, L., Zhou, B.: Applying deep learning to answer selection: A study and an open task. In: IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU (2015)
11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1126–1135. JMLR. org (2017)

12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. Springer-Verlag (1995)
13. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: International Conference on Knowledge Discovery and Data Mining (2016)
14. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems (2017)
15. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis **30**(2), 129–150 (2011)
16. Herbrich, R., Minka, T., Graepel, T.: Trueskill™: A bayesian skill rating system. In: International Conference on Neural Information Processing Systems (2006)
17. Jenders, M., Krestel, R., Naumann, F.: Which answer is best?: Predicting accepted answers in MOOC forums. In: International Conference on World Wide Web (2016)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016)
20. Krishnan, A., Cheruvu, H., Tao, C., Sundaram, H.: A modular adversarial approach to social recommendation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1753–1762. ACM (2019)
21. Krishnan, A., Sharma, A., Sankar, A., Sundaram, H.: An adversarial approach to improve long-tail performance in neural collaborative filtering. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1491–1494. ACM (2018)
22. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research (2008)
23. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: International Conference on Knowledge Discovery and Data Mining (2014)
24. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: International Conference on Neural Information Processing Systems. NIPS'16 (2016)
25. Sankar, A., Krishnan, A., He, Z., Yang, C.: Rase: Relationship aware social embedding. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
26. Sankar, A., Zhang, X., Krishnan, A., Han, J.: Inf-vae: A variational autoencoder framework to integrate homophily and influence in diffusion prediction. In: Proceedings of the 13th International Conference on Web Search and Data Mining. p. 510–518. WSDM '20 (2020). https://doi.org/10.1145/3336191.3371811
27. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. Springer (2018)
28. Schwenk, H., Bengio, Y.: Boosting neural networks. Neural Computation **12**(8), 1869–1887 (2000)
29. Tan, M., Xiang, B., Zhou, B.: Lstm-based deep learning models for non-factoid answer selection. CoRR **abs/1511.04108** (2015)
30. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: International Conference on World Wide Web, WWW (2015)
31. Tian, Q., Li, B.: Weakly hierarchical lasso based learning to rank in best answer prediction. In: International Conference on Advances in Social Networks Analysis and Mining, ASONAM (2016)
32. Tian, Q., Zhang, P., Li, B.: Towards predicting the best answers in community-based question-answering services. In: International Conference on Weblogs and Social Media, ICWSM (2013)
33. Wang, D., Nyberg, E.: A long short-term memory model for answer sentence selection in question answering. In: ACL (2015)
34. Wang, X.J., Tu, X., Feng, D., Zhang, L.: Ranking community answers by modeling question-answer relationships via analogical reasoning. In: SIGIR Conference on Research and Development in Information Retrieval. SIGIR '09, ACM (2009)
35. Wu, L., Baggio, J.A., Janssen, M.A.: The role of diverse strategies in sustainable knowledge production. PLoS ONE (2016)
36. Wu, W., Wang, H., Sun, X.: Question condensing networks for answer selection in community question answering. In: Association for Computational Linguistics (2018)
37. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. CoRR **abs/1709.04875** (2017)
38. Zhang, X., Li, S., Sha, L., Wang, H.: Attentive interactive neural networks for answer selection in community question answering. In: AAAI Conference on Artificial Intelligence (2017)
39. Zhuang, C., Ma, Q.: Dual graph convolutional networks for graph-based semi-supervised classification. In: World Wide Web Conference (2018)