Ranking User-Generated Content via Multi-Relational Graph Convolution

Kanika Narang, Adit Krishnan, Junting Wang, Chaoqi Yang, Hari Sundaram, Carolyn Sutter
University of Illinois at Urbana-Champaign, IL, USA
{knarang2,aditk2,junting3,chaoqiy2,hs1,carolyns}@illinois.edu

ABSTRACT

The quality variance in user-generated content is a major bottleneck to serving communities on online platforms. Current content ranking methods primarily evaluate textual/non-textual features of each user post in isolation. This paper demonstrates the utility of the implicit and explicit relational aspects across user content to assess their quality. First, we develop a modular platform-agnostic framework to represent the contrastive (or competing) and similarity-based relational aspects of user-generated content via independently induced content graphs. Second, we develop two complementary graph convolutional operators that enable *feature contrast* for competing content and *feature smoothing/sharing* for similar content. Depending on the edge semantics of each content graph, we embed its nodes via one of the above two mechanisms. We show that our contrastive operator creates *discriminative magnification* across the embeddings of competing posts. Third, we show a surprising result—applying classical boosting techniques to combine embeddings across the content graphs significantly outperforms the typical stacking, fusion, or neighborhood aggregation methods in graph convolutional architectures. We exhaustively validate our method via *accepted answer prediction* over fifty diverse Stack-Exchanges[1] with consistent *relative* gains of $\sim 5\%$ *accuracy* over state-of-the-art neural, multi-relational and textual baselines.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; **Personalization**; **Web and social media search**; *Recommender systems*;

KEYWORDS

User-Generated Content, Content Ranking, Graph Convolution, Multi-View Learning, Multi-Relational Graphs

ACM Reference Format:
Kanika Narang, Adit Krishnan, Junting Wang, Chaoqi Yang, Hari Sundaram, Carolyn Sutter. 2021. Ranking User-Generated Content via Multi-Relational Graph Convolution. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3404835.3462857

[1]https://stackexchange.com/

1 INTRODUCTION

With technology affordances and low entry barriers, large-scale online platforms such as discussion forums, product/business review platforms, and social networks incorporate and even rely on user-generated content. Unlike published content, user content is added at an unprecedented rate. The popular Stack-Overflow Q&A platform hosts more than twenty million questions with over ten thousand added each day. Unlike traditional query-document ranking, user-generated content is organized by the intended audience. For instance, answers on the Stack-Exchange platform are grouped by questions, tags, and other criteria, while product categories and other identifiers group product reviews. Most platforms annotate posts or reviews as *accepted*, *verified* or *most useful* by category, either via expert feedback or user votes. Thus, unlike the *Learning-to-Rank* training approach [19], user-generated content requires category-specific ranking irrespective of the model choice, i.e., given a particular category (or grouping) of interest, what is the most useful post that may be shown to a user?

Prior work typically identifies salient content features to estimate quality [2, 13, 20]. In this vein, neural text models represent textual features [41, 42, 37] to assess the quality of each post in isolation with a category-independent ranking function. We refer to this as a *reflexive approach* since posts are evaluated agnostic to competitors or peer content. The reflexive approach is insufficient when content is specifically ranked by contrast to other content in the same category (e.g., answers to a specific question or critical reviews to a product), or even by similarity to content in a different category (e.g., early answers on Stack-Exchanges [36]).

In contrast to *reflexive* approaches, we can contextualize the category-specific roles of user content by explicitly incorporating their relational aspects. Specifically, we represent these relational aspects as content graphs, where the edge semantics can be chosen to represent various relations across user-generated content. The links of a content graph can be chosen to connect competing content and contrast their features to obtain an improved ranking, i.e., *contrastive graphs.* Alternately, it can connect content across different categories based on their relative similarities to provide cues to content quality, i.e., *similarity graphs.* There may be several platform-specific approaches to construct *contrastive graphs* or *similarity graphs* to represent the relational aspects of content on the platform. In contrast to reflexive approaches that only consider content features, we develop a *platform-agnostic* framework to simultaneously incorporate *contrastive and similarity content graphs* to holistically rank user-generated content.

Graph Convolutional Networks (GCNs) learn scalable attributed graph representations towards tasks such as node classification [15, 38] and link prediction [28]. However, popular GCN architectures [15, 28, 11] implicitly assume latent similarities among connected node neighborhoods and generate smoothed vertex representations [24]. This view is incompatible with contrastive content graphs linking competing posts on online platforms. GCN extensions such as signed and multi-relational networks [4, 28] do not address the fundamental challenge of modeling feature contrast among connected nodes. GCN layers implicitly smooth neighbor features and shrink feature contrast [39, 24].

This paper develops a novel framework to incorporate both the similarity and contrastive relational aspects of user-generated content through *independently induced content graphs*. We classify content graphs in three broad platform-agnostic aspects: *contrastive graphs* (where content nodes are ranked against their neighbors), *similarity graphs*, and finally, *reflexive graphs* with only self-edges to incorporate purely feature-based ranking. We then develop graph convolutional operators to achieve feature smoothing and contrast mechanisms for similarity and contrastive graphs. Finally, we develop a parsimonious boosting framework to combine representation across all content graphs towards content quality estimation. In summary, our contributions are:

- **Modular Platform-Agnostic Relational Framework for Content Ranking:** We develop a unified boosting framework incorporating content graphs to represent the relational aspects of user content. While prior work evaluates content in isolation [2, 13], we develop feature contrast and feature smoothing mechanisms to accommodate diverse content graph semantics in our Induced Relational Graph Convolutional Network (**IR-GCN**) framework.
- **Discriminative Semantics:** We propose a simple contrastive convolution approach to differentiate connected vertices in a contrastive graph. While graph convolution typically encapsulates neighbor feature smoothing (e.g., [15, 43]), we show that our contrast mechanism creates *discriminative magnification* to identify feature contrasts between competing content nodes.
- **Boosting vs. Neural Aggregation:** With extensive empirical results, we show the effectiveness of learning different convolutional representations across diverse content graphs followed by a boosting approach to combine their predictions. The proposed boosting approach is computationally faster and more effective than neural architectures that stack, fuse, or aggregate the content vertices representations across the different content graphs.

We extensively validate our IR-GCN framework on *fifty* Stack-Exchange Question-Answer websites chosen over all five discussion domains [2]. We achieve consistent relative improvements of over 5% accuracy and 3% MRR over state-of-the-art text and multi-relational graph-based baselines on the *accepted answer prediction* task.

2 PRELIMINARIES

Let Q denote the set of categories of user-generated content on the platform or website of interest. Each category $q \in Q$ has an associated set of competing user-generated posts $\mathcal{A}_q$. Thus, for each category-post pair (q, a) where $q \in Q, a \in \mathcal{A}_q$, we generate a quality score to rank posts within the same category.

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

Content Graphs: We induce multiple content graphs to represent the relational aspects of the user-generated posts. The content graph induction techniques depend on the content platform. We classify each content graph as a similarity graph, where connections between nodes indicate similarities in content quality, or a contrastive graph where connected nodes directly compete against each other, e.g., posts under the same category. Finally, we include the trivial reflexive content graph with only self- edges between content nodes to incorporate relation-agnostic feature-based ranking. Thus, the set of all content graphs $\mathcal{G}$ can be given by,

$$\mathcal{G} = \mathcal{G}_C \cup \mathcal{G}_S \cup \mathcal{G}_R$$

where $\mathcal{G}_C$, $\mathcal{G}_S$ and $\mathcal{G}_R$ denotes the sets of contrastive graphs, similarity graphs and the reflexive graph respectively. We may have multiple contrastive graphs connecting competing content or multiple similarity graphs connecting similar content depending on the platform, but only one reflexive graph in the set $\mathcal{G}_R$ (the set of all nodes with only self-edges).

Each graph $\mathbf{G} \in \mathcal{G}_C$ / $\mathcal{G}_S$ / $\mathcal{G}_R$ is given by $\mathbf{G} = (V, E_\mathbf{G})$ (adjacency matrix $\mathbf{A}_\mathbf{G}$), where node set V contains all category-post pairs (q, a) and edge set $E_\mathbf{G}$ varies for each content graph. Our framework unifies content graphs to generate an aggregate quality score for each category-post pair (q, a).

Task Description: On most platforms incorporating user content, users (or platform moderators) annotate a single post in each category as the highest quality/most helpful post. For instance, accepted Stack-Exchange answers [32], verified Reddit posts [20] and the most helpful product reviews on E-commerce websites [21]. To evaluate our ranking approach, we generate quality scores for each category-post pair (q, a) $(q \in Q, a \in \mathcal{A}_q)$ and match the ground truth accepted/verified post with the highest-scored post in each category. In essence, we can only validate the top result of our ranking against the ground truth data. However, depending on the available ground truth labels, our model outputs trivially lend to a *top-N* ranking evaluation.

3 INDUCED RELATIONAL GCN (IR-GCN)

We now briefly review Graph Convolution Networks (GCNs) and develop graph convolutional operators to achieve feature contrast between connected nodes in the contrastive graphs $\mathbf{G} \in \mathcal{G}_C$ (Section 3.2) and feature smoothing across connected nodes in the similarity graphs $\mathbf{G} \in \mathcal{G}_S$ (Section 3.3) respectively.

3.1 Graph Convolution

Graph convolution models adapt the convolution operation on regular grids (such as image pixels) to graph-structured data $\mathbf{G} = (V, E_\mathbf{G})$, learning low-dimensional vertex representations. Let N denote the number of vertices, and $\mathbf{X} \in \mathbb{R}^{N*d}$ the d-dimensional features of the vertices. The graph convolution operation for vertex $v \in V$ with features $\mathbf{X}_v \in \mathbb{R}^N$, and a learned filter g_θ in the fourier domain can be efficiently approximated via first-order terms [15],

$$g_\theta * \mathbf{X}_v = \theta_0 \mathbf{X}_v + \theta_1 (\mathbf{L} - \mathbf{I}_N) \mathbf{X}_v \quad (1)$$

with the normalized graph Laplacian, $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{A}$ denotes the adjacency matrix of graph $\mathbf{G}$ with N vertices and $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ is the corresponding diagonal degree matrix. The filter parameters, θ_0 and θ_1 are shared across all vertices.

3.2 Contrastive Graph Convolution

3.2.1 Contrastive Graph Structure. We define contrastive graphs $\mathbf{G} \in \mathcal{G}_C$ to inter-connect competing posts that are ranked against each other. There are several ways to determine sets of competing posts depending on the platform and viewer objectives, e.g., answers to the same question in a Q&A forum or product reviews of the same product or product category. In each case, the resulting graph is a collection of *disconnected cliques* of competing content, where each clique inter-connects all competing posts within a specific category, irrespective of how they are constructed. We illustrate the resulting disconnected clique structure in Figure 1.

3.2.2 Contrastive Convolution. We now describe our graph convolution approach for the contrastive graphs to enhance feature contrast in the latent representation space for competing posts within each clique. To achieve this, we modify Equation (1) by setting $\theta = \theta_0 = \theta_1$. Consider a specific vertex u with features $\mathbf{x}_u$ (such as user features and textual features). The above modification then leads to the following convolution operation for vertex $\mathbf{u}$:

$$g_\theta * \mathbf{x}_u = \theta\left(\mathbf{I}_N + \mathbf{L} - \mathbf{I}_N\right)\mathbf{x}_u = \theta\left(\mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)\mathbf{x}_u \quad (2)$$

When we apply Equation (2) to modify each graph convolution layer of the GCN, the output for the k^{th} layer can be given by:

$$\mathbf{Z}^k_{\mathbf{G}} = \sigma\left(\left(\mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}_{\mathbf{G}}\mathbf{D}^{-1/2}\right)\mathbf{Z}^{k-1}_{\mathbf{G}}\mathbf{W}^k_{\mathbf{G}}\right) \quad (3)$$

where $\mathbf{A}_{\mathbf{G}}$ denotes the adjacency matrix of the contrastive graph $\mathbf{G} \in \mathcal{G}_C$, $\mathbf{Z}^k_{\mathbf{G}} \in \mathbb{R}^{N\times d}$ is the matrix of the k^{th} layer vertex embeddings, and $\mathbf{Z}^{k-1}_{\mathbf{G}}$ denotes the previous layer embeddings under contrastive convolution, N the total number of vertices and d is the chosen embedding dimensionality. Note that $\mathbf{Z}^0_{\mathbf{G}} = \mathbf{X}$ where $\mathbf{X}$ is the input feature matrix of the vertices. $\mathbf{W}^k_{\mathbf{G}}$ denotes the filter parameters of the k^{th} convolutional layer on graph $\mathbf{G}$, and $\sigma(\cdot)$ is the chosen layer activation (e.g. ReLU, tanh).

Let us now consider Equation (3) for a specific vertex u. The convolution operation in Equation (3) expands as follows, where $\mathbf{Z}^k_{\mathbf{G}}(u)$ denotes the k^{th} convolution layer output for u:

$$\mathbf{Z}^k_{\mathbf{G}}(u) = \sigma\left(\left(\mathbf{Z}^{k-1}_{\mathbf{G}}(u) - \frac{1}{|\mathcal{N}_u|}\sum_{v\in\mathcal{N}_u}\mathbf{Z}^{k-1}_{\mathbf{G}}(v)\right)\mathbf{W}^k_{\mathbf{G}}\right) \quad (4)$$

Clearly, the k^{th} layer computes the feature differences (or *feature contrast*) between the node u and its neighbors in the $k-1^{th}$ layer as opposed to the default operation in Equation (1).

Since our contrastive graph contains cliques of competing vertices, each vertex is a neighbor to all the other competing vertices. To understand the effect of Equation (3) on a specific (q, a) pair, consider competing vertices u and v in a clique of size n (so that $|\mathcal{N}_u| = |\mathcal{N}_v|$ = n-1). From Equation (4):

$$\mathbf{Z}^k_{\mathbf{G}}(u) - \mathbf{Z}^k_{\mathbf{G}}(v) = \sigma\left(\left(\mathbf{Z}^{k-1}_{\mathbf{G}}(u) - \frac{1}{n-1}\sum_{v'\in\mathcal{N}_u}\mathbf{Z}^{k-1}_{\mathbf{G}}(v')\right)\mathbf{W}^k_{\mathbf{G}}\right) - \sigma\left(\left(\mathbf{Z}^{k-1}_{\mathbf{G}}(v) - \frac{1}{n-1}\sum_{v'\in\mathcal{N}_v}\mathbf{Z}^{k-1}_{\mathbf{G}}(v')\right)\mathbf{W}^k_{\mathbf{G}}\right) \quad (5)$$

Let us analyze Equation (5) by replacing the non-saturated part of σ with linear function $\sigma(y) = \alpha y + \beta$,

$$\begin{aligned}\underbrace{\mathbf{Z}^k_{\mathbf{G}}(u) - \mathbf{Z}^k_{\mathbf{G}}(v)}_{\text{contrast in the current layer}} &= \alpha\mathbf{W}^k_{\mathbf{G}}\left(\mathbf{Z}^{k-1}_{\mathbf{G}}(u) - \frac{1}{n-1}\mathbf{Z}^{k-1}_{\mathbf{G}}(v) - \mathbf{Z}^{k-1}_{\mathbf{G}}(v) + \frac{1}{n-1}\mathbf{Z}^{k-1}_{\mathbf{G}}(u)\right)\\ &= \alpha\underbrace{\mathbf{W}^k_{\mathbf{G}}}_{\text{learned filter}}\underbrace{\left(1+\frac{1}{n-1}\right)}_{\text{magnification}}\times\underbrace{\left(\mathbf{Z}^{k-1}_{\mathbf{G}}(u) - \mathbf{Z}^{k-1}_{\mathbf{G}}(v)\right)}_{\text{contrast in the previous layer}}\end{aligned} \quad (6)$$

Note that the other neighbor terms cancel out in Equation (6) except for the two vertices u & v, since they are part of the same clique and have the same neighbors except each other.

As a result, across each convolutional layer, the feature contrasts between competing vertices in the same clique are expanded by the inverse clique size. The learned filters $\mathbf{W}^k_{\mathbf{G}}$ have a more significant impact on the embedding displacements of competing vertices. We term this translation as *Discriminative Feature Magnification.* Equation (6) also shows a more pronounced magnification effect for smaller cliques, leading to sharper distinctions.

3.3 Similarity Graph Convolution

3.3.1 Similarity Graph Structure. We define similarity graphs $\mathbf{G} \in \mathcal{G}_S$ to link posts with similar characteristics across different categories. As opposed to direct feature similarity measures, we link posts that compare in similar ways against their respective competitors, i.e., similar by comparison to peers. Note that similarity by *comparison with the competing content* is particularly suited to category-wise ranking scenarios.

As in the contrastive case, there are multiple criteria to determine similar posts across two different categories, e.g., two product reviews may be similar by both being more informative than their respective competitors. At the same time, faster answers to questions on Q&A forums are similar by appearing before their respective competitors. In each case, the resulting similarity graph is a collection of *disconnected cliques* of content that compares in similar ways to their respective competitors, as illustrated in Figure 1.

3.3.2 Similarity Graph Convolution. Unlike the contrastive modification we introduce in Equation (2), we aim to identify feature similarities among connected nodes rather than feature differences or contrast. To enable feature smoothing for closely connected nodes in the similarity graph, we transform Equation (1) with the parameter inversion $\theta = \theta_0 = -\theta_1$ (as opposed to $\theta = \theta_0 = \theta_1$ in the contrastive case). The convolution for a specific vertex u with features $\mathbf{X}_u$ in a similarity graph can then be given by (following from Equation (1)),

$$g_\theta * \mathbf{X}_u = \theta\left(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)\mathbf{X}_u, \quad (7)$$

Another common formulation in practice normalizes the sum term, $\left(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)$ as $\mathbf{D}^{-1/2}\tilde{\mathbf{A}}\mathbf{D}^{-1/2}$ where $\tilde{\mathbf{A}} = \mathbf{I}_N + \mathbf{A}$. However, this formulation results in oversmoothing by averaging all neighbors of each node, including the node itself. For our disconnected clique graph, it results in all nodes within the same clique receiving the same identical result, which is the average of all embeddings in the clique. Thus, in Equation (7) we use the unnormalized sum term $\left(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)$.

When we apply Equation (7) to modify each graph convolution layer of a GCN, the output for the k^{th} layer can be given by:

$$\mathbf{Z}_{\mathrm{G}}^{k} = \sigma\left(\left(\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}_{\mathrm{G}}\mathbf{D}^{-1/2}\right)\mathbf{Z}_{\mathrm{G}}^{k-1}\mathbf{W}_{\mathrm{G}}^{k}\right) \tag{8}$$

with $\mathbf{A}_{\mathrm{G}}$ denoting the adjacency matrix of similarity graph $\mathrm{G} \in \mathcal{G}_{\mathcal{S}}$, and all the other notations following from Equation (3).

Let us consider the effect of the convolution in Equation (8) for a specific vertex u with $k-1^{th}$ layer embedding $\mathbf{Z}_{\mathrm{G}}^{k-1}(u)$. Unlike the contrastive equation in Equation (4), Equation (8) computes $\mathbf{Z}_{\mathrm{G}}^{k}(u)$ as follows:

$$\begin{aligned}\mathbf{Z}_{\mathrm{G}}^{k}(u) &= \sigma\left(\left(\mathbf{Z}_{\mathrm{G}}^{k-1}(u) + \frac{1}{|\mathcal{N}_u|}\sum_{v\in\mathcal{N}_u}\mathbf{Z}_{\mathrm{G}}^{k-1}(v)\right)\mathbf{W}_{\mathrm{G}}^{k}\right)\\ &= \sigma\left(\left(\frac{|\mathcal{N}_u|-1}{|\mathcal{N}_u|}\mathbf{Z}_{\mathrm{G}}^{k-1}(u) + \frac{|\mathcal{N}_u|+1}{|\mathcal{N}_u|}\mu_{\mathrm{G}}^{k-1}\right)\mathbf{W}_{\mathrm{G}}^{k}\right)\end{aligned} \tag{9}$$

where μ_{G}^{k-1} is the average over all $k-1^{th}$ layer embeddings in the clique, including u. Unlike the feature difference computed by Equation (6), the vertex embedding in the k^{th} convolution layer now aggregates the average of the previous layer neighbor embeddings with the vertex instead of computing their difference, due to the sign inversion in Equation (7).

When we analyze the above transformation analogous to Equation (5) by replacing $\sigma(y) = \alpha y + \beta$ in the non-saturated region, we can easily verify that for vertices u and v in a clique of size n:

$$\begin{aligned}&\underbrace{\mathbf{Z}_{\mathrm{G}}^{k}(u) - \mathbf{Z}_{\mathrm{G}}^{k}(v)}_{\text{contrast in the current layer}}\\ &= \alpha \underbrace{\mathbf{W}_{\mathrm{G}}^{k}}_{\text{learned filter}} \underbrace{\left(1 - \frac{1}{n-1}\right)}_{\text{shrinkage}} \times \underbrace{\left(\mathbf{Z}_{\mathrm{G}}^{k-1}(u) - \mathbf{Z}_{\mathrm{G}}^{k-1}(v)\right)}_{\text{contrast in the previous layer}}\end{aligned} \tag{10}$$

As a result, each convolutional layer with Equation (7) shrinks the feature differences between vertices in the same clique. The learned filters $\mathbf{W}_{\mathrm{G}}^{k}$ have a progressively smaller impact on the embedding displacements of connected vertices over embedding layers. This enables identifying weighted feature similarities within a clique, as opposed to weighted feature differences in Equation (6).

3.4 Reflexive Graph Convolution

3.4.1 Reflexive Graph. We construct the reflexive graph $\mathrm{G} \in \mathcal{G}_{\mathcal{R}}$ with the set of all self-edges over the content nodes. We now demonstrate how convolving the reflexive graph is equivalent to learning a feedforward neural network classifier on the vertex feature set.

3.4.2 MLP Equivalence. We construct the reflexive graph with self-loops over all vertices, resulting $\mathbf{A}_{\mathrm{G}} = \mathbf{D} = \mathbf{I}_N$. Thus, applying the similarity convolution layer defined in Equation (8),

$$\mathbf{Z}_{r}^{k} = \sigma\left(2 * \mathbf{I}_N\mathbf{Z}_{r}^{k-1}\mathbf{W}_{r}^{k}\right) \tag{11}$$

The above reflexive convolution is equivalent to a feedforward neural network classifier with the same activation function (ignoring the x2 scaling), transforming vertex features in isolation.

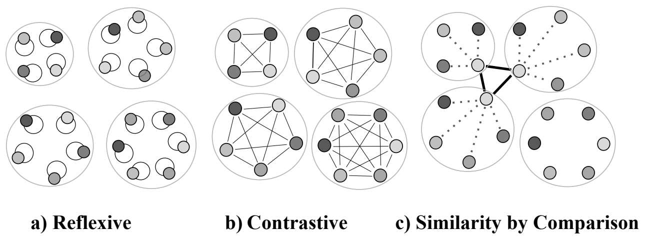

Figure 1: Reflexive, Contrastive and Similarity graphs among category-post (q, a) pairs. The reflexive graph considers only node features, contrastive graphs connect competing content; similarity graphs connect user content across different categories if they differ from their competitors similarly. Solid lines indicate similarity, while dotted lines indicate contrast. The similarity graph in the above example connects content in three out of four categories.

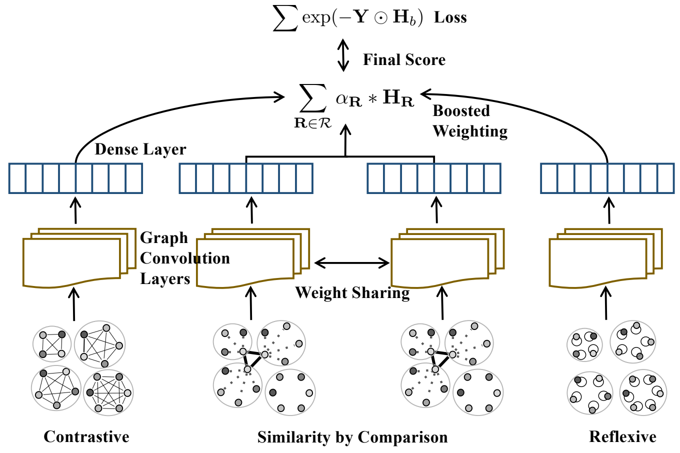
$\sum \exp(-\mathbf{Y} \odot \mathbf{H}_b)$ Loss
Final Score
$\sum_{\mathrm{R}\in\mathcal{R}} \alpha_{\mathrm{R}} * \mathbf{H}_{\mathrm{R}}$
Boosted Weighting
Dense Layer
Graph Convolution Layers
Weight Sharing
Contrastive
Similarity by Comparison
Reflexive

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

While the content graphs in $\mathcal{G}_{C}$ and $\mathcal{G}_{\mathcal{S}}$ represent the contrasts and similarities between user content, the reflexive graph enables independent feature-based vertex embeddings. In the next section, we describe our aggregation strategy to combine the vertex representations across the three sets of graphs, $\mathcal{G}_{C}$, $\mathcal{G}_{\mathcal{S}}$ and $\mathcal{G}_{\mathcal{R}}$ to compute the aggregate quality score for each content vertex.

4 CONTENT GRAPH AGGREGATION

The convolutional operators described in the previous section generate vertex embeddings for each contrastive graph via Equation (4), similarity graphs via Equation (9), and the reflexive graph via Equation (11). We now describe a novel hierarchical aggregation approach to unify the vertex embeddings within each set of graphs, i.e., $\mathcal{G}_{C}$, $\mathcal{G}_{\mathcal{S}}$ and $\mathcal{G}_{\mathcal{R}}$. We then introduce a top layer to combine per-vertex scores across the three sets, $\mathcal{G}_{C}$, $\mathcal{G}_{\mathcal{S}}$ and $\mathcal{G}_{\mathcal{R}}$ with a complementary boosting approach. Our approach enables a vertex-specific weighting of the three types of representations to estimate the overall quality score.

4.1 Within-Set Graph Aggregation

The contrastive graphs $\mathrm{G} \in \mathcal{G}_C$ share structural similarities representing how competing content and categories are organized on the platform. Analogously, the similarity graphs $\mathrm{G} \in \mathcal{G}_S$ share correlated sub-structures across the similarity criteria, representing user participation patterns on the platform.

Weight sharing: To facilitate the discovery of structural commonalities in their respective vertex convolutions, we share the weight parameters of the GCN layers ($\mathbf{W}_\mathrm{G}^k$) across all the graphs within the same set. Thus, we only learn three distinct sets of convolutional filter weights, one for graphs in $\mathcal{G}_C$ (i.e., for Equation (4)), one for graphs in $\mathcal{G}_S$ (i.e., for Equation (9)), and one for the reflexive graph in $\mathcal{G}_\mathcal{R}$ (i.e., for Equation (11)).

Further, we employ a simple alignment loss term ([43, 23]) to align the learned vertex representations at the *final convolutional layer*, i.e., $\mathbf{Z}_\mathrm{G}^K$ across the graphs within each set. This results in the following two norm loss terms:

$$||\mathbf{Z}_\mathrm{G}^K - \mathbf{Z}_{\mathrm{G}'}^K|| \ \ \forall \mathrm{G}, \mathrm{G}' \in \mathcal{G}_C, \ \ ||\mathbf{Z}_\mathrm{G}^K - \mathbf{Z}_{\mathrm{G}'}^K|| \ \ \forall \mathrm{G}, \mathrm{G}' \in \mathcal{G}_S \quad (12)$$

We detail the overall loss function and training method in Algorithm 2.

Score computation: For each graph $\mathrm{G} \in \mathcal{G}_C / \mathcal{G}_S / \mathcal{G}_\mathcal{R}$, we obtain the d-dimensional vertex embeddings at the K^{th} layer of the respective graph convolution operators (where K denotes the last convolution layer), $\mathbf{Z}_\mathrm{G}^K \in \mathbb{R}^{NXd}$ and compute a quality score by multiplying $\mathbf{Z}_\mathrm{G}^K \in \mathbb{R}^{NXd}$ with graph-specific transform parameters $\widetilde{W}_\mathrm{G} \in \mathbb{R}^{d \times 1}$, corresponding to the dense layers in Figure 2.

The graph-specific scores are combined to compute three set-level quality scores for each vertex, ($\mathbf{H}_C, \mathbf{H}_S, \mathbf{H}_\mathcal{R} \in \mathbb{R}^{NX1}$),

$$\mathbf{H}_C = \sum_{\mathrm{G} \in \mathcal{G}_C} \mathbf{Z}_\mathrm{G}^K \widetilde{W}_\mathrm{G}, \ \ \mathbf{H}_S = \sum_{\mathrm{G} \in \mathcal{G}_S} \mathbf{Z}_\mathrm{G}^K \widetilde{W}_\mathrm{G}, \ \ \mathbf{H}_\mathcal{R} = \sum_{\mathrm{G} \in \mathcal{G}_\mathcal{R}} \mathbf{Z}_\mathrm{G}^K \widetilde{W}_\mathrm{G} \quad (13)$$

Note that the third sum, $\sum_{\mathrm{G} \in \mathcal{G}_\mathcal{R}}$ is trivial since there is only one reflexive graph, while we may define multiple contrastive and similarity graphs depending on the platform.

Algorithm 1 IR-GCN Boosted Score Computation

```
1:  function Forward($\mathbf{X}, \mathbf{Y}, \{A_\mathrm{G}\}$)
2:      $\mathbf{H}_b \leftarrow \mathbf{0}$
3:      for $t \in \{C, S, \mathcal{R}\}$ do
4:          $\{\mathbf{Z}_\mathrm{G}^K\}_{\mathrm{G} \in \mathcal{G}_t} \leftarrow Conv(\mathbf{X}, \{A_\mathrm{G}\}_{\mathrm{G} \in \mathcal{G}_t})$
5:                                                  ▷ Equation 3, 8, 11
6:          $\mathbf{H}_t = \sum_{\mathrm{G} \in \mathcal{G}_t} \mathbf{Z}_\mathrm{G}^K \times \widetilde{\mathbf{W}}_\mathrm{G}$          ▷ Equation 13
7:          $\mathbf{e}_t \leftarrow \exp(-\mathbf{Y} \odot \mathbf{H}_b)$
8:                                      ▷ $\odot \longrightarrow$ Hadamard Product
9:          $\alpha_t \leftarrow \frac{1}{2} \ln \frac{\sum \mathbf{e}_t \odot \mathbb{1}((\mathbf{Y} \odot \mathbf{H}_t) > 0)}{\sum \mathbf{e}_t \odot \mathbb{1}((\mathbf{Y} \odot \mathbf{H}_t) < 0)}$
10:                                        ▷ $\sum \longrightarrow$ reduce-sum
11:                   ▷ $\mathbb{1}(.) \longrightarrow$ element-wise Indicator function
12:         $\mathbf{H}_b \leftarrow \mathbf{H}_b + \alpha_t * \mathbf{H}_t$          ▷ Update boosted score
13:     end for
14:     return $\mathbf{H}_b$, $\{\mathbf{H}_t\}_{t \in \{C, S, \mathcal{R}\}}$, $\{\mathbf{Z}_\mathrm{G}^K\}_{\mathrm{G} \in \mathcal{G}_t}$
15:       ▷ Boosted scores, Set-level scores, Vertex representations
16: end function
```

Algorithm 2 IR-GCN Overall Training Algorithm

Input: Input Feature Matrix X, Acceptance labels for each tuple, Y, Adjacency matrices $\{A_\mathrm{G}\} \forall \mathrm{G}$

Output: Trained Model i.e., Weight parameters $W_\mathrm{G}^1 \dots W_\mathrm{G}^k$ and transform parameters $\widetilde{W}_\mathrm{G}$, $\forall \mathrm{G}, k \in [1, K]$

```
1:  for $n \leftarrow 1$ to num-epochs do
2:      $\mathbf{H}_b, \{\mathbf{H}_C, \mathbf{H}_S, \mathbf{H}_\mathcal{R}\}, \{\mathbf{Z}_\mathrm{G}^K\}_{\forall \mathrm{G}} \leftarrow$ Forward$(X, Y, \{A_\mathrm{G}\}_{\forall \mathrm{G}})$
3:                                                      ▷ Algorithm 1
4:      for $t \in \{C, S, \mathcal{R}\}$ do
5:          $\mathcal{L}_b \leftarrow \sum \exp(-\mathrm{Y} \odot \mathrm{H}_b) + \gamma_1 \mathcal{L}_1(.) + \gamma_2 \mathcal{L}_2(.)$
6:                                          ▷ $\sum \longrightarrow$ reduce-sum
7:                                  ▷ $\odot \longrightarrow$ Hadamard Product
8:          $\mathcal{L}_t \leftarrow 0$
9:          for $\mathrm{G} \in \mathcal{G}_t$ do
10:             $\mathcal{L}_\mathrm{G} \leftarrow \sum \exp(-\mathrm{Y} \odot \mathrm{H}_t)$
11:             $\mathcal{L}_t \leftarrow \mathcal{L}_t + \mathcal{L}_\mathrm{G} + \frac{1}{2} \sum_{\mathrm{G}' \neq \mathrm{G}} ||\mathbf{Z}_{\mathrm{G}'}^K - \mathbf{Z}_\mathrm{G}^K||$
12:         end for
13:         $\mathcal{L}_b \leftarrow \mathcal{L}_b + \lambda(n) \mathcal{L}_t$
14:         for $\mathrm{G} \in \mathcal{G}_t$ do
15:             $W_\mathrm{G}^k \leftarrow W_\mathrm{G}^k + \eta_{\mathrm{ADAM}} \frac{\partial \mathcal{L}_b}{\partial W_\mathrm{G}^k}$          ▷ $\forall k \in [1, K]$
16:             $\widetilde{W}_\mathrm{G} \leftarrow \widetilde{W}_\mathrm{G} + \eta_{\mathrm{ADAM}} \frac{\partial \mathcal{L}_b}{\partial \widetilde{W}_\mathrm{G}}$
17:         end for
18:     end for
19: end for
```

4.2 Cross-Set Aggregation

In contrast to the within-set similarities that we model in Equation (12), the set-level quality scores for a given vertex u, i.e.,$\mathbf{H}_C(u)$ (contrast score), $\mathbf{H}_S(u)$ (similarity score) and $\mathbf{H}_\mathcal{R}(u)$ (reflexive/feature score) represent different facets of the vertex. Gradient boosting techniques are known to improve performance when individual classifiers, including neural networks [29], are accurate on different subsets or facets of data entities. A natural solution then is to apply boosting to the three set-level scores to bridge the weaknesses of each set. We employ Adaboost [9] to compute boosted scores, $\mathbf{H}_b \in \mathbb{R}^{N \times 1}$ over the three sets of graphs (Line 12,algorithm 1). While the three set-level scores in Equation (13) act as independent regressors for each vertex with weight parameters $\widetilde{W}_\mathrm{G}$ (Equation (13)) on the underlying content graphs, the adaboost coefficients α_C, α_S and $\alpha_\mathcal{R}$ (Line 9, Algorithm 1) weight the three set-level scores in Line 12, Algorithm 1 to generate the aggregate boosted score $\mathbf{H}_b$. Quality score $\mathbf{H}_b$ is then used to rank competing posts under each category in our experiments.

As stated in Section 2, we employ the ground truth acceptance or verification labels for posts in each category, $\mathbf{Y} \in \mathbb{R}^{NX1}$ (+1 for (q, a) category-post pairs, -1 for the rest) to estimate the Adaboost coefficients for the set-level scores. The entries of the element-wise product of the label and the three set-level scores in Line 9, Algorithm 1, $\mathbf{Y} \odot \mathbf{H}_t$ are positive for content vertices where $sign(\mathbf{H}_t)$ matches the ground truth label $\mathbf{Y}$, and thus independently weight each set-level score for each vertex.

4.3 Overall Training Algorithm

Algorithm 2 describes the overall training algorithm for our IR-GCN model. In each training epoch, we first compute the boosted

Table 1: Dataset statistics for the three largest Stack Exchanges across all five domains. $|Q|$: number of questions; $|\mathcal{A}|$: number of answers; $|U|$: number of users; $\mu(|\mathcal{A}_q|)$: mean number of answers per question. The professional/business domain has slightly more answers per question on average. Technology exhanges typically have the largest number of question among the five domains.

	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

prediction scores $\mathbf{H}_b$, as described in algorithm 1. We then compute the overall exponential loss $\mathcal{L}_b$ (Line 5, algorithm 2),

$$\mathcal{L}_b \leftarrow \sum \exp(-\mathbf{Y} \odot \mathbf{H}_b) + \gamma_1 \mathcal{L}_1(.) + \gamma_2 \mathcal{L}_2(.) \qquad (14)$$

where the L1 and L2-norm regularizers ($\mathcal{L}_1$, $\mathcal{L}_2$) are computed over the weight parameters, $\mathbf{W}_{\mathrm{G}}^k$, $\forall \mathrm{G}, k$. Note that we employ weight sharing within each set of graphs.

The within-set exponential losses for the three sets, $\mathcal{G}_C, \mathcal{G}_S, \mathcal{G}_{\mathcal{R}}$ (Line 9-12,algorithm 2) are alternatingly optimized with $\mathcal{L}_b$. We apply an *exponential annealing schedule*, $\lambda(n)$ over training epochs (n) to dampen the within-set losses in later training epochs. Annealing ensures a robust allocation of the Adaboost coefficients among the set-level scores for each vertex.

5 EXPERIMENTS

This section describes our dataset, experimental setup, baselines, evaluation metrics, and implementational details. We present a wide range of quantitative and qualitative experimental results to validate our approach. Our implementations are available here[3].

5.1 Dataset

We validate our approach on *all large Stack-Exchange* Q&A websites across all five discussion domains, Technology (**T**), Culture/Recreation (**C**), Life/Arts (**L**), Science (**S**) and Professional (**P**). Specifically, we collect the ten largest Stack-Exchanges as of March 2019 in each of the five domains (Table 1). Note that categories are represented by questions in each Stack-Exchange and posts by user-generated answers to the respective questions.

5.1.1 Content Graph Creation: We now describe the contrastive, similarity, and reflexive content graph construction for each Stack-Exchange. The set of all question-answer pairs in a given Stack-Exchange constitutes the content graphs vertices, while the edge set differs across the graphs.

Contrastive graph creation: In each Stack-Exchange, we create a single contrastive graph $\mathrm{G_C}$, i.e., contrastive graph set, $\mathcal{G}_C = \{\mathrm{G_C}\}$. For each question $q \in Q$, we inter-connect competing answers $a \in \mathcal{A}_q$. Thus, $\mathrm{G_C}$ is a graph of disconnected cliques as described in Section 3.2.1 with one clique per question on the Stack-Exchange.

Similarity graph creation: For each Stack-Exchange, we create two similarity graphs with two different similarity criteria, motivated by prior work [36] demonstrating the importance of the user skill and relative arrival times of answers on Q&A for accepted answer prediction:

TrueSkill similarity graph $\mathrm{G_{TS}}$ connects answers where the author skills differ by a similar margin $\delta = 4$ against competing authors, i.e., when the authors are equally less or more skilled than their competitors. We compute author skill via Bayesian TrueSkill [12].

Arrival similarity graph $\mathrm{G_{AS}}$ connects answers to different questions based on their relative arrival times against competitors, e.g., answers that both arrive more than a $\delta = 0.95$ fraction of a day before their respective competing answers. We chose δ values via aprior statistical analysis of the datasets. Thus, the similarity graph set is given by $\mathcal{G}_S = \{\mathrm{G_{TS}}, \mathrm{G_{AS}}\}$.

Reflexive Graph Creation For each Stack-Exchange, we create a single reflexive content graph $\mathbf{G_R}$ with every vertex connected to itself, i.e., $\mathcal{G}_{\mathcal{R}} = \{\mathbf{G_R}\}$.

5.1.2 Graph vertex features: We input the following node features for each (q, a) pair (vertex) to the first convolutional layers:

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 value 1 to the first posted answer, and 0 to the rest) [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 counts in the Aboutme profile sections for the user posting the question and the user posting the answer.

Time-dependent features such as answer upvotes/downvotes, user reputation, or user badges ([2]) are problematic since we only know the aggregate values, not their specific change with time. Second, since these values typically increase over time, it is unclear if an accepted answer receives votes *before* or *after* it was accepted. We discard these time-dependent features in our experiments.

5.2 Experimental Setup

5.2.1 Baselines. We pick diverse state-of-the-art baselines (Table 2) to incorporate relation-agnostic feature embedding methods, multi-relational approaches to fuse content graphs [43, 28], generate text embeddings via LSTMs [30], a text-based extension of our model, and variants of our model with different content graphs and aggregation strategies.

Random Forest (RF) [2, 32] trains on the feature set described in section 5.1 on each Stack-Exchange.

Feed-Forward network (FF) [13] learns non-linear transformations of the feature vectors for each (q, a) pair, and is equivalent to our reflexive graph convolution model.

Relational GCN (RGCN) [28] fuses the vertex embeddings across the different content graphs in each convolutional layer to compute an aggregated input (referred as early fusion in Table 2).

Dual GCN (DGCN) [43] trains a separate GCN for each graph and regularizes the output representations (referred as late fusion in Table 2).

[3]https://github.com/CrowdDynamicsLab/IRGCN

Table 2: Method summary. Graph convolutional approaches do not learn isolated vertex representations (we address this via reflexive convolution). Early fusion methods tightly couple convolutional layers for multiple graphs in contrast to final layer aggregation.

Methods	Models Contrast	Models Similarity	Incorporates Text	Isolated Vertex Representation	Early vs. Late Relation Embedding Fusion
			Feature-based Models		
RF, FF [2, 13]	No	No	No	Yes	NA
			Multi-Relational Graph Models		
RGCN [28]	No	Yes	No	No	Early Fusion
DGCN [43]	No	Yes	No	No	Late Fusion
			Text Embedding Model		
QA-LSTM/CNN [30]	No	No	Learns word embeddings	NA	NA
			Our Model Variants		
IR-GCN	Yes (C-GCN)	Yes (AS-GCN, TS-GCN)	No	Yes (R-GCN)	Late Fusion
IR-GCN + T-GCN	Yes (C-GCN)	Yes (AS-GCN, TS-GCN)	Text similarity graph, T-GCN	Yes (R-GCN)	Late Fusion

QA-LSTM/CNN [30] applies stacked bidirectional LSTMs followed by convolutional filters to embed the question and answer text, followed by cosine-similarity ranking.
IR-GCN Our model with all three types of graphs (contrastive, similarity, reflexive) described in Section 5.1, and the aggregation strategy in Section 4.
IR-GCN + Textual Similarity (T-GCN) We incorporate a textual *similarity* graph that connects answers that exhibit a similar cosine similarity value with their respective question text, based on the learned text representations from the QA-LSTM approach. The textual similarity graph is included as the third *similarity* graph in addition to the Arrival and TrueSkill graphs.
Individual GCN: We also enlist the performance results of each GCN model in isolation, i.e., **C-GCN** for Contrastive graph, **TS-GCN** for Trueskill similarity graph, **AS-GCN** for Arrival time similarity graph. As the reflexive graph is equivalent to **FF** (Section 3.4), we do not separately enlist results of **R-GCN**.
Neural Aggregators [11, 40, 6] In Section 5.4, we compare our aggregation strategy in Section 4 against common neural aggregation architectures to merge vertex embeddings across the content graphs.

- **Neighborhood Aggregation**: This approach embeds vertices by aggregating the embeddings of its neighbors over all content graphs [11, 28].
- **Stacking**: Multiple graph convolution layers stacked side-to-side, each handling a different content graph [40].
- **Fusion**: Follows a multi-modal fusion approach [6], where graphs are treated as distinct data modalities.
- **Shared Latent Structure**: Transfers knowledge across content graphs with embedding norm penalties (e.g., [43]).

5.2.2 Evaluation Metrics. We randomly construct a test question-set $\mathcal{T}_q \subset Q$ with 20% questions on each Stack-Exchange, and evaluate the acceptance labels of test (q, a) pairs $\mathcal{T} = \{(q, a)\}$ where $q \in \mathcal{T}_q$. Specifically, we train the model with the non-test (q, a) pairs and their acceptance labels and evaluate on two metrics: *Accuracy* to measure how often our top-scored answer for each test question matches the ground truth, and *Mean Reciprocal Rank (MRR)* to measure the predicted positions of the accepted answers for each test question [35]. Note that *accuracy* is averaged over all test (q, a) pairs $\mathcal{T} = \{(q, a)\}$, while *MRR* is averaged over the set of test questions, $\mathcal{T}_q$.

5.2.3 Implementation Details. We implemented all models with the PyTorch framework, and trained with the ADAM optimizer [14] and 50% dropout. We use four hidden layers in each GCN with hidden dimensions [50, 10, 10, 5] and ReLU activations. The $\mathcal{L}_1$ and $\mathcal{L}_2$ coefficients were set to $\gamma_1 = 0.05$ and $\gamma_2 = 0.01$. For TrueSkill similarity, we use margin $\delta = 4$, while for Arrival similarity, we use $\delta = 0.95$ based on apriori data analysis. We construct question-based mini-batches for parallelized training, owing to the disconnected clique structure of our graphs.

5.3 Performance Analysis

We show significant gains (Table 3) of around 5% relative accuracy and 3% relative MRR over state-of-the-art baselines across all five domains. We report mean results in each domain via 5-fold cross-validation on each Stack-Exchange. Note that MRR only measures the accepted answer rank. At the same time, accuracy considers *both*, accepted and unaccepted answers, indicating that our model achieves a balanced performance gain across both positives and negatives. Our results also show significantly less variance than the competing baselines, indicating more stable performance across questions and datasets.

Among the multiple individual graph variants of our model, C-GCN consistently achieves the highest performance on each Stack-Exchange, indicating the importance of feature contrast instead of feature smoothing. C-GCN outperforms the best baseline DGCN despite only convolving one of the four induced graphs. The contrastive graph compares the feature representations of candidate answers to each question with our proposed contrastive convolution approach, thus identifying the most prominent and indicative feature differences between accepted answers and their competitors. The AS-GCN model's strong performance shows that early answers tend to get accepted, and conversely, late ones do not. The reflexive graph predicts vertex labels independent of other answers to the same question. Thus, the performance of R-GCN measures the utility of the vertex features in isolation. TrueSkill similarity (TS-GCN) performs at par or slightly below R-GCN, but is significantly outperformed by AS-GCN and C-GCN. As expected,

Table 3: Accuracy and MRR values for StackExchanges vs. state-of-the-art baselines. Our model leads by around 5% accuracy and 3% MRR relative to the baselines. Contrastive GCN performs best among the strategies. The second-best model is marked with the $*$ symbol in each column. Our gains are statistically significant at 0.01 over the second-best model on a single tail paired t-test.

Method	Technology		Culture/Recreation		Life/Arts		Science		Professional/Business	
	Acc	MRR	Acc	MRR	Acc	MRR	Acc	MRR	Acc	MRR
RF [2, 32]	0.668 ± .02	0.683 ± .04	0.725 ± .02	0.626 ± .05	0.727 ± .05	0.628 ± .09	0.681 ± .02	0.692 ± .05	0.747 ± .04	0.595 ± .08
FF [13]	0.673 ± .03	0.786 ± .02	0.722 ± .02	0.782 ± .02*	0.736 ± .05	0.780 ± .03	0.679 ± .02	0.800 ± .03	0.746 ± .04	0.760 ± .05
DGCN [43]	0.707 ± .02	0.782 ± .02	0.752 ± .02	0.772 ± .03	0.767 ± .03	0.784 ± .04	0.714 ± .02*	0.792 ± .04	0.769 ± .03	0.751 ± .05
RGCN [28]	0.544 ± .05	0.673 ± .05	0.604 ± .02	0.646 ± .04	0.597 ± .04	0.655 ± .05	0.586 ± .05	0.683 ± .04	0.632 ± .04	0.657 ± .06
AS-GCN	0.678 ± .03	0.775 ± .02	0.730 ± .02	0.763 ± .03	0.738 ± .05	0.777 ± .04	0.669 ± .05	0.788 ± .03	0.749 ± .05	0.742 ± .05
TS-GCN	0.669 ± .03	0.779 ± .02	0.722 ± .02	0.764 ± .02	0.720 ± .06	0.766 ± .05	0.659 ± .04	0.790 ± .03	0.742 ± .05	0.747 ± .04
C-GCN	0.716 ± .02*	0.790 ± .02*	0.762 ± .02*	0.781 ± .02	0.774 ± .03*	0.788 ± .04*	0.708 ± .04	0.800 ± .03*	0.776 ± .04*	0.768 ± .03*
IR-GCN	**0.739 ± .02**	**0.794 ± .02**	**0.786±0.02**	**0.791±0.03**	**0.792±0.03**	**0.800±0.04**	**0.749 ± .02**	**0.809 ± .03**	**0.802 ± .03**	**0.785 ± .03**

the boosted graph aggregation model in IR-GCN outperforms the other variants, indicating the boosting approach's efficacy.

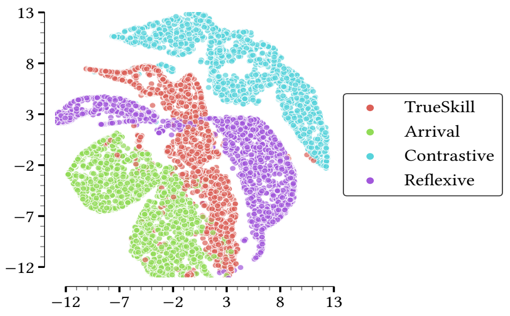

Figure 3: The t-stochastic neighbor embedding (t-SNE) [33] plot of our vertex representations clearly show the effect of our boosted aggregation approach. Each GCN learns a clearly demarcated facet of the vertex data and hence occupies a different part of the representation space (Chemistry StackExchange).

5.3.1 Qualitative Analysis. Figure 3 presents t-SNE distributions [33] of the learned vertex embeddings ($\mathbf{Z}_i^K$) of each GCN model when applied to the Chemistry Stack-Exchange (Science domain). Note that each GCN learns a sharply demarcated and distinct vertex embedding, owing to our boosting approach. Hence, all content graphs are essential to our final model performance.

5.3.2 DGCN vs. RGCN. Among the baseline graph ensemble approaches, DGCN performs significantly better than RGCN by an average relative difference of 26% for all domains. In the RGCN model, the semantically diverse embeddings of each GCN are linearly concatenated to compute outputs. Concatenation works well for knowledge graphs where each projected graph represents an aspect, thus accumulating information from each aspect. However, DGCN trains separate GCN models for each content graph and later merges their results via norm regularization instead of early fusion in RGCN.

5.3.3 DGCN vs. IR-GCN. DGCN incentivizes final-layer similarity in vertex representations learned by each GCN. This restriction is counterproductive with semantically diverse graphs. In contrast, we apply final-layer norm alignments separately within each set and not across all content graphs simultaneously, thus accounting for semantically diverse sets of graphs.

Method	**Tech**	**Culture**	**Life**	**Sci**	**Bus**
QA-LSTM/CNN[30]	0.665	0.717	0.694	0.629	0.726
T-GCN	0.692	0.738	0.764	0.678	0.771
IR-GCN	**0.739**	**0.786**	0.792	**0.749**	**0.802**
IR-GCN + T-GCN	**0.739**	0.780	**0.811**	0.745	0.789

Table 4: 5-fold accuracy comparison for the text-based methods QA-LSTM and T-GCN against IR-GCN.

5.3.4 Comparing against textual features. Our T-GCN extension outperforms QA-LSTM by significant margins as shown in Table 4. Since we use the embeddings learned by QA-LSTM to construct the T-GCN graph, we effectively improve performance simply by connecting similar answers across different questions. Connecting similar answers enables cross-question information sharing, thus improving performance. However, adding the T-GCN to IR-GCN does not lead to improvements. The similarity graphs based on the user features (Arrival and TrueSkill) do not benefit from alignment with the textual similarity graph.

5.4 Aggregator Architecture Variants

Method	**Tech**	**Culture**	**Life**	**Sci**	**Bus**
Stacking [40]	0.686	0.744	0.792	0.703	0.755
Fusion [6]	0.723	0.772	0.808	0.739	0.790
NeighborAgg [11, 28]	0.693	0.743	0.779	0.684	0.786
IR-GCN	**0.739**	**0.787**	**0.816**	**0.748**	**0.806**

Table 5: 5-fold Accuracy (in %) comparison of different aggregator architectures. Most architectures underperform our Contrastive GCN despite using all the 4 graphs. Fusion performs similarly, but is computationally expensive.

We compare our gradient boosting aggregation approach against other popular methods to merge convolutional neural representations, described in section 5.3. We conducted this study on the largest Stack-Exchanges in each of the five domains, i.e., Server-Fault (Technology), English (Culture), Science Fiction (Life), Physics (Science), Workplace (Business). Table 5 shows that C-GCN alone outperforms aggregator variants (which use all the 4 graphs) with Fusion being the best baseline. These results reaffirm that neural aggregation methods are not suited to combine semantically diverse content graph vertex embeddings. The fusion approach is also computationally expensive since the inputs and parameter sizes scale with the number of graphs.

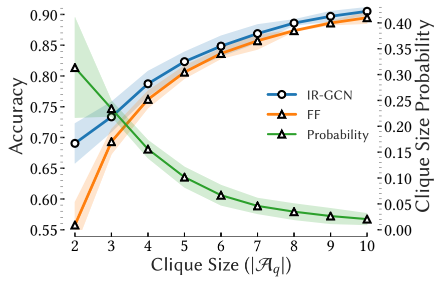

Figure 4: Accuracy of our IR-GCN model compared to the FF model with varying clique size (i.e. number of answers to a question, $|\mathcal{A}_q|$) for C-GCN. We report averaged results over the largest Stack-Exchange in each domain. Our model performs much better for smaller cliques, and the effect diminishes for larger cliques (eq. (3)). 80% questions comprise small cliques (<4 answers).

5.5 Discriminative Magnification effect

We show that our contrastive convolution operation achieves *Discriminative Magnification* (eq. (3)) to classify vertices. Figure 4 compares IR-GCN to the FeedForward (FF) approach across vertex cliques of varying sizes. Since the FF approach predicts node labels independent of other nodes, it is not impacted by the clique sizes. However, our magnification effect scales with the clique size as $(1 + 1/n - 1)$ (eq. (3)), resulting in stronger accuracy gains for smaller cliques (25% avg relative gain for $|\mathcal{A}_q| = 2$ and 7% for $|\mathcal{A}_q| = 3$). Thus, our model significantly outperforms the FF model for questions with fewer candidate answers. Further, 80% of the Stack-Exchange questions have fewer than four answers, resulting in significant overall gains.

5.6 Content Graph Ablation Study

We present the results of an ablation study with different graph combinations, C-GCN, AS/TS-GCN, and R-GCN in Table 6. We observe that the isolated variants underperform the unified ones, indicating the importance of cross-relation boosted combinations. Training contrastive and similarity graphs together in our boosted framework performs similar to our final model. This indicates that R-GCN is not very informative when contrast and similarity are included, since feature contrast and feature smoothing jointly account for most feature variations across nodes. As mentioned previously, C-GCN outperforms all the other individual content graphs, underscoring the importance of feature contrast. The ability to contrast or identify (as opposed to smoothing/aggregating) the distinguishing features of accepted answers against their competitors proves critical to the final classification result.

{Graphs}	Tech	Culture	Life	Sci	Bus
C-GCN	0.712	0.759	0.787	0.730	0.768
{TS, AS}-GCN	0.679	0.742	0.757	0.658	0.761
R-GCN	0.683	0.734	0.766	0.674	0.758
{TS, AS}-GCN + R-GCN	0.693	0.755	0.764	0.701	0.779
C-GCN + R-GCN	0.730	0.776	0.802	0.737	0.800
C-GCN + {TS, AS}-GCN	0.728	0.780	0.814	0.722	0.802
IR-GCN (all)	**0.739**	**0.787**	**0.816**	**0.746**	**0.806**

Table 6: 5-fold Accuracy(in %) comparison of different combinations of content graphs in our overall approach. Contrastive and similarity graphs combined perform similar to the final model.

6 RELATED WORK

Our work intersects multiple broad research threads connected to content selection and multi-relational graph convolution. In the context of user-generated content (primarily in discussion fora), prior content selection literature primarily includes feature-based models and deep text models.

Feature-based Models identify and incorporate user features, text 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. [13] trained classifiers for online forums, Burel et al. [2] emphasize the Q&A thread structure.

Deep Text Models learn question/answer text representations to rank answers [41, 37, 34]. Feng et al. [7] augment CNNs with discontinuous convolution for improved representations; Wang et al. [30] use stacked biLSTMs to match question-answer semantics.

Graph Convolution is applied in spatial and spectral domains to compute graph vertex representations for downstream tasks including classification [15], link prediction [28], multi-relational tasks [27] etc. Spatial approaches employ random walks or k-hop neighborhoods to compute vertex representations [22, 10, 31] while fast localized convolutions are applied in the spectral domain[3, 5]. Our work is inspired by spectral Graph Convolution [15], which outperforms spatial convolutions and scales to larger graphs. GCN extensions have been proposed for signed networks [4], motif-structures [25], inductive settings [11], multiple relations [43, 28] and diffusion [26]. However, GCN variants assume feature smoothing, which cannot model vertex feature contrast.

Multi-Relational Fusion: Fusing relation data modalities for joint representation is a well studied problem [1]. A few closely related threads include adversarial approaches to integrate social neighbor data [16, 17]; meta-learning to adapt across tasks [8, 18]. In contrast to these approaches, we emphasize the flexibility and simplicity of our multi-relational graph formulation for modeling the relational aspects of user-generated content.

7 CONCLUSION

This paper leverages the relational aspects of user-generated content to provide holistic content ranking on online platforms. We developed a novel induced relational graph convolution framework (IR-GCN) to address this question in a platform-agnostic manner. We made three contributions. First, we incorporate platform-specific criteria to induce content graphs representing relational aspects of user content and identify three broad platform-agnostic graph semantics to classify them. Second, we developed convolutional operators to efficiently achieve feature sharing and feature contrast across these induced content graphs. Our novel contrastive architecture achieves *Discriminative Magnification* between competing vertices.

Finally, we developed a hierarchical aggregation framework to integrate these graphs and outperform a wide range of neural aggregators on the Answer Selection task in fifty Stack-Exchanges. We identify two main threads to extend our work: developing a more nuanced classification beyond contrast and similarity and extending our aggregation strategy's critical concepts to a broader range of content search and ranking problems.

REFERENCES

[1] Pradeep K Atrey et al. 2010. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16, 6, 345–379.

[2] Grégoire Burel et al. 2016. Structural normalisation methods for improving best answer identification in question answering communities. In *International Conference on World Wide Web*.

[3] Michaël Defferrard et al. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*.

[4] Tyler Derr et al. 2018. Signed graph convolutional network. *CoRR*, abs/1808.06354.

[5] David K. Duvenaud et al. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*.

[6] Golnoosh Farnadi et al. 2018. User profiling through deep multimodal fusion. In *International Conference on Web Search and Data Mining* (WSDM '18). ACM.

[7] Minwei Feng et al. 2015. Applying deep learning to answer selection: A study and an open task. In *IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU*.

[8] Chelsea Finn et al. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.

[9] Yoav Freund and Robert E. Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*. Springer-Verlag.

[10] Aditya Grover and Jure Leskovec. 2016. Node2vec: scalable feature learning for networks. In *International Conference on Knowledge Discovery and Data Mining*.

[11] Will Hamilton et al. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*.

[12] Ralf Herbrich et al. 2006. Trueskill™: a bayesian skill rating system. In *International Conference on Neural Information Processing Systems*.

[13] Maximilian Jenders et al. 2016. Which answer is best?: predicting accepted answers in MOOC forums. In *International Conference on World Wide Web*.

[14] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980. arXiv: 1412. 6980.

[15] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907. arXiv: 1609.02907.

[16] Adit Krishnan et al. 2019. A modular adversarial approach to social recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1753–1762.

[17] Adit Krishnan et al. 2018. 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*. ACM, 1491–1494.

[18] Adit Krishnan et al. 2020. Transfer learning via contextual invariants for one-to-many cross-domain recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1081–1090.

[19] Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

[20] Alex Morales et al. 2020. Crowdqm: learning aspect-level user reliability and comment trustworthiness in discussion forums. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 592–605.

[21] Gerardo Ocampo Diaz and Vincent Ng. 2018. Modeling and prediction of online product review helpfulness: a survey. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, (July 2018), 698–708.

[22] Bryan Perozzi et al. 2014. Deepwalk: online learning of social representations. In *International Conference on Knowledge Discovery and Data Mining*.

[23] Mehdi Sajjadi et al. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *International Conference on Neural Information Processing Systems* (NIPS'16).

[24] Aravind Sankar et al. 2020. Beyond localized graph neural networks: an attributed motif regularization framework. *arXiv preprint arXiv:2009.05197*.

[25] Aravind Sankar et al. 2020. Beyond localized graph neural networks: an attributed motif regularization framework. *arXiv preprint arXiv:2009.05197*.

[26] Aravind Sankar et al. 2020. 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*. Houston, TX, USA, 510–518.

[27] Aravind Sankar et al. 2019. Rase: relationship aware social embedding. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[28] Michael Schlichtkrull et al. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer.

[29] Holger Schwenk and Yoshua Bengio. 2000. Boosting neural networks. *Neural Computation*, 12, 8, 1869–1887.

[30] Ming Tan et al. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

[31] Jian Tang et al. 2015. LINE: large-scale information network embedding. In *International Conference on World Wide Web, WWW*.

[32] Qiongjie Tian et al. 2013. Towards predicting the best answers in community-based question-answering services. In *International Conference on Weblogs and Social Media, ICWSM*.

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*.

[34] Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*.

[35] Xin-Jing Wang et al. 2009. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '09). ACM.

[36] Lingfei Wu et al. 2016. The role of diverse strategies in sustainable knowledge production. *PLoS ONE*.

[37] Wei Wu et al. 2018. Question condensing networks for answer selection in community question answering. In *ACL*.

[38] Rex Ying et al. 2018. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[39] Jiaxuan You et al. 2019. Position-aware graph neural networks. In *International Conference on Machine Learning*. PMLR, 7134–7143.

[40] Bing Yu et al. 2017. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR*, abs/1709.04875.

[41] Xiaodong Zhang et al. 2017. Attentive interactive neural networks for answer selection in community question answering. In *AAAI Conference on Artificial Intelligence*.

[42] W. Zhao et al. 2018. Weakly-supervised deep embedding for product review sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 30, 1, 185–197.

[43] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *World Wide Web Conference*.