Graph Neural Networks for Friend Ranking in Large-scale Social Platforms

Aravind Sankar
University of Illinois at Urbana-Champaign
asankar3@illinois.edu

Yozen Liu, Jun Yu, Neil Shah
Snap Inc.
{yliu2,jyu3,nshah}@snap.com

ABSTRACT

Graph Neural Networks (GNNs) have recently enabled substantial advances in graph learning. Despite their rich representational capacity, GNNs remain under-explored for large-scale social modeling applications. One such industrially ubiquitous application is friend suggestion: recommending users other candidate users to befriend, to improve user connectivity, retention and engagement. However, modeling such user-user interactions on large-scale social platforms poses unique challenges: such graphs often have heavy-tailed degree distributions, where a significant fraction of users are inactive and have limited structural and engagement information. Moreover, users interact with different functionalities, communicate with diverse groups, and have multifaceted interaction patterns.

We study the application of GNNs for friend suggestion, providing the first investigation of GNN design for this task, to our knowledge. To leverage the rich knowledge of in-platform actions, we formulate friend suggestion as *multi-faceted friend ranking* with multi-modal user features and link communication features. We design a neural architecture GraFRank to learn expressive user representations from multiple feature modalities and user-user interactions. Specifically, GraFRank employs *modality-specific* neighbor aggregators and *cross-modality attentions* to learn multi-faceted user representations. We conduct experiments on two multi-million user datasets from Snapchat, a leading mobile social platform, where GraFRank outperforms several state-of-the-art approaches on candidate retrieval (by 30% MRR) and ranking (by 20% MRR) tasks. Moreover, our qualitative analysis indicates notable gains for critical populations of less-active and low-degree users.

CCS CONCEPTS

• **Information systems** → *Social networking sites*; **Social networks**; **Social recommendation**; **Social networks**; • **Human-centered computing** → **Social recommendation**.

KEYWORDS

Graph Neural Network, Social Network, Recommendation System

ACM Reference Format:
Aravind Sankar and Yozen Liu, Jun Yu, Neil Shah. 2021. Graph Neural Networks for Friend Ranking in Large-scale Social Platforms. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450120

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450120

1 INTRODUCTION

Learning latent user representations has become increasingly important in advancing user understanding, with widespread adoption in various industrial settings, *e.g.*, video recommendations on YouTube [10], pin suggestions on Pinterest [50] etc. The user representations learned using deep models are *effective* at complementing or replacing conventional collaborative filtering methods [22], and are *versatile*, *e.g.*, the embeddings can be used to suggest friendships and also infer profile attributes (*e.g.*, age, gender) in social networks.

Learning latent representations of nodes in graphs has prominent applications in multiple *academic* settings, such as link prediction [53], community detection [8], and *industrial* recommender systems, including e-commerce [44, 45], content discovery [49, 50], and food delivery [24]. Graph Neural Networks (GNNs) [47] have emerged as a popular graph representation learning paradigm due to their ability to learn representations combining graph structure and node/link attributes, without relying on expensive feature engineering. GNNs can be formulated as a *message passing* framework where node representations are learned by propagating features from local graph neighborhoods via trainable neighbor aggregators. Recently, GNNs have demonstrated promising results in a few industrial systems designed for item recommendations in bipartite [50] or multipartite [49] user-to-item interaction graphs.

Despite their rich representational ability, GNNs have been relatively unexplored in large-scale *user-user social interaction* modeling applications, like friend suggestion. Recommending new potential friends to encourage users to expand their networks, is a cornerstone of social networking, and plays an important role towards user retention, and promoting engagement within the platform.

Prior efforts typically formulate friend suggestion as link prediction (or matrix completion) with a rich literature of graph-based heuristics [31] to quantify user-user affinity, *e.g.*, two users are more likely to connect if they have many common friends. A few GNN models target link prediction by learning aggregators over enclosing subgraphs around each candidate link [52–54]; such models do not scale to industry-scale social graphs with over millions of nodes and billions of edges. Still, GNNs have enormous potential for learning expressive user representations in social networks, due to their intuitive message-passing paradigm that enables attention to social influence from friends in their ego-network.

Yet, designing GNNs for friend recommendations in large-scale social platforms poses unique challenges. First, social networks are characterized by *heavy-tailed* degree distributions, *e.g.*, many networks approximately follow power-law distributions [3]. This poses a key challenge of *limited structural information* for a significant proportion of users with very few friends. A related challenge is *activity sparsity* where a very small fraction of users actively form new friendships at any given time. Second, modern social platforms

offer a multitude of avenues for users to interact, *e.g.*, users can communicate with friends either by *directly* exchanging messages and pictures, or through *indirect* social actions by liking and commenting on posts. Extracting knowledge from such *heterogeneous* in-platform user actions is challenging, yet extremely *valuable* to address sparsity challenges for a vast majority of inactive users.

Present Work: We overcome structural and interactional sparsity by exploiting the rich knowledge of heterogeneous in-platform actions. We formulate friend recommendation on social networks as *multi-faceted friend ranking* on an *evolving* friendship graph, with multi-modal user features and link communication features (Figure 1). We represent users with heterogeneous feature sets spanning multiple *modalities*, that include a collection of *static* profile attributes (*e.g.*, demographic information) and *time-sensitive* in-platform activities (*e.g.*, content interests and interactions). We also leverage pairwise link features on existing friendships, which capture recent communication activities across multiple direct (*e.g.*, messages) and indirect (*e.g.*, stories) channels within the platform.

To understand the complexity of user interactions and gain insights into various factors impacting friendship formation, we conduct an empirical analysis to investigate *attribute homophily* with respect to different user feature modalities. Our analysis reveals *diverse homophily distributions* across modalities and users, and indicates non-trivial *cross-modality* correlations. Motivated by these observations, we design an end-to-end GNN architecture, GraFRank (Graph Attentional Friend Ranker) for multi-faceted friend ranking.

GraFRank learns user representations by *modality-specific* neighbor aggregation and *cross-modality* attention. We handle heterogeneity in modality homophily by learning modality-specific neighbor aggregators to compute a set of representations for each user; the aggregator is modeled by friendship attentions to capture the influence of individual features and pairwise communications. We introduce a *cross-modality* attention module to compute the final user representation by attending over the modality-specific representations of each user, thereby learning non-linear correlations across modalities. We summarize our key contributions below:

- **Graph-Neural Friend Ranking**: To our knowledge, ours is the first work to investigate *graph neural network* usage and design for social user-user interaction modeling applications. Unlike prior work that typically view friend recommendation as structural link prediction, we present a novel formulation with multi-modal user features and link features, to leverage knowledge of rich heterogeneous user activities in social networking platforms.
- **GraFRank Model:** Motivated by our empirical study that reveals heterogeneity in modality homophily and cross-modality correlations, we design a neural architecture, GraFRank, to learn *multi-faceted* user representations. Distinct from conventional GNNs operating on a single feature space, GraFRank learns from multiple feature modalities and user-user interactions.
- **Robust Experimental Results:** Our extensive experiments on two large-scale datasets from a popular social networking platform Snapchat, indicate significant gains for GraFRank over state-of-the-art baselines on friend candidate retrieval (relative MRR gains of 30%) and ranking (relative MRR gains of 20%) tasks. Our qualitative analysis indicates stronger gains for a large, but especially crucial population of *less-active* and *low-degree* users.

2 RELATED WORK

We briefly review a few related lines of work on friend recommendations, graph neural networks, and multi-modal learning.

Friend Recommendation: The earliest methods were carefully designed *graph-based heuristics* of user-user proximity in social networks [31], *e.g.*, path-based Katz centrality [26] or common neighbor-based Adamic/Adar [1]. Supervised learning techniques exploited a collection of such pairwise features to train ranking models [12, 34]. However, extracting heuristic features *on-the-fly* for each potential link is *infeasible* in large-scale evolving networks.

Recently, *graph embedding* methods learn latent node representations to capture the structural properties of a node and its neighborhoods [11], *e.g.*, popular embedding models like node2vec [16] and Deepwalk [35] learn unsupervised embeddings to maximize the likelihood of co-occurrence in fixed-length random walks, and have shown effective link prediction performance. Since graph embedding methods learn latent embeddings per node, the number of *model parameters* scales with the *size of the graph*, which is *prohibitive* for large-scale networks with over multi-million users.

A related direction is social recommendation [13, 29, 38], which utilizes the social network as an auxiliary data source to model user behavior in social platforms [39, 42, 48] and improve quality of item recommendations to users. In contrast, our problem, friend suggestion is a user-user recommendation task that is complementary to social recommendation, since it facilitates creating a better social network of users to benefit social recommendation.

Graph Neural Networks: GNNs learn node representations by recursively propagating features (*i.e.*, message passing) from local neighborhoods through the use of aggregation and activation functions [36, 47]. Graph Convolutional Networks (GCNs) [28] learn degree-weighted aggregators by operating on the graph Laplacian. Many models generalize GCN with a variety of learnable aggregators, *e.g.*, self-attentions [43], mean and max pooling functions [17, 18]; these approaches have consistently outperformed embedding techniques based upon random walks [16, 35]. In contrast to most GNN models that store the entire graph in GPU memory, GraphSAGE [17] is an inductive variant that reduces memory footprint by sampling a fixed-size set of neighbors in each GNN layer. A few scalable extensions include minibatch training with variance reduction [6, 23], subgraph sampling [51], and graph clustering [9].

Despite the successes of GNNs, *very few* industrial systems have developed large-scale GNN implementations. One recent system, PinSage [50] extends GraphSAGE to user-item bipartite graphs in Pinterest; MultiSage [49] extends PinSage to multipartite graphs.

However, GNNs remain unexplored for large-scale *user-user* social modeling applications where users exhibit multifaceted behaviors by interacting with different functionalities on social platforms. In our work, we design GNNs for the important application of *friend suggestion*, through a *novel multi-faceted friend ranking* formulation with *multi-modal user features* and *link communication features*.

Multi-Modal Learning: Deep learning techniques have been explored for multi-modal feature fusion over diverse modalities such as text, images, video, and graphs [14, 25]. Specifically, multi-modal extensions of GNNs have recently been examined in micro-video recommendation [46] and urban computing [15] applications. Unlike prior work that regard modalities as largely *independent*

data sources, user feature modalities in social networks tend to be correlated. In this work, we model non-linear cross-modality correlations to learn *multi-faceted* user representations.

3 PRELIMINARIES

We first formulate the problem of *multi-faceted friend ranking* in large-scale social platforms (Section 3.1) and then briefly introduce relevant background on *graph neural networks* (Section 3.2).

3.1 Problem Formulation

In this section, we introduce the different information sources in a social platform that are relevant to friend suggestion. Each individual in the platform is denoted by a *user* u or v and a pair of users (u, v) may be connected by a *friendship*, which is an undirected relationship, *i.e.*, if u is a friend of v, v is also a friend of u. We assume a set of users $\mathcal{V}$ introduced until our latest observation time of the platform. The friendship graph $\mathcal{G}$ evolves when new friendships form and when new users join the platform. Here, we only consider the emergence of new users and friendships while leaving the removal of existing users and edges as future work.

Prior work typically represent a dynamic network as a sequence of static snapshots, primarily due to scaling concerns. However, graph snapshots are coarse approximations of the actual continuous-time network and rely on a user-specified discrete time interval for snapshot creation [40, 41]. We also assume multiple time-aware user-level features (across modalities) and link(edge)-level features capturing pairwise user-user communications. In industrial settings, such features are commonly extracted by routine batch jobs and populated in an upstream database at regular time intervals (*e.g.*, daily batch inference jobs), to facilitate efficient model training.

Thus, we adopt a hybrid data model that achieves the best of both worlds. We formulate the *friendship graph* as a continuous-time dynamic graph (CTDG) with the expressivity to record friendships at the finest possible temporal granularity; and represent *features* as a sequence of daily snapshots where the time-sensitive features (*e.g.*, engagement activity) are recorded at different time scales.

Friendship Graph: Let us consider an observation time window (t_s, t_e) such that friendships created in this window specify the training data for the friend ranking model. We divide this window (t_s, t_e) into a sequence of S daily snapshots, denoted by $1, 2, \ldots, S$. Formally, we model the friendship graph $\mathcal{G}$ as a timed sequence of friend creation events over the entire time range $(0, t_e)$, defined as:

Definition 3.1 (Friendship Graph). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, let $\mathcal{V}$ be the set of users, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathbb{R}$ be the set of friendship links between users in $\mathcal{G}$. At the finest granularity, each link $e = (u, v, t) \in \mathcal{E}$ is assigned a unique timestamp $t \in \mathbb{R}^+; 0 < t < t_e$ that denotes the link creation time, and $\mathcal{T} : \mathbb{R}^+ \mapsto [0, S]$ is a function that maps each timestamp t to a corresponding snapshot in $[0, S]$.

Here, the window (t_s, t_e) corresponds to snapshots $[1, S]$ and snapshot 0 is a placeholder for any $t < t_s$, while the graph $\mathcal{G}$ includes *all* friendships with time-stamped links in $(0, t_e)$. The set of temporal neighbors of user v at time t includes friends created before t, defined as $N_t(v) = \{w : e = (v, w, t') \in \mathcal{E} \wedge t' < t\}$.

Multi-Modal Evolving User Features: In a social platform, users typically use multiple functionalities, such as posting videos, exchanging messages with friends, or liking and sharing posts, which are indicative of their stable traits and mutable interests. We extract user attributes spanning a total of $K = 4$ modalities, which include profile attributes, in-app interests, friend creation activities, and user engagement activities, described in detail below:

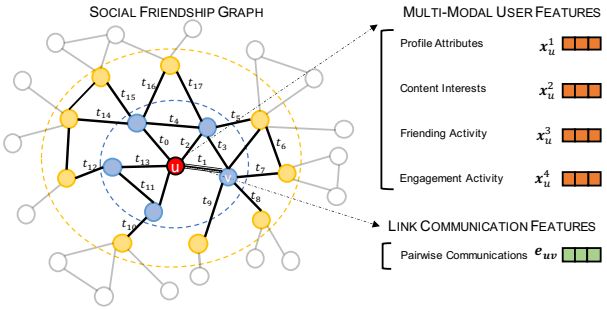

Figure 1: Desiderata for Multi-faceted Friend Ranking: temporally evolving friendship graph with multi-modal user features and pairwise link communication features.

- *Profile Attributes*: a set of (mostly) static demographic features describing the user, including age, gender, recent locations, languages, etc., that are listed or inferred from their profile.
- *Content Interests*: a real-valued feature vector describing the textual content (*e.g.*, posts, stories) interacted by the user within the platform, *e.g.*, topics of stories viewed by the user on Snapchat.
- *Friending Activity*: aggregated number of sent/received friend requests, reciprocated friendships, and viewed suggestions of the user in different time ranges (*e.g.*, daily, weekly, and monthly).
- *Engagement Activity*: aggregated number of in-app direct and indirect engagements for the user (*e.g.*, text messages, snaps, and comments on posts) with other friends in different time ranges.

The user feature modalities include a combination of *static* and *time-sensitive* features, *i.e.*, the profile attributes are static while the rest of the modalities are time-sensitive and often evolve at different scales across users, *e.g.*, a long-time active user may frequently communicate with a stable set of friends, while a new user is more likely to quickly add new friends before communicating.

The feature vector of a user $u \in \mathcal{V}$ in snapshot $s \in [1, S]$ is defined by $\mathbf{x}_v^s = [\mathbf{x}_v^{s,1}, \ldots, \mathbf{x}_v^{s,K}]$ where $\mathbf{x}_v^{s,k} \in \mathbb{R}_k^D$ is the k-th user feature modality and $[\cdot]$ denotes row-wise concatenation.

Pairwise Link Communication Features: Social networks comprise two predominant types of communication channels: *conversations* and *social actions*. Conversations include exchanges of text messages and media content with friends, which indicate direct user-user communications. In contrast, social actions facilitate indirect user-user interactions, *e.g.*, posting a Snapchat Story or liking a Facebook post results in a passive broadcast to friends.

For conversational channels, we extract bidirectional link features reflecting the number of communications sent and received by each pair of users (who are friends). We capture indirect social actions by recording the number of actions of each type per friend. Similar to user features, we extract link features per snapshot by aggregating communications at different time intervals. The link feature vector for a pair of users (u, v) at time t (who became friends before t), is denoted by $\mathbf{e}_{uv}^s \in \mathbb{R}^E$ where $s = \mathcal{T}(t)$ is the snapshot associated with time t and E is the cardinality of link features.

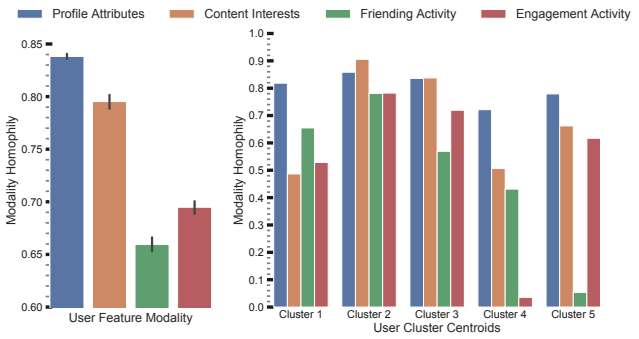

Figure 2: Users exhibit different extents of homophily across feature modalities. (a) Overall modality homophily scores, with 95% confidence interval bands (b) five representative cluster centroids identified by clustering users based on their homophily distributions over the K modalities.

We formally define the problem of *multi-faceted friend ranking* in large-scale social platforms, over friendship graph $\mathcal{G}$ with multi-modal user features and pairwise link features, as follows:

Problem (Multi-Faceted Friend Ranking). *Leverage multi-modal user features* $\{\mathbf{x}_v^s : v \in \mathcal{V}, 1 \leq s \leq S\}$, *link features* $\{\mathbf{e}_{uv}^s : s = \mathcal{T}(t), (u, v, t) \in \mathcal{E}\}$ *and friendship graph* $\mathcal{G}$, *to generate user representations* $\{\mathbf{h}_v(t) \in \mathbb{R}^D : v \in \mathcal{V}\}$ *at time* t, *that facilitate friend suggestion tasks of candidate retrieval and re-ranking.*

3.2 Background on GNNs

We briefly introduce a generic formulation of a graph neural network layer with message-passing for neighborhood aggregation.

GNNs use multiple layers to learn node representations. At each layer $l > 0$ ($l = 0$ is the input layer), GNNs compute a representation for node u by aggregating features from its neighborhood, through a learnable aggregator $F_{\theta,l}$ per layer. Stacking k layers allows the k-hop neighborhood of a node to influence its representation.

$$\mathbf{h}_{u,l} = F_{\theta,l}\Big(\mathbf{h}_{u,l-1}, \{\mathbf{h}_{v,l-1}\}\Big), \quad v \in N(u) \tag{1}$$

Equation (1) indicates that the embedding $\mathbf{h}_{u,l} \in \mathbb{R}^D$ for node u at the l-th layer is a non-linear aggregation of its embedding $\mathbf{h}_{u,l-1}$ from layer $l-1$ and the embeddings of its immediate neighbors $v \in \mathcal{N}(u)$. The function $F_{\theta,l}$ defines the message-passing function at layer l and can be instantiated using a variety of aggregators, including graph convolution [28], attention [43], and pooling [17]. The node representation for u at the input layer is $\mathbf{h}_{u,0}$, where $\mathbf{h}_{u,0} = \mathbf{x}_u \in \mathbb{R}^D$. The representation of node u at the final GNN layer is typically trained using a supervised learning objective.

The above formulation (Equation 1) operates under the assumption of a static graph and a single static input feature vector per node. In contrast, our setting involves a time-evolving friendship graph $\mathcal{G}$ with pairwise link features and multi-modal node features.

4 GRAPH NEURAL FRIEND RANKING

In this section, we present our approach to inductively learn user representations in a dynamic friendship graph with pairwise link features and time-sensitive multi-modal node features.

We first conduct an empirical analysis on user feature modalities, to gain insights into various factors impacting friendship formation (Section 4.1). We then formulate the design choices of our model GraFRank for friend ranking based on our acquired insights (Section 4.2), followed by model training details (Section 4.3)

4.1 Motivating Insight: Modality Analysis

We conduct an empirical study that helps us formulate the design choices in our model. We aim to validate the existence and understand the extent and variance of attribute homophily with respect to the different user feature modalities. We begin by analyzing users' ego-networks to characterize *modality homophily*, both overall and broken-down across different user segments. The definition of *modality homophily* echoes the standard definition of attribute homophily [33], but generalized to include a modality of attributes, *i.e.,* the tendency of users in a social graph to associate with others who are *similar* to them along attributes of a certain modality.

We define a homophily measure $\mathbf{m}_{uv}^k$ between a user u and her friend v on modality k by the standard cosine similarity [2], which is a normalized metric that accounts for heterogeneous activity across users. We compute a modality homophily score $\mathbf{m}_u^k$ for user u on modality k by the mean over all her neighbors, defined by:

$$\mathbf{m}_u^k = \frac{1}{|N_u|} \sum_{v \in N_u} \mathbf{m}_{uv}^k \qquad \mathbf{m}_{uv}^k = cos(\mathbf{x}_u^k, \mathbf{x}_v^k) \tag{2}$$

Note that we omit the snapshot s above since the discussion is restricted to a single feature snapshot. Figure 2 (a) shows the overall modality homophily scores (averaged across all users), for each of the K modalities. We observe differing extents of attribute homophily across modalities, with higher variance for the time-sensitive modalities (*e.g.*, friending and engagement activities).

We further extend our analysis to examine the *homophily distribution* over modalities at the granularity of individual users, to understand if modality homophily varies across different users. We first represent each user u by a K-dimensional *modality vector* $\mathbf{m}_u = [\mathbf{m}_u^1, \ldots, \mathbf{m}_u^K]$ that describes the homophily distribution over K modalities. We then use k-means [19] to cluster the user set $\mathcal{V}$ based on their modality vectors. Figure 2 (b) shows the centroids of five representative clusters. We observe stark differences in the modality vector centroids across the five clusters, indicating the existence of user segments with diverse homophily distributions over the K modalities. This motivates our first key observation:

Observation 1 (Heterogeneity in Modality Homophily). *Users exhibit different extents of homophily across modalities, and the homophily distribution over modalities varies across user segments.*

Each modality enables identification of a subset of friends that exhibit modality homophily. However, this poses a question: do the K modalities induce the same (or disparate) subsets of homophilous friends, or are the friends that exhibit homophily in each modality, correlated? We now investigate this relationship in this section.

For every modality k, we cluster the ego-network (set of direct friends) $N(u)$ of each user u, which is represented as a set of modality-specific vectors $\{\mathbf{x}_v^k \in \mathbb{R}^{D_k} : v \in N(u)\}$; this results in ego-clustering assignments per modality. To quantify *cross-modality*

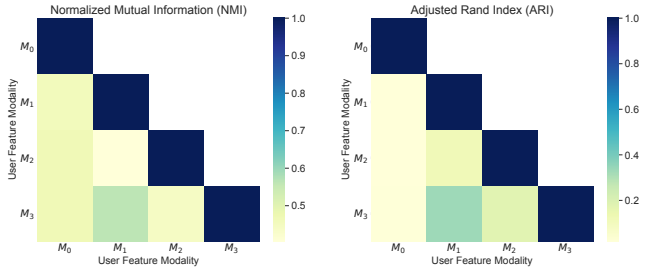

Figure 3: Cross-modality Correlation Study: NMI (a) and ARI (b) metrics for each pair of modalities, quantifying pairwise correlation by consensus in ego-clustering assignments (obtained independently with respect to each modality). We observe substantial correlations across pairs of modalities.

correlations, we compute a *correlation* score for each pair of modalities by the *consensus* between their ego-clustering assignments.

We use two standard measures: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) to evaluate consensus between clusterings [4]. NMI measures the statistical correlation between two clustering assignments; however, NMI increases with the number of distinct clusters. ARI measures the percentage of correct pairwise assignments, and is chance-corrected with an expected value of zero. Note that NMI and ARI are symmetric metrics.

Figure 3 depicts average NMI and ARI scores for each pair of feature modalities. We observe substantial correlation in cluster assignments across a few modalities (*e.g.*, time-sensitive modalities M_1 and M_3) while some (*e.g.*, static modality M_0) are quite distinct from the rest. Our key takeaway regarding modality correlation is:

Observation 2 (Cross-Modality Correlation). *Non-trivial correlations exist between pairs of feature modalities, as indicated by the consensus in their induced clusterings of ego-networks.*

4.2 GraFRank: Multi-Faceted Friend Ranking

In this section, we first introduce the key components of our model GraFRank (Graph Attention Friend Ranker) for friend ranking. Our modeling choices in designing a multi-modal GNN, directly follow from our observations. GraFRank has two modules (Figure 5):

- Modality-specific neighbor aggregation.
- Cross-modality attention layer.

Below, we present a detailed description of each module.

4.2.1 ***Modality-specific Neighbor Aggregation.*** Since the different modalities vary in the extent of induced homophily (Observation 1), we treat each modality individually as opposed to the popular choice of combining all features by concatenation. Thus, we learn a modality-specific representation $\mathbf{z}_u^k(t) \in \mathbb{R}^D$ for each user $u \in \mathcal{V}$ at time $t \in \mathbb{R}^+$, that encapsulates information from modality k. Each user u flexibly prioritizes different friends in her temporal neighborhood $N_t(u)$, thereby accounting for the variance in homophily distribution across user segments.

We design a *modality-specific neighbor aggregation* module to compute K representations $\{\mathbf{z}_u^1(t), \ldots, \mathbf{z}_u^K(t)\}, \mathbf{z}_u^k(t) \in \mathbb{R}^D$ for each user $u \in \mathcal{V}$ at time $t \in \mathbb{R}^+$, where each $\mathbf{z}_u^k(t)$ is obtained using an independent and unique message-passing function per modality.

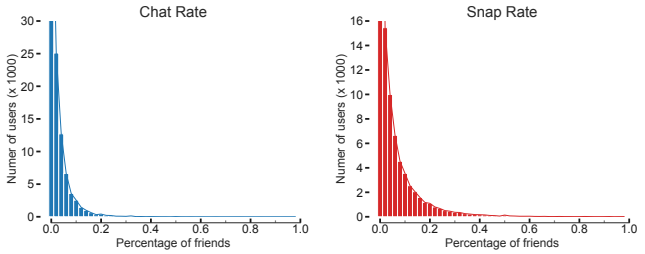

Figure 4: Friend communication rate in two direct channels (Chat, Snap) on Snapchat. Most users communicate frequently only with a subset ($\leq$ 20%) of their friends, making influence modeling critical during neighbor aggregation.

We begin by describing a single layer, which consists of two major operations: *message propagation* and *message aggregation*. We subsequently discuss generalization to multiple successive layers.

Message Propagation: We define the message-passing mechanism to aggregate information from the ego-network $N_t(u)$ of user u at time t. Specifically, the propagation step for modality k aggregates the k-th modality features $\{\mathbf{x}_v^{s,k} : v \in N_t(u), s = \mathcal{T}(t)\}$ from the corresponding snapshot $s = \mathcal{T}(t)$ of temporal neighbors $N_t(u)$. To quantify the importance of each friend v in the ego-network, we propose a friendship attention [37, 43] which takes embeddings $\mathbf{x}_u^{s,k}$ and $\mathbf{x}_v^{s,k}$ as input, and computes an attentional coefficient $\alpha^k(u, v, t)$ to control the influence of friend v on user u at time t, given by:

$$\alpha^k(u,v,t) = \text{LeakyRELU}\Big(\mathbf{a}_k^T\big(\boldsymbol{W}_1^k\mathbf{x}_u^{s,k} \,||\, \boldsymbol{W}_1^k\mathbf{x}_v^{s,k}\big)\Big) \quad s = \mathcal{T}(t) \quad (3)$$

where $\boldsymbol{W}_1^k \in \mathbb{R}^{D_k \times D}$ is a shared linear transformation applied to each user, $||$ is the concatenation operation, and the friendship attention is modeled as a single feed-forward layer parameterized by weight vector $\mathbf{a}_k \in \mathbb{R}^{2D}$ followed by the LeakyReLU nonlinearity. We then normalize the attentional coefficients across all friends connected with u at time t by adopting the softmax function:

$$\alpha^k(u,v,t) = \frac{\exp\big(\alpha^k(u,v,t)\big)}{\sum_{w \in N_t(u)} \exp\big(\alpha^k(u,w,t)\big)} \quad (4)$$

We now define an ego-network representation $\mathbf{z}_u^k\big(t, N_u(t)\big) \in \mathbb{R}^{D_k}$ for user u in modality k that captures messages propagated from first-order neighbors in the ego-network $N_u(t)$. The message $\mathbf{m}_{u \leftarrow v}^k \in \mathbb{R}^D$ propagated from friend v to user u at time t is defined as the transformed friend embedding, *i.e.*, $\mathbf{m}_{u \leftarrow v}^k = \boldsymbol{W}_1^k\mathbf{x}_v^{s,k}$. We then compute $\mathbf{z}_u^k\big(t, N_u(t)\big)$ through a weighted average of message embeddings from each friend $v \in N_u(t)$ and guided by normalized friendship weights $\alpha^k(u, v, t)$, which is defined as:

$$\mathbf{z}_u^k\big(t, N_u(t)\big) = \sum_{v \in N_u(t)} \alpha^k(u,v,t)\mathbf{m}_{u \leftarrow v}^k \quad (5)$$

In the above equation, the friendship weights are learnt merely from the connectivity of the ego-network. In reality, most users have very few close friends, and users with many friends only frequently communicate with a few of them. We empirically validate this by examining *friend communication rate*, defined by the percentage of friends that a user has communicated with at least once (directly sent a Chat/Snap with on Snapchat) in a one-month window. From

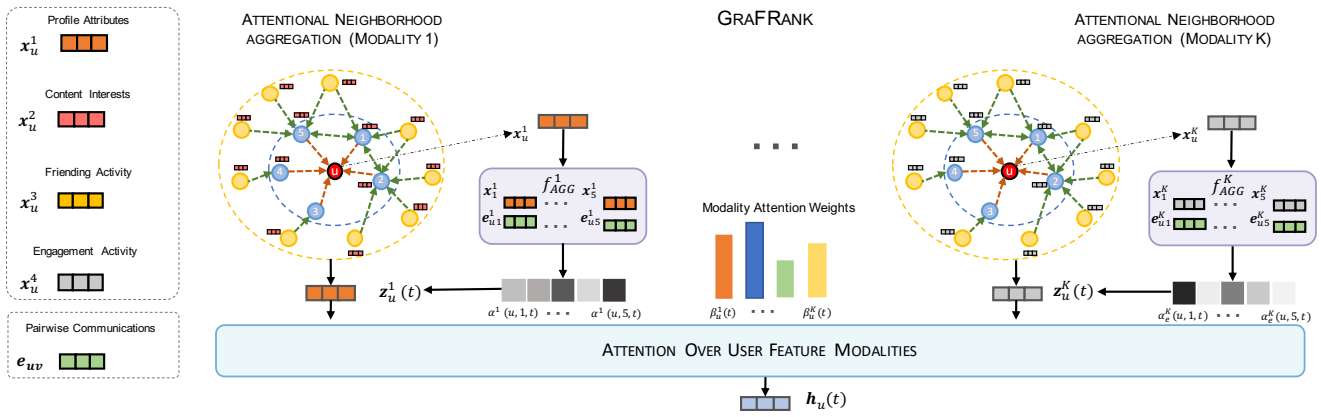

Figure 5: Overall framework of GRAFRANK: a set of K modality-specific neighbor aggregators (parameterized by individual modality-specific user features and link communication features) to compute K intermediate user representations; cross-modality attention layer to compute final user representations by capturing discriminative facets of each modality.

Figure 4, we find that a vast majority of users communicate with a small percentage (10-20%) of their friends; thus, we posit that *friendship activeness* is critical to precisely model user affinity.

Towards this goal, we incorporate *pairwise link communication* features to parameterize both the attentional coefficients and the message aggregated from friends in the ego-network. Specifically, we formulate the message $\mathbf{m}^k_{u\leftarrow v} \in \mathbb{R}^D$ from friend v to user u at time t as a function of both friend feature $\mathbf{x}^{s,k}_v$ and link feature $\mathbf{e}^s_{uv}$.

$$\mathbf{m}^k_{u\leftarrow v} = \boldsymbol{W}^k_2 \mathbf{x}^{s,k}_v + \boldsymbol{W}^k_e \mathbf{e}^s_{uv} + \mathbf{b} \qquad s = \mathcal{T}(t) \tag{6}$$

where $\boldsymbol{W}^k_2 \in \mathbb{R}^{D_k \times D}, \boldsymbol{W}^k_e \in \mathbb{R}^{E\times D}$ are trainable weight matrices operating on the user and link features respectively, and $\mathbf{b} \in \mathbb{R}^D$ is a learnable bias vector. The attentional co-efficient $\alpha^k(u,v,t)$ is then computed as a function of the user feature $\mathbf{x}^{s,k}_u$ and message embedding $\mathbf{m}^k_{u\leftarrow v} \in \mathbb{R}^D$ from friend v to user u, defined by:

$$\alpha^k(u,v,t) = \sigma\Big(\mathbf{a}^T_k\Big(\boldsymbol{W}^k_1 \mathbf{x}^{s,k}_u \;||\; \mathbf{m}^k_{u\leftarrow v}\Big)\Big) \qquad s = \mathcal{T}(t) \tag{7}$$

where σ is a non-linearity such as LeakyRELU. We similarly normalize the attentional coefficients $\alpha^k(u,v,t)$ using Equation 4 and compute the ego-network representation $\mathbf{z}^k_u(t, N_u(t))$ for user u on modality k using Equation 5 with the message embedding $\mathbf{m}^k_{u\leftarrow v}$ from Equation 6 conditioned on the link features.

Distinct from conventional GNNs [17, 28, 43, 50] that only consider user $\mathbf{x}^{s,k}_u$ and friend $\mathbf{x}^{s,k}_v$ features to parameterize the neighbor aggregation, we additionally incorporate the link features $\mathbf{e}^s_{uv}$ into the attentional co-efficient $\alpha^k(u,v,t)$ (Equation 7) and the message $\mathbf{m}^k_{u\leftarrow v}$ (Equation 6) passed from friend v; this encourages the aggregation to be cognizant of pairwise communications with friends, *e.g.*, passing more messages from the active friendships. Empirically, we observe a boost in friend ranking performance (Section 5.4) due to our communication-aware message passing strategy.

Message Aggregation: We refine the representation of user u by aggregating the messages propagated from friends in $N_u(t)$. In addition, we consider self-connections $\mathbf{m}^k_{u\leftarrow u} = \boldsymbol{W}^k_1 \mathbf{x}^{s,k}_u$ to retain knowledge of the original features ($\boldsymbol{W}_1$ is the same transformation used in Equation 3). Specifically, we concatenate the ego-network and self-representations of user u, and further transform the concatenated embedding through a dense layer F^k_θ, defined by:

$$\mathbf{z}^k_u(t) = F^k_\theta\Big(\mathbf{m}^k_{u\leftarrow u}, \mathbf{z}^k_u\big(t, N_u(t)\big)\Big) \tag{8}$$

$$= \sigma\Big(\boldsymbol{W}^k_a\Big(\mathbf{z}^k_u(t, N_u(t)) \;||\; \mathbf{m}^k_{u\leftarrow u}\Big) + \mathbf{b}_a\Big) \tag{9}$$

where $\boldsymbol{W}^k_a \in \mathbb{R}^{D\times D}$, $\mathbf{b}_a \in \mathbb{R}^D$ are trainable parameters of the aggregator and σ denotes the ELU activation function which allows messages to encode both positive and small negative signals. Empirically, we observe significant improvements due to self-connections, compared to directly using the propagated ego-network representation from the neighborhood, as in GCN [28], and GAT [43].

Higher-order Propagation: We stack multiple neighbor aggregation layers to model high-order connectivity information, *i.e.*, propagate features from l-hop neighbors. The inputs to layer l depend on the user representations output from layer $(l-1)$ where the initial (*i.e.*, "layer 0") representations are set to the input user features in modality k. By stacking l layers, we recursively formulate the representation $\mathbf{z}^k_{u,l}$ of user u at the end of layer l by:

$$\mathbf{z}^k_{u,l} = F^k_{\theta,l}\Big(\mathbf{m}^k_{u\leftarrow u,l-1}, z^k_{u,l-1}\big(t, N_u(t)\big)\Big) \quad \mathbf{m}^k_{u\leftarrow u,l-1} = z^k_{u,l-1} \tag{10}$$

where $z^k_{u,l-1}$ is the representation of user u in modality k after $(l-1)$ layers and $z^k_{u,l-1}(t, N_v(t))$ denotes the $(l-1)$-ego-network representation of user u. We apply L neighbor aggregation layers to generate the layer-L representation $\mathbf{z}^k_{u,L}$ of user u in modality k.

4.2.2 ***Cross Modality Attention.*** To learn complex non-linear correlations between different feature modalities, we design a *cross-modality attention* mechanism. Specifically, we learn modality attention weights $\beta^k_u(t)$ to distinguish the influence of each modality k using a two-layer Multi-Layer Perceptron, defined by:

$$\beta^k_u(t) = \frac{\exp\left(\mathbf{a}^T_m \boldsymbol{W}_m \mathbf{z}^k_{u,L} + b_m\right)}{\sum_{k'=1}^K \exp\left(\mathbf{a}^T_m \boldsymbol{W}_m \mathbf{z}^{k'}_{u,L} + b_m\right)} \tag{11}$$

with weights $\boldsymbol{W}_m \in \mathbb{R}^{D\times D}, \mathbf{a}_m \in \mathbb{R}^D$ and scalar bias b_m. The final representation $\mathbf{h}_u(t) \in \mathbb{R}^D$ of user u is computed by a weighted aggregation of the layer-L modality-specific user representations $\{\mathbf{z}^1_{u,L}, \ldots, \mathbf{z}^K_{u,L}\}$, guided by modality weights $\beta^k_u(t)$, defined by:

$$\mathbf{h}_u(t) = \sum_{k=1}^{K} \beta^k_u(t)\ \boldsymbol{W}_m \mathbf{z}^k_{u,l} \tag{12}$$

4.3 Temporal Model Training

We train GraFRank using a temporal pairwise ranking objective to differentiate *positive* and *negative* neighborhoods. We assume access to training data described by a set of timestamped links $\mathcal{L}$ created in a window (t_s, t_e), where $(u, v, t) \in \mathcal{L}$ is a bi-directional friendship link between *source* user u and *target* friend v formed at time $t \in (t_s, t_e)$. To train the parameters of GraFRank, we define a triplet-like learning objective based on max-margin ranking.

4.3.1 ***Pairwise Ranking Objective.*** We define a time-sensitive ranking loss over the final user embeddings ($\mathbf{h}_u(t)$ for user u at time t) to rank the inner product of positive links $(u, v, t) \in \mathcal{L}$, higher than sampled negatives (u, n, t) by a margin factor Δ, as:

$$L = \sum_{(u,v,t)\in\mathcal{L}} \mathbb{E}_{n\sim P_n(u)} \max\{0, \mathbf{h}_u(t)\cdot\mathbf{h}_n(t) - \mathbf{h}_u(t)\cdot\mathbf{h}_v(t) + \Delta\} \tag{13}$$

where Δ is a margin hyper-parameter and $P_n(u)$ is the negative sampling distribution for user u. Here, we use a single forward pass to inductively compute a time-aware representation $\mathbf{h}_u(t)$ for each user $u \in \mathcal{V}$ at time t based on the appropriate user and link features in temporal neighborhoods. Each minibatch of training examples is then optimized independently which precludes the need to explicitly model temporal dependencies. This generic contrastive learning formulation enables usage of the same framework for different recommendation tasks such as candidate retrieval and ranking, with different negative sampling distributions.

4.3.2 ***Candidate Retrieval and Ranking Tasks.*** We learn user embeddings towards two key use-cases in friend ranking: *candidate retrieval* and *candidate ranking*. Candidate retrieval aims to generate a list of top-N (*e.g.*, $N = 100$) potential friend suggestions out of a very large candidate pool (over millions of users), while candidate ranking involves fine-grained re-ranking within a much smaller pool of the generated candidates, to determine the top-n (*e.g.*, $n = 10$) suggestions shown to end users in the platform. We define different negative sampling distributions $P_n(u)$ for each task owing to their different ranking granularities, as follows:

- *Candidate Retrieval*: For the *coarse-grained* task of candidate retrieval, we uniformly sample five *random negative* users for each positive link, from the entire user set $\mathcal{V}$. Generating random negatives is efficient and effective at quickly training the model to identify potential friend candidates for each user. However, random negatives are often too *easy* to distinguish and may not provide the requisite resolution for the model to learn *fine-grained distinctions* necessary for candidate friend ranking.
- *Candidate Ranking*: To enhance model resolution for candidate ranking, we also use *hard negative* examples; for each positive pair (u, v), we generate five hard negatives related to the source u, but not as relevant as the target friend v. For a 2-layer GNN ($L = 2$), we randomly choose users in the 3-4 hop neighborhood of u as hard negatives; 2-hop neighbors are excluded since such *friends of friends* are expected to be relevant suggestion candidates owing to triadic closure in social networks. In practice, we pre-compute hard negative examples to facilitate efficient model training.

  We adopt a *two-phase* learning approach for candidate ranking. We pre-train the model on random negatives (as in candidate retrieval), to identify good model initialization points, followed by fine-tuning on hard negatives. Ranking hard negatives is more challenging, hence encouraging the model to progressively learn friend distinctions at a finer granularity. We empirically show notable gains on candidate ranking due to our two-phase strategy, compared to training individually on random or hard negatives.

Dataset	Region 1	Region 2
# users	3.1 M	17.1 M
# links	286 M	2.36 B
# user features	79	79
# link features	6	6
# test set friend requests	46K	340K

Table 1: Dataset statistics

4.3.3 ***Temporal Neighborhood Sampling.*** We learn a temporal user representation $\mathbf{h}_u(t)$ for user u at time t by selecting a fixed number of friends from $N_u(t)$ for neighbor aggregation in each layer; this controls the memory footprint during training [17].

To efficiently identify and sample neighbors of u at any time t, we represent the time-evolving friendship graph $\mathcal{G}$ as a *temporal adjacency list* at its latest time t_s where each user u has a list of (friend, time) tuples sorted by link creation times. This data representation enables $O(\log d)$ neighbor lookup at an arbitrary timestamp t via binary search where d is the average user degree in the graph.

4.3.4 ***Multi-GPU Minibatch Training.*** We train GraFRank using minibatches of links from $\mathcal{L}$ over multiple GPUs on a single shared memory machine. The temporal adjacency list of $\mathcal{G}$ and feature matrices $\boldsymbol{X}, \boldsymbol{E}$ are placed in shared CPU memory to enable fast parallel neighborhood sampling and feature lookup. We adopt a *producer-consumer* framework [50] that alternates between CPUs and GPUs for model training. A CPU-bound producer constructs friend neighborhoods, looks up user and link features, and generates negative samples for the links of a minibatch. We then partition each minibatch across multiple GPUs, to compute forward passes and gradients with a PyTorch model over dynamically constructed computation graphs. The gradients from different GPUs are synchronized using PyTorch's Distributed Data Parallel [30] construct.

5 EXPERIMENTS

To analyze the quality of user representations learned by GraFRank, we propose five research questions to guide our experiments:

- ($\mathbf{RQ_1}$) Can GraFRank outperform feature-based models and state-of-the-art GNNs on candidate *retrieval* and *ranking* tasks?
- ($\mathbf{RQ_2}$) How does GraFRank compare to prior work under alternative metrics of *reciprocated* and *communicated* friendships?
- ($\mathbf{RQ_3}$) How do the different *architectural* design choices and *training* strategies in GraFRank impact performance?

Dataset	Region 1					Region 2				
Metric	**N@5**	**N@50**	**HR@5**	**HR@50**	**MRR**	**N@5**	**N@50**	**HR@5**	**HR@50**	**MRR**
LogReg	0.1752	0.2460	0.2452	0.5262	0.1751	0.0761	0.1367	0.1134	0.3654	0.0831
MLP	0.1923	0.2679	0.2721	0.5726	0.1903	0.0973	0.1720	0.1466	0.4541	0.1046
XGBoost	0.2099	0.2865	0.2932	0.5957	0.2071	0.1366	0.2097	0.1936	0.4921	0.1409
GCN	0.0934	0.1836	0.1490	0.5154	0.1034	0.1651	0.2634	0.2503	0.6427	0.1678
GAT	0.0851	0.1813	0.1424	0.5352	0.0960	0.1797	0.2794	0.2698	0.6663	0.1812
SAGE + Max	0.1790	0.2736	0.2695	0.6409	0.1797	0.1520	0.2505	0.2315	0.6269	0.1566
SAGE + Mean	0.2378	0.3240	0.3338	0.6757	0.2333	0.2870	0.3805	0.4005	0.7655	0.2790
GraFRank	**0.3152**	**0.3983**	**0.4318**	**0.7533**	**0.3035**	**0.4166**	**0.4950**	**0.5386**	**0.8395**	**0.4012**
Percentage Gains	32.55 %	22.93 %	29.36 %	11.48 %	30.09 %	45.16 %	30.09 %	34.48 %	9.67 %	43.8 %

Table 2: GraFRank outperforms feature-based models and GNNs (relative gains of 30-43 % MRR with respect to the best baseline) on *candidate retrieval* in Regions 1 and 2. HR@K and N@K denote Hit-Rate@K and NDCG@K metrics for $K = 5, 50$.

- (**RQ$_4$**) How do the training strategies and hyper-parameters impact convergence and performance of GraFRank?
- (**RQ$_5$**) How do the learned user embeddings in GraFRank perform across diverse user cohorts?

5.1 Experiment Setup

We now present our experimental setup with a brief description of datasets, evaluation metrics, and model training details.

5.1.1 ***Datasets.*** We evaluate GraFRank on friend recommendations using two large-scale datasets from Snapchat (Table 1). Each dataset is constructed from the interactions among users belonging to a specific country (obscured for privacy reasons). We collect 79 user features spanning four modalities and six pairwise link features, as described in Section 3.1. All features are standardized by zero-mean and unit-variance normalization before model training. In each dataset, the training set comprises timetamped friendships created during a span of 7 contiguous days. Empirically, we find that 7 days suffices to achieve good results (comparable to 1 month), thus significantly more efficient. To evaluate the quality of friendship suggestions, the test set comprises all *friend add requests* over the subsequent four days. We observe consistent results for different train-test splits across 5 time periods and 2 geographic regions. We use 10% of the labeled examples for hyper-parameter tuning.

5.1.2 ***Evaluation Metrics.*** We experiment on two friend suggestion tasks: candidate retrieval and candidate ranking (Section 4.3). To evaluate friend recommendation, we use ranking metrics Hit-Rate (HR@K), Normalized Discounted Cumulative Gain (NDCG@K) and Mean Reciprocal Rank (MRR). We adopt negative-sample evaluation [21] to generate N negative samples per positive pair (u, v) in the test set (user u has sent a friend request to user v). We then evaluate metrics for each test pair (u, v) by ranking v among the N negative samples via inner products in the latent space.

To evaluate candidate retrieval, we use $N = 10000$ *randomly sampled negative users* for each positive test pair, to emulate retrieval from a large candidate pool. Ideally, candidate ranking should operate over a shortlisted set of potential friends identified by the retrieval system. However, we aim to provide a fair benchmark comparison of different models for candidate ranking that is agnostic to the biases of the upstream retrieval system. Thus, we instead generate $N = 500$ *hard negatives samples* per test pair based on K-hop neighborhoods (Section 4.3), to ensure an unbiased comparison.

5.1.3 ***Training Details.*** We train GraFRank using $L = 2$ message passing layers per modality with a hidden dimension size of 64 and output embedding dimension $D = 32$. In each layer, we sample 15 first-order neighbors and 15 second-order neighbors for each sampled first-order neighbor; each user receives messages propagated from upto 225 friends. During model training, we apply dropout with rate of 0.3 in both layers. The model is trained for a maximum of 30 epochs with a batch size of 256 positive pairs (apart from 5 random/hard negatives per pair) and learning rate of 0.001 using Adam optimizer. We benchmark our experiments using a machine with 32 cores, 200 GB shared CPU memory, and a single Nvidia Tesla P100 GPU on the Linux platform. Our PyTorch [30] implementation of GraFRank is publicly available[1].

5.2 Baselines

We compare GraFRank on friend ranking against strong feature-based ranking models and state-of-the-art GNN models.

- **LogReg** [20]: Logistic regression classifier for link prediction. The input feature for each pair of users, is a concatenation of source and target features across the K user feature modalities.
- **XGBoost** [7]: Tree boosting model for pairwise learning to rank, trained using the same input features as LogReg. It is currently deployed at Snapchat for quick-add friend suggestions.
- **MLP** [20]: Two-layer Multi-Layer Perceptron with fully-connected layers and ReLU activations to learn user representations. We train MLP using the same ranking loss as our model (Equation 13).
- **GCN** [28]: Scalable graph convolutional networks with degree-weighted aggregation and neighborhood sampling. We concatenate user features across the K modalities into a single vector.
- **GAT** [43]: Graph attention networks with self-attentional aggregation and neighborhood sampling for scalable training.
- **SAGE + Max** [17]: Element-wise max pooling for neighbor aggregation and self-embedding concatenation at each layer.

[1] https://github.com/aravindsankar28/GraFRank

Dataset	Region 1					Region 2				
Metric	**N@5**	**N@10**	**HR@5**	**HR@10**	**MRR**	**N@5**	**N@10**	**HR@5**	**HR@10**	**MRR**
LogReg	0.1521	0.1795	0.2268	0.3116	0.1523	0.1398	0.1711	0.2136	0.3106	0.1449
MLP	0.1873	0.2190	0.2663	0.3644	0.1915	0.1927	0.2241	0.2721	0.3695	0.1967
XGBoost	0.1714	0.2002	0.2394	0.3287	0.1779	0.1844	0.2174	0.2605	0.363	0.1911
GCN	0.1345	0.1698	0.2039	0.3136	0.1462	0.1758	0.2147	0.2619	0.3826	0.1831
GAT	0.1416	0.1776	0.2197	0.3313	0.1503	0.2028	0.2445	0.2984	0.4276	0.2077
SAGE + Max	0.2063	0.2441	0.2980	0.4151	0.2094	0.2426	0.2818	0.3443	0.4654	0.2426
SAGE + Mean	0.2232	0.2607	0.3165	0.4330	0.2255	0.2766	0.3164	0.3835	0.5064	0.2744
GraFRank	**0.2684**	**0.3098**	**0.3772**	**0.5051**	**0.2669**	**0.3342**	**0.3767**	**0.4529**	**0.5841**	**0.3282**
Percentage Gains	20.25 %	18.83 %	19.18 %	16.65 %	18.36 %	20.82 %	19.06 %	18.1 %	15.34 %	19.61 %

Table 3: GraFRank achieves significant improvements (relative gains of 18-20% MRR with respect to the best baseline) over both feature-based models and prior GNNs in all ranking metrics on friend *candidate ranking* in both Region 1 and Region 2.

- **SAGE + Mean** [17]: Same as SAGE + Max with element-wise mean pooling function for neighbor aggregation.

Note that neural graph autoencoders [27] and graph convolutional matrix completion models [5] are not comparable because they cannot scale to our large-scale social network datasets.

We train all baseline GNNs in a time-sensitive manner following the same training strategy as GraFRank with time-aware neighborhood sampling and feature lookups, to compute temporal user representations. For each baseline, we train separate models for retrieval and ranking tasks; we use random negatives for retrieval while resorting to hard negatives for ranking; empirically, training separate models is vastly superior to training a single model using a mixture of random and hard negatives, or even a curriculum training scheme [50]. Our experimental results are averaged over 5 independent runs with random initializations for all methods.

5.3 Experimental Results

We first present our main results comparing GraFRank with competing baselines on candidate retrieval and ranking tasks, followed by comparisons using alternative measures of friendship quality.

5.3.1 ***Friend Candidate Retrieval and Ranking (RQ_1).*** We compare friend recommendation performance (based on add requests) of various approaches on retrieval and ranking in Tables 2 and 3 respectively. Interestingly, we find that SAGE variants outperform popular GNN models GCN and GAT. A possible explanation is the impact of feature space heterogeneity in social networks and stochastic neighbor sampling; this results in noisy user representations for GNN models (GCN, GAT) that recursively aggregate neighbor features without emphasizing self-connections. Preserving knowledge of the original features by concatenating the self-embedding in each layer results in noticeable gains (SAGE variants).

GraFRank significantly outperforms state-of-the-art approaches with over 20-30% relative MRR gains. The performance gains of GraFRank over the best baseline are statistically significant with $p < 0.01$ judged by the paired t-test. In contrast to singular aggregation over the entire feature space by prior GNNs, GraFRank handles variance in homophily across different modalities through modality-specific communication-aware neighbor aggregation. Further, the final user representations are learnt by a correlation-aware attention layer to capture discriminative facets of each modality.

5.3.2 ***Alternative Friendship Quality Indicators (RQ_2).*** In addition to evaluating friend suggestion based on friend *addition* requests, we consider other metrics to quantify friendship quality, *e.g.*, social platforms often incentivize friendships that result in greater downstream engagement. We therefore define friendship *reciprocation* and future bi-directional *communication* as two alternative measures of friendship quality. We evaluate *reciprocated* and *communicated* friend suggestion results on retrieval in Table 4.

We observe consistently high gains for GraFRank on the reciprocated and communicated friend retrieval tasks; this also demonstrates the generality of our pairwise friend ranking objective (Equation 13) in learning user representations that promote downstream engagement. Designing multi-criteria ranking objectives to balance different quality measures is worth exploring in the future.

Dataset	Add		Reciprocate		Communicate	
Metric	**HR@50**	**MRR**	**HR@50**	**MRR**	**HR@50**	**MRR**
LogReg	0.5262	0.1751	0.5582	0.2029	0.5495	0.1811
MLP	0.5726	0.1903	0.6006	0.2165	0.6001	0.1979
XGBoost	0.5957	0.2071	0.6286	0.2322	0.6407	0.2274
GCN	0.5154	0.1034	0.5329	0.1113	0.5273	0.1038
GAT	0.5352	0.0960	0.5596	0.1045	0.5654	0.0971
SAGE + Max	0.6409	0.1797	0.6653	0.2043	0.6670	0.1834
SAGE + Mean	0.6757	0.2333	0.6984	0.2609	0.7056	0.2446
GraFrank	**0.7533**	**0.3035**	**0.7756**	**0.3367**	**0.7942**	**0.3152**
Percent Gains	11.48 %	30.09%	11.05%	29.05%	12.56%	28.86 %

Table 4: Comparison of all models on *add*, *reciprocate* and *communicate* friendship retrieval tasks (reported on Region 1). GraFRank has consistent gains across all tasks.

5.4 Ablation Study (RQ_3)

In this section, we present an ablation study to analyze the *architectural modeling* choices and *training strategies* in GraFRank.

5.4.1 ***Model Architecture.*** We design three variants to study the utilities of *communication-aware* and *modality-specific aggregation.*

- **GraFRank$_{UM}$ (User-Modality)**: We analyze the contribution of link features by parameterizing the k-th modality aggregator with *just* the k-th modality user features (Equation 3). Note that link features are excluded during neighbor aggregation.

Dataset	Retrieval		Ranking	
Metric	HR@50	MRR	HR@10	MRR
(a) GraFRank$_U$	0.6968	0.2346	0.4255	0.2164
(b) GraFRank$_{UL}$	0.7069	0.2423	0.4450	0.2301
(c) GraFRank$_{UM}$	0.7239	0.2823	0.4887	0.2480
GraFRank	**0.7533**	**0.3035**	**0.5051**	**0.2669**

Table 5: Model architecture ablation study of GraFRank. Removing (c) link communication features, (b) modality-specific aggregation, or (a) both, hurt model performance.

- **GraFRank$_{UL}$ (User-Link)**: To study the effectiveness of learning *modality-specific* aggregators, we define a *single modality-agnostic* aggregator over user feature vectors obtained by concatenation across the K modalities and link features.
- **GraFRank$_U$ (User)**: We remove link features from the aggregator in **GraFRank$_{UL}$** to further test the standalone benefits of link features in parameterizing a single neighbor aggregator.

The performance of all architectural variations are reported in Table 5. GraFRank$_{UL}$ performs much worse than GraFRank highlighting the benefits of learning multiple modality-specific aggregators to account for varying extents of modality homophily.

Communication-aware neighbor aggregation is effective at identifying actively engaged friends during neighbor aggregation; this is evidenced by the gains of GraFRank over GraFRank$_{UM}$ (modality-aware user feature aggregation). We find noticeable gains from parameterizing the aggregator with link features, even in the absence of modality-specific aggregation, from the comparison between GraFRank$_{UL}$ and GraFRank$_U$ (single user feature aggregator).

5.4.2 ***Training Strategy.*** We examine different training strategies to learn GNN models for friend recommendation. We train GraFRank with random negatives for candidate retrieval, but adopt two-phase hard negative fine-tuning (with random negative pretraining) for candidate ranking. To validate our choices, we examine three model training settings: (a) random negative training, (b) hard negative training, and (c) fine-tuning (after pretraining on random negatives), for two GNN models: SAGE + Mean and GraFRank, across both candidate retrieval and ranking tasks. Note that training with combination of random and hard negatives, as proposed in [50], is excluded since it consistently performs worse than the above three strategies on both retrieval and ranking tasks.

Dataset	Retrieval		Ranking	
Metric	HR@50	MRR	HR@10	MRR
SAGE + Mean (random)	0.6757	0.2333	0.3943	0.1923
SAGE + Mean (hard)	0.3275	0.0766	0.4330	0.2255
SAGE + Mean (fine-tune)	0.3978	0.0965	0.4561	0.2372
GraFRank (random)	**0.7533**	**0.3035**	0.4655	0.2254
GraFRank (hard)	0.4542	0.1461	0.4823	0.2594
GraFRank (fine-tune)	0.5283	0.1871	**0.5051**	**0.2669**

Table 6: Training strategy comparison of two GNNs across retrieval and ranking tasks. Random negative training achieves best results for retrieval. Random negative pretraining with hard negative fine-tuning benefits ranking.

We make three consistent observations from the performance comparison (Table 6) across all of the compared GNN models:

- Random negative training achieves best results for retrieval, but performs poorly on ranking; such models lack the *resolution* to discriminate amongst potential candidates for re-ranking.
- Training on hard negatives improves candidate ranking as expected, yet results in poor retrieval performance. Learning fine profile-oriented distinctions among graph-based neighbors is actually detrimental to the coarse-grained task of retrieval.
- Random negative pretraining yields good parameter initialization points that are more conducive for effective fine-tuning on hard negatives. Fine-tuning improves results for all GNNs over direct hard negative training on ranking, but is ineffective for retrieval.

5.5 Training and Sensitivity Analysis (RQ$_4$)

In this section, we quantitatively analyze model *convergence* and model *sensitivity* to sampled *neighborhood sizes* in GNN models.

5.5.1 ***Model Training Analysis.*** We investigate the relative abilities of different models to optimize the pairwise friend ranking objective (Equation 13). We compare the convergence rates of baselines MLP, SAGE + Mean, and our model GraFRank under both random and hard negative training settings, by examining the average training loss per epoch in Figures 6 (a) and (b) respectively.

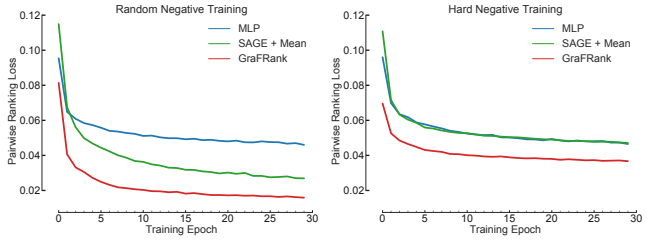

Figure 6: GraFRank converges faster to better optimization minima in *random* and *hard* negative settings, which translates to notable gains on both retrieval and ranking tasks.

As expected, all models converge to a lower training loss against random negatives (Figure 6 (a)) when compared to hard negatives (Figure 6 (b)). Interestingly, SAGE + Mean shows similar training convergence as MLP in Figure 6 (b), but achieves better test results; this indicates better generalization for GNNs over feature-based models. Compared to baselines, GraFRank converges to a better optimization minimum under both random and hard negative settings, which also generalizes to better test results (Tables 2 and 3).

5.5.2 ***Runtime and Sensitivity Analysis.*** A key trade-off in training scalable GNN models lies in choosing the size of sampled neighborhoods T in each message-passing layer. In our experiments, we train two-layer GNN models for friend ranking. Figure 8 shows the runtime and performance of SAGE + Mean and GraFRank for different sizes of sampled neighborhoods T from 5 to 20.

Model training time generally increases linearly with T, but has a greater slope after $T = 15$. We also observe diminishing returns in model performance (MRR) with increase in the size of sampled neighborhood T after $T = 15$. Thus, we select a two-layer GNN model with layer-wise neighborhood size of 15, to provide an effective trade-off between computational cost and performance.

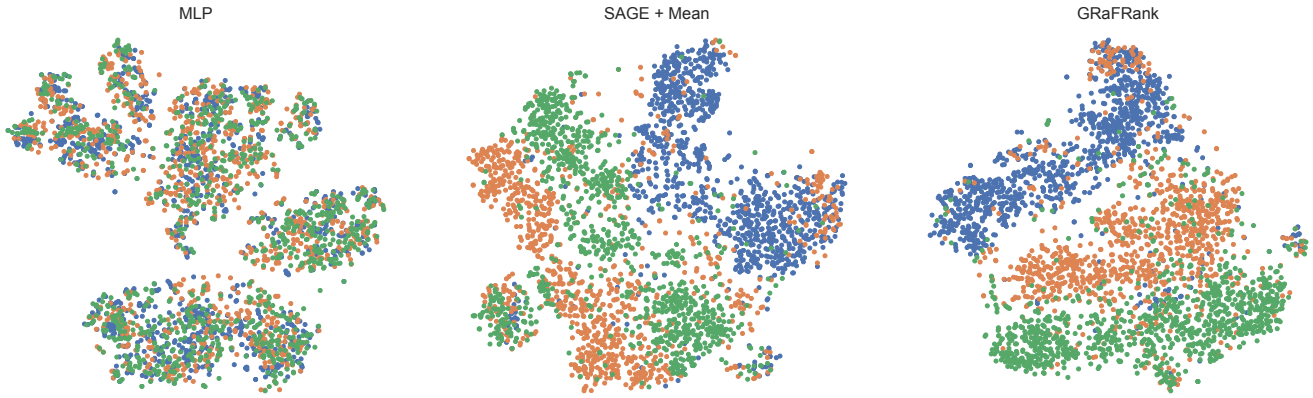

Figure 7: Visualization of two-dimensional t-SNE transformed user representations from feature-based MLP, and GNN models: SAGE + Mean, and GraFRank. Users with the same color belong to the same city. Compared to MLP and SAGE + Mean, the friendship relationships learnt by GraFRank result in well-separated user clusters capturing geographical proximity.

Compared to SAGE + Mean, GraFRank has marginally higher training times, yet achieves significant performance gains (20% MRR), justifying the added cost of modality-specific aggregation.

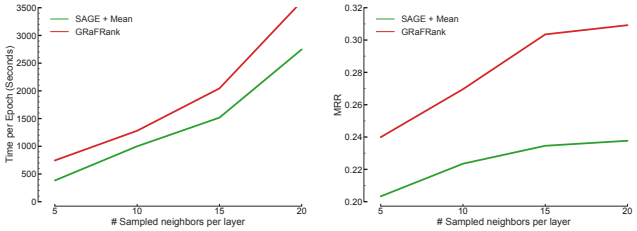

Figure 8: We observe diminishing returns in MRR after neighborhood size $T = 15$; GraFRank has significant gains over SAGE + Mean, with marginally higher training times.

5.6 User Cohort Analysis (RQ_5)

In this section, we present multiple *qualitative* analyses to examine model performance across user segments with varied node *degree* and friending *activity* levels, and compare *t-SNE visualizations* of user representations learned by different neural models.

5.6.1 ***Impact of degree and activity.*** We examine friend recommendation performance across users with different node degrees and friending activities. Specifically, for each test user, *degree* is the number of friends, and *activity* is the number of friend requests sent/received in the past 30 days. We divide the test users into groups independently based on their degree and activity levels. We compare GraFRank with feature-based models MLP, XGBoost, and the best GNN baseline SAGE + Mean. Figures 9(a) and (b) depict friend candidate retrieval performance HR@50 across user segments with different degrees and activities respectively.

From Figure 9(a), overall model performance generally increases with node degree due to the availability of more structural information. GraFRank has significant improvements across all user segments, with notably higher gains for *low-to-medium* degree users (relative gains of 20%). GraFRank prioritizes active friendships by communication-aware message-passing, which compensates for the lack of sufficient local connectivities in the ego-network.

The performance variation across users with different *activity* levels in Figure 9(b), exhibits more distinctive trends with clear gains for GNN models over feature-based MLP and XGBoost for less-active users. Significantly, GraFRank has much stronger gains over SAGE + Mean, in *less-active user segments*, owing to its *multi-faceted modeling* of heterogeneous in-platform user actions. GraFRank effectively overcomes sparsity concerns for less-active users, through modality-specific neighbor aggregation over multi-modal user features to learn expressive user representations for friend ranking.

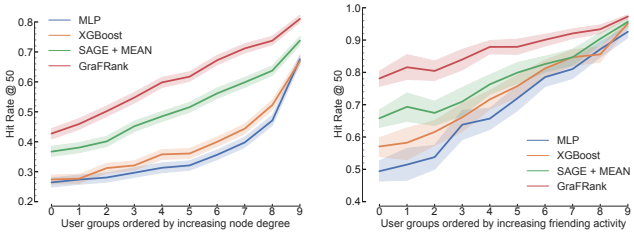

Figure 9: GraFRank has significant improvements across all user segments, with notably larger gains for *low-to-medium degree* users (a), and *less-active* users (b).

5.6.2 ***Visualization.*** To analyze the *versatility* of learned user embeddings, we present a qualitative visualization to compare different models on their expressivity to capture geographical user proximity. We randomly select users from three different cities within Region 1 and use t-SNE [32] to transform their learned embeddings into two-dimensional vectors. Figure 7 compares the visualization results from different neural models. Evidently, the visualization learned by MLP does not capture geographical proximity, while the GNN models are capable of grouping users located within the same city. Compared to SAGE + Mean, GraFRank forms even more well-segmented groups with minimal inter-cluster overlap.

6 CONCLUSION

This paper investigates graph neural network design for friend suggestion in large-scale social platforms. We formulate friend suggestion as multi-faceted friend ranking with multi-modal user features and link communication features. Motivated by our empirical

insights on user feature modalities, we design a neural architecture GraFRank that handles heterogeneity in modality homophily via modality-specific neighbor aggregators, and learns non-linear modality correlations through cross-modality attention. Our experiments on two multi-million user datasets from Snapchat reveal significant improvements in friend candidate retrieval (30% MRR gains) and ranking (20% MRR gains), with stronger gains for the crucial population of less-active and low-degree users. Although our case studies are conducted on a single platform Snapchat, we expect GraFRank to be directly applicable to popular bidirectional friending platforms (*e.g.*, Facebook, LinkedIn) with minor adaptations for unidirectional scenarios (*e.g.*, Twitter, Instagram).

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Luca Maria Aiello, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. 2012. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)* 6, 2 (2012), 1–33.

[3] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[4] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval* 12, 4 (2009), 461–486.

[5] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[6] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*. 942–950.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*. ACM, 785–794.

[8] Zhengdao Chen, Lisha Li, and Joan Bruna. 2019. Supervised Community Detection with Line Graph Neural Networks. In *ICLR*. OpenReview.net.

[9] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*. ACM, 257–266.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.

[11] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE TKDE* 31, 5 (2018), 833–852.

[12] Daizong Ding, Mi Zhang, Shao-Yuan Li, Jie Tang, Xiaotie Chen, and Zhi-Hua Zhou. 2017. Baydnn: Friend recommendation with bayesian personalized ranking deep neural network. In *CIKM*. 1479–1488.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[14] Golnoosh Farnadi, Jie Tang, Martine De Cock, and Marie-Francine Moens. 2018. User profiling through deep multimodal fusion. In *WSDM*. 171–179.

[15] Xu Geng, Xiyu Wu, Lingyu Zhang, Qiang Yang, Yan Liu, and Jieping Ye. 2019. Multi-modal graph interaction for multi-graph convolution network in urban spatiotemporal forecasting. *arXiv preprint arXiv:1905.11395* (2019).

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

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

[18] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[19] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society.* 28, 1 (1979), 100–108.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media.

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

[22] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM TOIS* 22, 1 (2004), 5–53.

[23] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. In *Advances in neural information processing systems*. 4558–4567.

[24] Ankit Jain, Isaac Liu, Ankur Sarda, , and Piero Molino. 2019. Food Discovery with Uber Eats: Recommending for the Marketplace. (2019). https://eng.uber.com/uber-eats-graph-learning/

[25] Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In *MM*. 795–816.

[26] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[27] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[28] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[29] Adit Krishnan, Hari Cheruvu, Cheng Tao, and Hari Sundaram. 2019. A Modular Adversarial Approach to Social Recommendation. In *CIKM*. ACM, 1753–1762.

[30] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. 2020. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704* (2020).

[31] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *JASIST* 58, 7 (2007), 1019–1031.

[32] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[33] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[34] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. 2005. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD explorations newsletter* 7, 2 (2005), 23–30.

[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.

[36] Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. 2020. Beyond Localized Graph Neural Networks: An Attributed Motif Regularization Framework. In *ICDM*.

[37] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*. 519–527.

[38] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and Hari Sundaram. 2020. GroupIM: A Mutual Information Maximization Framework for Neural Group Recommendation. In *SIGIR*. 1279–1288.

[39] Aravind Sankar, Xinyang Zhang, Adit Krishnan, and Jiawei Han. 2020. Inf-VAE: A Variational Autoencoder Framework to Integrate Homophily and Influence in Diffusion Prediction. In *WSDM*. 510–518.

[40] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. Timecrunch: Interpretable dynamic graph summarization. In *KDD*. ACM.

[41] Sucheta Soundarajan, Acar Tamersoy, Elias B Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. 2016. Generating graph snapshots from streaming edge data. In *WWW*. 109–110.

[42] Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. 2020. Knowing your FATE: Friendship, Action and Temporal Explanations for User Engagement Prediction on Social Apps. In *KDD*. ACM, 2269–2279.

[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR* (2018).

[44] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*. ACM, 839–848.

[45] Menghan Wang, Yujie Lin, Guli Lin, Keping Yang, and Xiao-Ming Wu. 2020. M2GRL: A Multi-task Multi-view Graph Representation Learning Framework for Web-scale Recommender Systems. In *KDD*. ACM, 2349–2358.

[46] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *MM*. 1437–1445.

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE TNNLS* (2020).

[48] Yuxin Xiao, Adit Krishnan, and Hari Sundaram. 2020. Discovering strategic behaviors for collaborative content-production in social networks. In *WWW*. 2078–2088.

[49] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *KDD*. ACM, 2434–2443.

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

[51] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*. OpenReview.net.

[52] Muhan Zhang and Yixin Chen. 2017. Weisfeiler-lehman neural machine for link prediction. In *KDD*. ACM, 575–583.

[53] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*. 5165–5175.

[54] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *ICLR*. OpenReview.net.