Improving Embedding-Based Retrieval in Friend Recommendation with ANN Query Expansion

Pau Kung, Zihao Fan, Tong Zhao, Yozen Liu, Zhixin Lai, Jiahui Shi, Yan Wu, Jun Yu, Neil Shah, Ganesh Venkataraman
{pkung,zfan3,tzhao,yliu2,zlai,jshi3,ywu,jyu,nshah,gvenkataraman}@snapchat.com
Snap Inc.
Santa Monica, CA

ABSTRACT

Embedding-based retrieval in graph-based recommendation has shown great improvements over traditional graph walk retrieval methods, and has been adopted in large-scale industry applications such as friend recommendations [16]. However, it is not without its challenges: retraining graph embeddings frequently due to changing data is slow and costly, and producing high recall of approximate nearest neighbor search (ANN) on such embeddings is challenging due to the power law distribution of the indexed users. In this work, we address theses issues by introducing a simple query expansion method in ANN, called *FriendSeedSelection*, where for each node query, we construct a set of 1-hop embeddings and run ANN search. We highlight our approach *does not require any model-level tuning*, and is inferred from the data at test-time. This design choice effectively enables our recommendation system to adapt to the changing graph distribution without frequent heavy model retraining. We also discuss how we design our system to efficiently construct such queries online to support 10k+ QPS. For friend recommendation, our method shows improvements of recall, and 11% relative friend reciprocated communication metric gains, now serving over 800 million monthly active users at Snapchat.

KEYWORDS

Embedding-Based Retrieval, Approximate Nearest Neighbors, Graph Neural Networks, Query Expansion

ACM Reference Format:
Pau Kung, Zihao Fan, Tong Zhao, Yozen Liu, Zhixin Lai, Jiahui Shi, Yan Wu, Jun Yu, Neil Shah, Ganesh Venkataraman. 2024. Improving Embedding-Based Retrieval in Friend Recommendation with ANN Query Expansion. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (SIGIR).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In online social and professional network applications, users build their list of friends based on their real life connections. App features like LinkedIn's People You May Know (PYMK)[2] and Snapchat's QuickAdd[16] serve a list of relevant friend recommendations to whom they believe the user is already connected outside the app. A typical friend recommendation pipeline can be broken into successive stages, as depicted in Figure 1. In the retrieval stage, the goal is to generate as many potential friends as possible, typically using a mix of machine learning models and graph walk methods[3, 4, 16]. Retrieval plays a particularly important part because the ability to fetch relevant candidates, i.e. recall of real-life friends, directly ties to better ranking results, and thus user's positive in-app engagements and retention.

Recently, embedding-based retrieval (EBR)[7] has demonstrated strong recall quality in search recommendation. Subsequent works showed EBR provides improvements over traditional graph walk techniques (e.g. fetching friends of friend as suggestions), and adopted in production in large-scale industrial social network applications [2, 16]. For friend recommendations, the process of EBR consists of training user embeddings using deep neural networks on the user-friend graph[6, 11]. Given the indexed user embeddings, we retrieve relevant results by running an approximate nearest neighbor (ANN) search on the query embedding. One notable

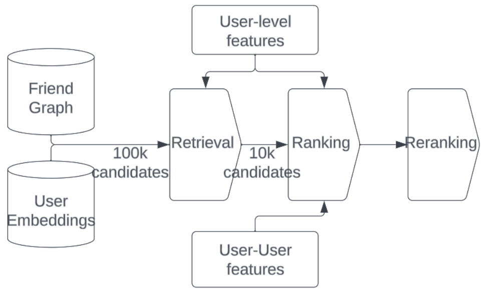

Figure 1: High-level architecture of our multi-phase friend Recommendation pipeline.

challenge to EBR in social network applications is the power-law distribution of the user node degrees [15]. Most users in the training set have sparse connections and edge properties during the training time. Additionally, a user's friends may be grouped into multiple social circles [13]. For example, a user u may have friends that belong to two separate social circles $c1$ and $c2$. The friends in $c1$ may not have mutual connections to friends in $c2$. Also, the friends in $c1$ have significantly different user behavior patterns than those in $c2$. Jointly learning on the friendship and interaction graphs of these users would bring u's embedding and u's friends embeddings closer. However, in practice, we want to be able to discern candidates across social circles so the retrieved candidates can cover as many user's social circles as possible. This challenge is akin to the problem of multi-modal user representation in a typical user-item recommender system [14]. Methods like PinnerSAGE [14] showed that using multiple embedding to represent user outperforms single vector representation in item recall and precision metrics.

Taking a multi-embedding perspective, we propose a simple post-training step, *FriendSeedSelection*, to produce a multi-modal user representation for ANN search, which can be implemented as a query expansion step in retrieval systems. User representation is constructed by sampling user's 1-hop friends, and using their respective embeddings. Importantly, our design is model agnostic and does not require any model-level tuning. In other words, we infer seeds directly from user data at test-time. This direct test-data adaptation allows us to work with the change graph distribution and reduces the frequency of cost-heavy model retraining. We also highlight the feature engineering required to achieve better results, due to the unique nature of graph-based recommendation. We show that this method generates higher recall at retrieval stage, as well as friending business metric improvements in a large industry scale application serving 800 million monthly active users. This method is currently in live production usage at Snapchat.

2 EMBEDDING-BASED RETRIEVAL AT SNAPCHAT FRIENDING

Embedding-based retrieval (EBR) for friend recommendation addresses the limitations of graph-traversal approaches (e.g., friend-of-friend) [16], namely (1) it is computationally expensive to fetch 2-hop candidates in small-world social graphs [10], (2) traversal in cold start friend graph yields limited results, and (3) traversal cannot easily incorporate connection strength or user behaviors.

At Snap, we have developed an embedding-based retrieval system that learns a graph-based embedding for each user, leveraging the corresponding user-user friendship graph containing rich information of user connections. The graph's node attributes contain individual user features, and edge attributes contain various user to user interaction patterns. Since we optimize for the users with similar interest and in close network proximity to be closer in the embedding space as an objective, therefore we can address the limitations of traversal in cold start user graph with approximate nearest neighbor (ANN) search in embedding space, and the search time complexity is constant, compared to traversal approaches that are edge topology dependent.

We utilize graph neural networks (GNN) for learning user embeddings. We define the user graph by an attributed graph $G = (V, E, X)$, where V denotes the node set with $|V| = n$, E denotes the edge set with $|E| = m$, $X \in \mathbb{R}^{n\times f}$ is the node embedding matrix. Each node v_i is a vector of user attributes, and each edge $e_{i,j}$ is a vector of user-user interaction features. During training, we are given a snapshot of G_D of all nodes and edges that are present at time period $[D - L, D]$. For each user pair (u_1, u_2) we want to predict whether the friend link is formed at time $D + 1$.

For EBR, we employ a 2-layer GNN model with GAT convolution as message passing layers [18]. Specifically, the k^{th} layer embeddings are dependent upon the $k - 1^{th}$ layer embeddings via:

$$x_i^k = a_{i,i}\theta x_i^{k-1} + \sum_{j\in N(i)} a_{i,j}\theta x_j^{k-1}$$

$$a_{i,j} = \mathrm{softmax}(\mathrm{LeakyReLU}(a^T[x_i^{k-1}||x_j^{k-1}||e_{i,j}^{k-1}]))$$

The attention weights $a_{i,j}$ can be seen as soft attention between node i and its 1-hop neighbors, where the attention weights are a shared additive vector, computed by respective edge features and node embeddings.

The model is training with contrastive loss [8], which is computed as an in-batch negative sampling calculation of cross entropy loss, similar to Tensorflow's retrieval loss [1]. For more details regarding the details of graph embedding model training and formulation, please see [16].

3 METHODOLOGY

In a typical ANN search over an index of all user embeddings, for a user query embedding vector x_i, we perform approximate embedding similarity search (e.g. product quantization [9] and efficient navigation such as HNSW [12]). However, a drawback of using purely user's embedding vector is the potential information loss trying to capture multiple social circles the user belongs to. We want to ensure that the query can sufficiently represent the user's respective social circles to improve the coverage and diversity of ANN search results.

We propose to instead use a set of embedding vectors to serve as the multi-modal user representation in the ANN search. For each searcher u, we span over all 1-hop bi-directional friends of u, F_u. $E_{fu} \in \mathbb{R}^{|F_u|\times e}$ denotes the corresponding embedding vectors for F_u, with embedding dimension e. We propose a simple query expansion step, *FriendSeedSelection*, as described below:

- Set ANN retrieval size R
- Sample k seeds $S_u \sim H$, where H is some distribution over the edge features between searcher u and their friends.
- $\forall s_i \in S_u$, fetch seed s_i's embedding $E_{fu}[i]$, and find R/k nearest neighbors. Concatenate all neighbors as final result

We believe that each searcher's friend embedding acts as an approximate representation of the corresponding embedding of the social circle both searcher and her friend belongs to, and can hopefully act as an ensemble to improve the generalization of recall.

3.1 Seed Selection Strategies

We believe the way seeds are sampled should be determined according to user-friend interaction behaviors. We mainly consider:

Recency of friendship formation: how recently the user makes a friend on the platform is indicative of how likely the user will be interested to onboard more friends from the underlying social circle they both belong to. We denote the friendship recency feature as $friend-made-ld_{i,j}^T$, which means if user i has formed a friendship with user j within a time window of [T-d, T].

Interaction frequency: on Snapchat, a user may view a friend's story, or engage in chat-related activities with friends. We represent such activities in terms of count-based features and active-day based features: $activity - d_{i,j}^T$, $activity - active - days - d_{i,j}^T$, where we look at chat-send, snap-send, story-viewing, etc. as some of the valid activities. Active-day features denote how many days users i and j have initiated the interaction between [T-d, T]. Count-based features denote the interaction count in [T-d, T].

Algorithm 1 Close Friend Seed Selection

```
Require: Searcher s, embedding lookup function $E(u) \longrightarrow x_u$, seed size $k$, cluster input size $n$
  $R \leftarrow Nei(s)$ sorted by (1), $R \leftarrow R[:n]$   ▷ Close Friends Filtering
  Run Ward clustering on the $E(R)$ yielding $C = c_1, .., c_n$ cluster assignment
  $\forall c_i \in C, c_i \in [0, G]$
  $M \leftarrow []$   ▷ Medoids assignment
  for $g \in range(G)$ do
      $m \leftarrow medoid\{c_i \forall c_i = g\}$ wrt embeddings
      $M \leftarrow (M, m)$
  end for
  Sort M by cosine similarity(E(s), E(m))
  return M
```

Based on the features, we describe several FriendSeedSelection variants that leverage such features:

3.1.1 Stochastic EBR (V1). As a baseline, we generate random samples from 1-hop bi-directional friends as S_u, and use the corresponding friend embeddings in ANN search.

3.1.2 Location-based Selection (V2). We rank 1-hop friends using registered city and select ones with the closest distances. Friends from similar locations are more likely to interact in real life and have denser social circles.

3.1.3 Recency-based Seed Selection (V3). We sample 1-hop bi-directional friends by linearly weighting recency of friend link formation:

$$\sum_{i \in S} a_i * friend - made_i, S = [l1, l7, l14, l28]$$

For date D, the features are indicator variables of friends made after D-1, D-7, D-14, and D-28, respectively.

3.1.4 Engagement strength-based Seed Selection (V4). For user u's 1-hop friend, we score the friends according to the following interaction feature which provides a measure of interaction strength:

$$F(u, f) = \sum_i a_i * f^i_{u,f} \tag{1}$$

f denotes a set of activity-active-days features. We select d nodes, N_d from 1-hop friend as qualified seeds. Next, we run hierarchical clustering on the embeddings of N_d, which generates c clusters. We rank the c medoids by embedding L2 distance to the user u's embedding and return top k medoids as the finalized seed set for ANN. We describe the process more formally in Algorithm 1.

For optimal online performance, we tune the clustering hyperparameters (e.g., dissimilarity threshold) such that the median empirical cluster number is around 5 to 8.

3.1.5 Ensemble Seed Selection (V5). We combine V2+V3 as an ensemble of seed selection strategies. Specifically, for the seed set size of k, we sample k2 seeds with V2, and k3 seeds with V3, k2+k3=k. This is a simple set union, and should help provide insight to see if V2 and V3 offer complementary signals.

4 SYSTEM DESIGN

4.1 Offline Evaluation

For offline evaluation, we want to iterate and validate our model architecture and embedding post-processing performance to align with online performance. To this end, we designed an offline recall evaluation framework, where we simulate embedding mapping and query expansion process to create a dataset of multi-modal user embeddings. Figure 2 outlines the system design. We create two tables, Query and Label tables, using BigQuery, to store ground truth and the simulated query set. For each user, we pass the constructed query to an offline ANN index server. For users queried at date D, we join the simulated suggestion results with online log data collected at date D, and evaluate the corresponding recall. This offline evaluation framework allows us to gate embedding quality using the offline metrics for each proposed model improvement.

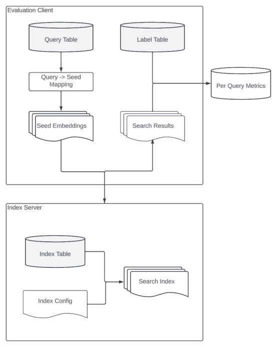

Figure 2: System design of offline evaluation pipeline

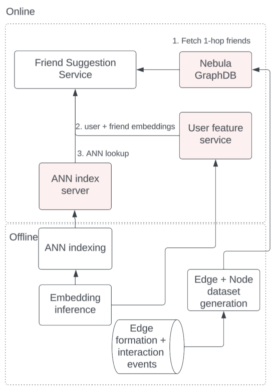

Figure 3: System design diagram for ANN serving and dynamic seed selection

4.2 Serving ANN

For online serving, we leverage Nebula graph database [19] that allows us to do efficient node and edge property lookup in graph walks. Our dataset works with over 10 billion edges and 800 million nodes. This implies 4TB of graph data, and 2TB of embedding index. To support large-scale realtime usage (10k RPS) at a cost-effective manner, we opted to partition our ANN indices by core geographical regions, based on the observation that friendship has a strong geographical correlation [17]. This optimization helps reap good infrastructure cost savings while achieving low end-to-end query latency. Figure 3 shows the overall system design.

5 EVALUATION

5.1 Evaluation Setup

To validate our method, we measure the recall with historical user data. We queried monthly active users in an anonymized country snapshot on 01/28/2024. We fetched all friendship and engagement-labeled edges with these users as the source node. We randomly sampled 2 million users for analysis.

To evaluate the results, we ran multiple trials of 10k unique sampled users with at least one friend request sent reciprocated between 01/29 and 02/11, and sampled up to 5 new friends made per add sender as their labels. This setup allows us to not biased towards power users in our analysis. In total we have around 20k new friends for each sample used as positive labels.

We adopt the following label pre-processing: assume we have 2 seeds **A, B**, where we retrieve top n results for each seed during

ANN search $A_r = [a1, a2, \ldots, an]$, $B_r = [b1, b2, \ldots, bn]$. We combine the results in the order of increasing embedding distance of the candidate to searcher's embedding.

For negative samples, we randomly selected 1M other MAUs as distractors. Then we built a HNSW index containing both the positive and negatives samples. We used FAISS [5] implementation of HNSWFlat index with default parameters (m=32, efConstruction=40, efSearch=16).

For each searcher, we fetched 1000 candidates in total to calculate their recall@100 and recall@1k metrics. That is 200 candidates fetched from each seed with 5 seeds if we enable seed selection, or 1000 candidates if we use the searcher's own embedding.

5.2 Offline Evaluation

For offline evaluation, We compare the following treatments, where we are given a query user, its corresponding embedding vector, seed set size 5:
Baseline: Using query's embedding vector for approximate nearest neighbor search; ***Random friends***: Using the query user's 1-hop neighbors, randomly sampled 5 seeds, and each fetches ANNs; ***PinnerSAGE***: We run embedding clustering on all searcher's friends, and select top 5 seeds; ***Location-based seed selection***: We use the searcher's friends by closest geodistance, and select top 5 seeds; ***Recency***: We select top 5 seeds from 1-hop neighbors (1) as the ranking criterion; ***Engagement***: We select top 5 seeds using (2) as the ranking criterion; ***Ensemble***: We combine recency-based seed selection and engagement-based seed selection where we select 4 seeds from recency-based and 1 from engagement based.

Each seed runs ANN that returns n/5 results, and are finally combined as the ANN result. Table 1 shows the results of offline evaluation. It is worth noting that most seed selection methods underperform in terms of recall@100, but when we increase the retrieval size, the recall increases dramatically. Also it is important to call out that seed selection methods are data adaptive, so if we include more data in test time window, we should expect recall to improve. For the retrieval size of 1000, for our usecase, this is a reasonable precision / recall tradeoff. In practice, we find that L2 rankers can be directly benefited when rolling out these changes on retrieval size, with such retrieval sizes.

In terms of features, we find that recency and engagement features are both significantly helping performance. Ensemble of the two signals work the best. It is interesting that PinnerSAGE, did not yield strong performance gains, highlighting the needs to tune seeds based on user interaction signals.

Additionally, we highlight that random selection is already doing very well for recall@1000. Although this may seem counterintuitive, we hypothesize that the role of social circle association is very important in the online setting. In the case of EBR serving candidates from multiple social circles, ranker is more likely to serve candidates with higher social circle coverage.

6 A/B TEST RESULTS

We setup an online A/B experiment to test the production impact . The A/B was done on 10M sampled users per treatment, and was run in succession with the following treatments:

- Control: Vanilla EBR
- Experiment 1, Treatment: V1 (Stochastic EBR)

Table 1: Offline evaluation

Method	**Recall@100**	**Recall@1000**
Baseline	0.1857	0.2426
Random	0.1739	0.3417
PinnerSAGE	0.1569	0.3226
Location	0.1717	0.3329
Recency	0.1800	0.3469
Engagement	0.1768	0.3549
Ensemble	**<u>0.1861</u>**	**<u>0.3560</u>**

Table 2: Online A/B Results

Experiment	**Friends Made with Communication**
Control (Baseline)	0%
Experiment 1	Control **+7.7%**
Experiment 2	Experiment 1 **+1.7%**
Experiment 3	Experiment 2 **+1.6%**
Overall	Implies 11% improvements over baseline

- Experiment 2, Treatment: V3 (Recency-Based Selection)
- Experiment 3, Treatment: V5 (Ensemble)

All treatments are selected with 5 seeds, so each seed is allocated the same number of ANN budget. Table 2 shows the A/B result. We measure the results by new friends made with communication (e.g., chatting, viewing friend contents). The metric is defined by: the user adds a suggested friend candidate, the candidate reciprocates and initiates qualified interactions within certain days. This metric denotes a high-quality friendship made on the platform and suggests that the friend is likely a valid friendship in the real world. From the result, naive treatment 1 already offers substantial gain. This confirms the hypothesis that multi-modal representation of users is much more effective than single embedding. As we include more refined selection criteria, we observe further metric gains, showing 11% over the EBR baseline.

We also tested variants where we try to explicitly include searcher embedding as part of seed selection. Interestingly, we did not find any notable difference whether or not we use the searcher embedding. We hypothesized that it is more important to do similarity search within each social circle, and excluding searcher embedding may balance the results more evenly across searcher's social circles.

7 CONCLUSION

In this work, we introduced *FriendSeedSelection*, a test-time query expansion method for improving EBR approximate nearest neighbor search (ANN) for friend recommendation at Snapchat. We presented several approaches to one-hop neighbor selection, using a mixture of feature engineering, and embedding space coverage using hierarchical clustering on embedding vectors. In online A/B presented to tens of millions of users, we showed empirically that such ensemble embeddings meaningfully improved friending engagement metrics.

A few promising directions for future work include (1) compute more expressive multi-hop graph features offline to guide seed selection, and (2) dynamically adjust seed selection and sampling via user's friending activity feedback.

8 PRESENTER BIO

Jun Yu is a senior engineering leader in Core Growth org at Snap Inc., building products that help users make virtuous friends on Snapchat and drive community growth. Prior to Snap, he has held senior applied machine learning positions at Amazon and eBay, and worked on a broad range of ML applications including paid internet marketing on Google/Facebook/Display Ads, Amazon promotion pricing and scheduling, notification campaign optimization, and buyer/seller risk management. He has a passion in teaching and is currently teaching several Machine Learning courses at University of Washington Michael G. Foster School of Business. He holds a Ph.D. in Computer Science from Oregon State University and extensively published in top-tier Machine Learning conferences and journals.

REFERENCES

[1] 2024. *Tensorflow Recommenders.* Retrieved Feb 15, 2024 from https://github.com/tensorflow/recommenders/blob/v0.7.3/tensorflow_recommenders/tasks/retrieval.py#L27-L211

[2] Parag Agrawal. 2024. *Building a Large-Scale Recommendation System: People You May Know.* Retrieved Feb 15, 2024 from https://www.linkedin.com/blog/engineering/recommendations/building-a-large-scale-recommendation-system-people-you-may-know

[3] Fedor Borisyuk, Krishnaram Kenthapadi, David Stein, and Bo Zhao. 2016. CaSMoS: A framework for learning candidate selection models over structured queries and documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 441–450.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems.* 191–198.

[5] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17).* Curran Associates Inc., Red Hook, NY, USA, 1025–1035.

[7] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20).* Association for Computing Machinery, New York, NY, USA, 2553–2561. https://doi.org/10.1145/3394486.3403305

[8] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research* (2022). https://openreview.net/forum?id=jKN1pXi7b0

[9] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[10] Jon Kleinberg. 2000. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing* (Portland, Oregon, USA) *(STOC '00).* Association for Computing Machinery, New York, NY, USA, 163–170. https://doi.org/10.1145/335305.335325

[11] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference.* Palo Alto, CA, USA.

[12] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (apr 2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[13] Julian McAuley and Jure Leskovec. 2012. Learning to discover social circles in ego networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Lake Tahoe, Nevada) *(NIPS'12).* Curran Associates Inc., Red Hook, NY, USA, 539–547.

[14] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20).* Association for Computing Machinery, New York, NY, USA, 2311–2320. https://doi.org/10.1145/3394486.3403280

[15] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference 2021.* 2535–2546.

[16] Jiahui Shi, Vivek Chaurasiya, Yozen Liu, Shubham Vij, Yan Wu, Satya Kanduri, Neil Shah, Peicheng Yu, Nik Srivastava, Lei Shi, Ganesh Venkataraman, and Jun Yu. 2023. Embedding Based Retrieval in Friend Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei) *(SIGIR '23).* Association for Computing Machinery, New York, NY, USA, 3330–3334. https://doi.org/10.1145/3539618.3591848

[17] Johan Ugander and Lars Backstrom. 2013. Balanced label propagation for partitioning massive graphs. In *Proceedings of the sixth ACM international conference on Web search and data mining.* 507–516.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations.* https://openreview.net/forum?id=rJXMpikCZ

[19] Min Wu, Xinglu Yi, Hui Yu, Yu Liu, and Yujue Wang. 2022. Nebula Graph: An open source distributed graph database. arXiv:2206.07278 [cs.DB]