Embedding Based Retrieval in Friend Recommendation

Jiahui Shi, Vivek Chaurasiya, Yozen Liu, Shubham Vij, Yan Wu, Satya Kanduri
Neil Shah, Peicheng Yu, Nik Srivastava, Lei Shi, Ganesh Venkataraman, Jun Yu
{jshi3,vchaurasiya,yliu2,svij,ywu,satya.kanduri,nshah,peicheng.yu,nsrivastava,lshi3,gvenkataraman,jyu3}@snap.com
Snap Inc.
Santa Monica, CA USA

ABSTRACT

Friend recommendation systems in online social and professional networks such as Snapchat helps users find friends and build connections, leading to better user engagement and retention. Traditional friend recommendation systems take advantage of the principle of locality and use graph traversal to retrieve friend candidates, e.g. Friends-of-Friends (FoF). While this approach has been adopted and shown efficacy in companies with large online networks such as Linkedin and Facebook, it suffers several challenges: *(i)* discrete graph traversal offers limited reach in cold-start settings, *(ii)* it is expensive and infeasible in realtime settings beyond 1 or 2 hop requests owing to latency constraints, and *(iii)* it cannot well-capture the complexity of graph topology or connection strengths, forcing one to resort to other mechanisms to rank and find top-K candidates. In this paper, we proposed a new ***Embedding Based Retrieval (EBR)*** system for retrieving friend candidates, which complements the traditional FoF retrieval by retrieving candidates beyond 2-hop, and providing a natural way to rank FoF candidates. Through online A/B test, we observe statistically significant improvements in the number of friendships made with EBR as an additional retrieval source in both low- and high-density network markets. Our contributions in this work include deploying a novel retrieval system to a large-scale friend recommendation system at Snapchat, generating embeddings for billions of users using Graph Neural Networks, and building EBR infrastructure in production to support Snapchat scale.

KEYWORDS

Social Networks, Friend Recommendation, Embedding Based Retrieval, User Embedding, Graph Neural Network

ACM Reference Format:
Jiahui Shi, Vivek Chaurasiya, Yozen Liu, Shubham Vij, Yan Wu, Satya Kanduri, Neil Shah, Peicheng Yu, Nik Srivastava, Lei Shi, Ganesh Venkataraman, 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 (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591848

1 FRIEND RECOMMENDATION SYSTEMS

In online social and professional networks [2, 6, 19, 20], a user's friends or connections are critical for one's engagement and retention. Research at Linkedin [36] showed that "members with at least 13 connections from companies other than their current employer are 22.9% faster in transitioning to their next job than those who do not". Friend recommendation can be formulated as a link prediction problem [22], where the goal is to predict the links that are to be formed at timestamp T given a snapshot of a social network at timestamp $T-1$. However, unlike classical link prediction problems in academic settings, friend recommendation in online networks operate on hundreds of millions or even billions of users, and evaluating link likelihood between every pair of users is computationally infeasible. Thus, in practice, friend recommendation is often formulated as an industrial recommendation problem and follows a typical large-scale recommendation system architecture [4] which consists of two tiers: *Retrieval* and *Ranking*.

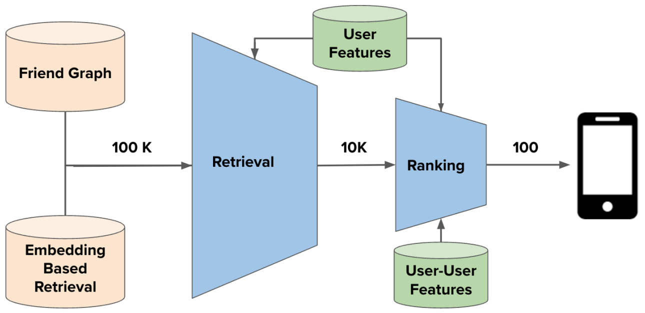

Figure 1: Friend recommendation architecture. We adopt a two-tier setup of coarse-grained retrieval (using FoF and EBR sources), and fine-grained ranking with an ML model. The funnel allows us to prune the space of all friend candidates to a highly personalized subset for each user.

Figure 1 demonstrates the friend recommendation funnel where candidates are first retrieved and then ranked before getting surfaced to the end users. In the *retrieval* phase, we extract friend candidates of a user using both Friends-of-Friends (FoF) approach and Embedding Based Retrieval (EBR). FoF fetches candidates from one's friend graph and EBR fetches candidates using one's user embedding. The number of candidates is often in the hundreds of thousands. These candidates are then ranked based on either heuristics or lightweight machine learning (ML) models, and roughly the top ten-thousand candidates are funneled to the next phase of ranking. The goal of the retrieval phase is to include as many potential friends as possible to the next phase and recall is used to measure its effectiveness (i.e. the ratio of the number of relevant candidates retrieved to the number of all relevant candidates). In the *ranking* phase, we rank retrieved candidates with a heavier deep neural network machine learning (ML) model, and send the top-K friend recommendations to the end users where K is in the order of hundreds. The ranker ML model is trained on historical friend recommendation outcomes and uses both user-level and user-to-user interaction features as signals to rank friend candidates. Different from the retrieval phase, the goal of ranking phase is to maximize precision, which measures the fraction of candidates surfaced to the users which they find relevant and worthwhile to friend request. This two-tier recommendation system advantageously sidesteps quadratic complexity from considering all user-user pairs, while also enabling flexible trade-offs in recall, precision and infrastructure cost.

2 EMBEDDING BASED RETRIEVAL

Retrieval in friend recommendation systems based on homogeneous graphs pose unique challenges as compared to product or content recommendations which are often based on heterogeneous graphs, e.g. video [4], content [35] and movie [10] recommendations. Friend retrieval is highly personalized and the popular retrieval approaches based on user demographic and historical engagement fail to work [27].

Graph traversal approaches (e.g. FoF [3]) have been used extensively in tackling this problem and are adopted widely in online social and professional networks such as LinkedIn [23] and Facebook [8]. This approach extracts candidates using breadth-first traversal and is often built on an inverted index for fast querying [26]. While it has been proved to be effective, there are several challenges with this approach: Firstly, in low-density networks where user adoption is low and a social network has not been fully established, FoF approach faces cold-start problems and struggles to reach quality friend candidates. Secondly, due to small-world network properties [18], it is computationally expensive to go beyond 2-hop connections, e.g. fetching friend candidates 3-hops away requires one additional query from all 2-hop connections. This can drastically increase the latency of real-time recommendations [38]. Lastly, the FoF approach does not well-capture graph topology (e.g. the number of mutual friends) nor connection strength (e.g. the historical engagement between two users), therefore one has to resort to other mechanisms to find and rank the top-K candidates fed into the next stage of recommendation systems.

To address the challenges above, we proposed a new retrieval system for friend recommendation systems - *Embedding Based Retrieval (EBR)*. EBR has been widely used in content or product recommendation systems to retrieve relevant items from a large corpus, e.g. YouTube [4], Facebook [15], and Pinterest [32]. Low-dimensional embeddings of nodes in large graphs have proved to be useful, and these embeddings once learnt can then be fed to downstream prediction tasks, e.g. node classification, link prediction, and graph clustering [34]. In retrieval use cases, EBR is typically used by first generating embeddings for each user and item, then retrieving items whose embeddings are closest to the target user's embedding in the embedding space. In friend recommendation systems where we only have users, we first learn an embedding for each user from rich information about users and their friend connections, such that users who are likely to be friends are close in the embedding space. Then in online retrieval, we retrieve candidates that are nearest neighbors in the embedding space for the target user we are generating recommendations for. EBR addresses the limitations of graph traversal based retrieval in a few key ways: *(i)* the nearest-neighbor search lookup in embedding space runs very efficiently [25] and its time complexity is constant regardless of network density, and *(ii)* it can pull in high-quality candidates that can be an arbitrary number of hops away from the target user, and need not be reached through cumbersome graph traversal. Next, we describe how we used Graph Neural Networks [28] to generate user embeddings and infrastructure optimizations that allows us to support EBR for friend recommendation at Snapchat scale in a cost-effective way.

2.1 Graph-Aware User Embedding

User embedding quality is key for EBR's effectiveness. In order to embed users in the Snapchat friend graph (user-friends-user), we utilize Graph Neural Networks (GNNs), which have risen to prominence in recent years as a leading technique in learning rich representations over graph data [39]. GNNs extend convolutions to graphs, and are commonly used to learn representations on an attributed graph of the form $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the node-set of n nodes ($|\mathcal{V}| = n$), $\mathcal{E}$ is the edge-set ($\mathcal{E} \in \mathcal{V} \subseteq \mathcal{V}$), and $\mathbf{X} \in \mathbb{R}^{n\times f}$ is a provided feature (attribute) matrix of f features[1]. Advantageously, GNNs can be trained flexibly towards node-level, link-level and graph-level tasks in both supervised and self-supervised settings [13], and have shown success in a variety of industrial tasks including recommendation [27, 30, 35] and forecasting [5, 31]. Many conventional GNN architectures employ a message passing paradigm [9], in which nodes exchange messages over graph topology. These messages are transformed and aggregated to derive intermediate embeddings for nodes [24]. A single message passing iteration (layer) at step $k + 1$ can be written as

$$\boldsymbol{h}_u^{(k+1)} = \mathrm{UPD}^{(k)}\left(\boldsymbol{h}_u^{(k)}, \mathrm{AGG}^{(k)}(\{\boldsymbol{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right) \quad (1)$$

where $\boldsymbol{h}_u^{(k)}$ denotes the intermediate embedding of node u at step k and $\boldsymbol{h}_u^0 = \boldsymbol{x}_u$, $\mathcal{N}(u)$ denotes the neighboring nodes of node u, and UPD and AGG are learnable updating and (permutation-invariant) aggregation functions. Various choices of the UPD and AGG functions yield different flavors of GNN models.

In the EBR usecase, we employ a 2-layer GraphSAGE [12] model, in which AGG is a simple mean-pooling, and UPD is a vector concatenation followed by a linear layer, with each layer having non-shared trainable parameters. To train the model, we utilize a transductive link prediction training setup [11, 29], where edges in the graph are split into training, validation and test sets and validation and test edges are masked off during training, and we use training edges for both message passing and supervision. We employ a standard max-margin ranking loss commonly utilized for link prediction settings [29, 35]; the loss per positive edge sample (q, i)

[1]We exclude discussion of edge features for brevity.

can be written as

$$\mathcal{L}_{(q,i)} = \sum_{n \in \mathrm{NEG}(q)} \max(0, \mathrm{sim}(z_q, z_n) - \mathrm{sim}(z_q, z_i) + \Delta) \quad (2)$$

where z_u denotes the last-layer representation (w.l.o.g. for any node u), and $\mathrm{NEG}(q)$ denotes the set of negative samples (non-edges) for the anchor node q, i denotes a positive sample (true edge) from q, Δ is the margin hyperparameter, and sim is a differentiable similarity function. In practice, we use cosine similarity for $\mathrm{sim}(\cdot,\cdot)$ and employ random negative sampling for $\mathrm{NEG}(q)$. We tune dropout, learning rate, weight decay, and non-linearity choices in addition to the margin Δ and the number of negative samples per anchor node. We use the Adam optimizer [17], and early stop training based on plateauing validation loss.

Training GNNs at large-scale is non-trivial and poses severe scalability challenges [14, 37, 38]. We first preprocess the graph into a compressed sparse row (CSR) format to enable $O(1)$ neighborhood look-ups, and compactly serialize node features as well. In order to train our model, we employ a minibatch training scheme: specifically, we utilize an internal model training framework which employs a producer-consumer shared-memory setup, in which producers sample minibatches by issuing queries to our compact graph and feature objects, and put them on queues, and consumers read from these queues and train using GPUs. We use a single n1-highmem-96 Google Compute Engine VM with 96 CPUs, 624G memory, and 4 Nvidia p100 GPUs for training. We use the same producer-consumer setup for parallel inference with the trained model upon convergence, and measure offline ranking metrics such as Hit Rates and Mean Reciprocal Rank [21]. We operationalize daily user embedding generation for hundreds of millions of Snapchat users with this model training and inference setup using Apache Airflow[2]. We version our embeddings and monitor their quality tightly given offline ranking metric thresholds.

2.2 Infrastructure Optimizations for Approximate Nearest-Neighbor Search

EBR typically uses Approximate Nearest-Neighbor (ANN) search [1] for candidate generation. There are many open source libraries such as Hnswlib [25] and FAISS [16], that can perform ANN search efficiently. At Snapchat, our candidate generation systems operate at a large scale (i.e. thousands of queries-per-second, or QpS) on a massive friend graph with billions of nodes, thus encoding the entire friend graph into a single index is undesirable. This is because a large index would typically lead to issues such as longer index load time, memory consumption issues, larger latencies, etc. One approach is to randomly partition a large index into multiple smaller indexes and have a federator layer that can fan queries out to all the smaller indexes and merge results. While this solution is feasible, we have developed a more cost-effective solution that capitalizes on some unique characteristics of Snapchat social graph.

First and foremost, we cut our index size down significantly by indexing only active users (e.g. users who are active on Snapchat at least once in the last 90 days) since the majority of friend links are established among active users. Even after this pruning, our index was still large with around 2 terabytes in size. Next, we

[2]https://airflow.apache.org/

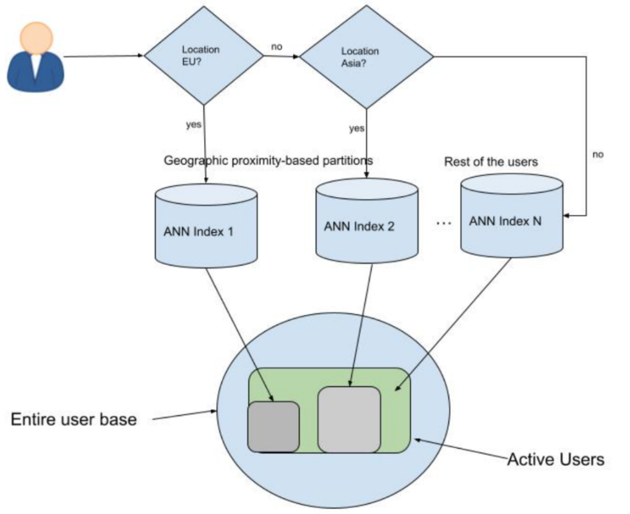

Figure 2: Embedding Query Workflow. We constrain ourselves to operations on only *active* users, greatly reducing ANN index size. We also geographically shard the index to leverage the geographic proximity of most friending behaviors. Both optimizations capture out-sized business value with massive infrastructure cost benefits.

exploit another aspect of our friend graph – a majority of friend links are established between friends who are geographically closer together. The exact geographic proximity that captures a majority of friend links varies across the world, e.g. Europe has a different proximity compared to North America or Asia. But the larger trend of friend links in relatively close proximity remains [33]. By taking advantage of this, we divided our index into a few segments based on geographic locations, such as North America, Europe, MENA, Asia, etc. This reduced the index size to a more manageable memory footprint. In addition, by avoiding the fan-out to all indexes and sending the requests to one of the geographic partitions based on searcher's location, we kept the query latency low while still capturing the majority of friending activities. Figure 2 illustrates the embedding query flow. The primary learning is that by taking advantage of certain characteristics of the friend graph, we can build a more scalable, robust, and cost effective system while realizing an outsized portion of the business impact.

3 RESULTS

We tested the effectiveness of EBR for friend recommendations on Snapchat using online A/B testing. The control group included candidates from retrieval algorithms in production including FoF and the treatment group included candidates from EBR as an additional retrieval source. We use the number of friendships made from friend recommendations as the success metric. Additionally, we ran AB tests in different markets to better understand its impact. The results discussed below are from the A/B tests that ran for 4 weeks with the significance level α set to be 0.01.

In the A/B tests, we saw statistically significant improvements for the treatments across all markets. The increases in the number of friendship made from friend recommendations are in the range of *5% to 10%* in different markets. We have also made the following observations. *(i)* EBR has a magnified impact in markets of higher user growth. *(ii)* The overlap of top suggestions from EBR and FoF is very low, making EBR a good complement to FoF in retrieval. *(iii)* The quality of candidates from EBR are on par with candidates from FoF in terms of conversion rate (i.e. impression to friend request) and reciprocation rate (i.e. friend request to reciprocation from the recipient).

4 CONCLUSION

In this paper, we introduced a novel retrieval system *Embedding Based Retrieval (EBR)* as a complementary source to the traditional graph traversal based retrieval for friend recommendation systems at Snapchat. To deploy EBR in production and support real-time friend recommendation to hundreds of millions of active users, we describe the graph-aware user embedding system using Graph Neural Networks that scales to billions of users and the optimizations in the EBR infrastructure to support Snapchat scale. Through online A/B tests, we demonstrated the effectiveness of EBR and observed statistically significant improvements in the number of friendships made with EBR as an additional retrieval source in both low- and high-density network markets.

For future work, we plan to work in the following areas: *(i)* improve the quality of user embeddings in GNN models (e.g. incorporating other types of links, enriching node and edge features, testing different loss functions). *(ii)* generate multiple embeddings to capture multiple social contexts [7], and *(iii)* further optimize EBR infrastructure through quantization (e.g. reducing the embedding index) and sharding strategies.

ACKNOWLEDGMENTS

We thank Jason Yeung for building the infrastructure to support testing different embeddings, Kai Wang for building the initial proof-of-concept, Sushobhan Nayak and Diwakar Punjani for brainstorming ideas, Zihao Fan, Tong Zhao and Elham Ghazizadeh for improving the quality of embedding quality, and Tanvi Motwani, Leo Claassen and David Boyle for insightful feedback and discussions.

COMPANY PORTRAIT

Snap Inc. is a technology company. We believe the camera presents the greatest opportunity to improve the way people live and communicate. We contribute to human progress by empowering people to express themselves, live in the moment, learn about the world, and have fun together.

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] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM* 45, 6 (nov 1998), 891–923. https://doi.org/10.1145/293347.293348

[2] Danah M Boyd and Nicole B Ellison. 2007. Social network sites: Definition, history, and scholarship. *Journal of computer-mediated Communication* 13, 1 (2007), 210–230.

[3] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. 2009. Make New Friends, but Keep the Old: Recommending People on Social Networking Sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 201–210. https://doi.org/10.1145/1518701.1518735

[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*. New York, NY, USA.

[5] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. 2021. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3767–3776.

[6] David Elsweiler, Ian Ruthven, and Christopher Jones. 2007. Towards memory supporting personal information management tools. *Journal of the American Society for Information Science and Technology* 58, 7 (2007), 924–946.

[7] Alessandro Epasto and Bryan Perozzi. 2019. Is a Single Embedding Enough? Learning Node Representations that Capture Multiple Social Contexts. In *The World Wide Web Conference*. ACM. https://doi.org/10.1145/3308558.3313660

[8] Facebook. 2023. *Where do People You May Know suggestions come from on Facebook?* https://www.facebook.com/help/163810437015615

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[10] Carlos A. Gomez-Uribe and Neil Hunt. 2016. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.* 6, 4, Article 13 (dec 2016), 19 pages. https://doi.org/10.1145/2843948

[11] Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh Chawla, Neil Shah, and Tong Zhao. 2022. Linkless Link Prediction via Relational Distillation. *arXiv preprint arXiv:2210.05801* (2022).

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

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

[14] Xiaotian Han, Tong Zhao, Yozen Liu, Xia Hu, and Neil Shah. 2022. MLPInit: Embarrassingly Simple GNN Training Acceleration with MLP Initialization. *arXiv preprint arXiv:2210.00102* (2022).

[15] 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 amp; 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

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

[18] Jon Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. 163–170.

[19] Pei Lee, Laks V.S. Lakshmanan, Mitul Tiwari, and Sam Shah. 2014. Modeling Impression Discounting in Large-Scale Recommender Systems. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1837–1846. https://doi.org/10.1145/2623330.2623356

[20] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting Positive and Negative Links in Online Social Networks. In *Proceedings of the 19th International Conference on World Wide Web* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 641–650. https://doi.org/10.1145/1772690.1772756

[21] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On sampling top-k recommendation evaluation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2114–2124.

[22] David Liben-Nowell and Jon Kleinberg. 2003. The Link Prediction Problem for Social Networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 556–559. https://doi.org/10.1145/956863.956972

[23] Linkedin. 2023. *People You May Know Feature.* https://www.linkedin.com/help/linkedin/answer/a544682/people-you-may-know-feature?lang=en

[24] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1202–1211.

[25] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[26] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge, UK. http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[27] 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.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[29] William Shiao, Zhichun Guo, Tong Zhao, Evangelos E Papalexakis, Yozen Liu, and Neil Shah. 2023. Link Prediction with Non-Contrastive Learning. *ICLR* (2023).

[30] Xianfeng Tang, Yozen Liu, Xinran He, Suhang Wang, and Neil Shah. 2022. Friend story ranking with edge-contextual local graph convolutions. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1007–1015.

[31] 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 *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2269–2279.

[32] George Wu Tim Koh and Michael Mi. 2021. *Manas HNSW Realtime: Powering Realtime Embedding-Based Retrieval.* Retrieved Jan 22, 2021 from https://medium.com/pinterest-engineering/manas-hnsw-realtime-powering-realtime-embedding-based-retrieval-dc71dfd6afdd

[33] 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.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[35] 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 *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[36] Guillaume Saint Jacques YinYin Yu and Paul Matsiras. 2021. *How to build an effective professional network on LinkedIn: Some data-driven insights.* Retrieved April 2, 2021 from https://engineering.linkedin.com/blog/2021/professional-network-checklist

[37] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).

[38] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2021. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727* (2021).

[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.