A Trust-Based Dating Network Using Friends-of-Friends-of-Friends(FoFoF) Recommendation

Name: Surya Narayan Swamy
Tuid : 916616559
Email : surya.narayan.swamy@temple.edu

Abstract

The goal of dating recommendation systems is to assist users in meeting new people who are not in their immediate social circles. Conventional methods that depend on friends or proximity frequently produce predictable recommendations. In this project, we present a novel technique that finds users in the social network that are precisely three hops away from the target user and do not have any mutual friends. By taking into account only users who have checked in at the same locations, we further refine these candidates based on location similarity. We used the Brightkite dataset to test Graph Convolutional Networks (GCNs) and Graph Autoencoders (GAEs) for link prediction. With an F1 score of 0.78 and a ROC-AUC of 0.88, the GAE model produced noticeably better results, proving its capacity to recommend varied and significant matches.This hybrid system balances novelty and compatibility, and while built for dating, it can be applied to broader social recommendation systems as well.

1. Introduction

One of the main challenges in modern dating platforms is helping users discover connections that are both meaningful and new. Traditional recommendation systems often suggest people based on mutual friends or location. While these strategies can produce relevant matches, they usually reinforce social bubbles and limit the diversity of suggestions **[3]**.

To overcome this, we created a graph-based dating recommendation system. Our method focuses on suggesting users who are exactly three hops away in the social network and share no mutual friends with the target user. This "Friends-of-Friends-of-Friends" (FoFoF) technique helps introduce socially distant, yet not entirely random, connections.

To improve relevant suggestions, we filter the 3-hop candidates by location history, keeping only those who have checked in at least one place in common with the user. This hybrid method combines graph structure with the real world to generate better suggestions. Graph learning models such as Graph Convolutional Networks (GCNs) **[7]** and Graph Autoencoders (GAEs) **[1]** are particularly useful in these scenarios, as they can capture relationships in the network without relying on labeled data.

2. Related Work

Earlier work on friend recommendation used basic rules like counting common friends or looking at shared connections. For example, Liben-Nowell and Kleinberg **[9]** studied how to

predict links in social networks using measures like common neighbors. Similarly, Adamic and Adar **[10]** suggested giving more weight to shared friends who are less connected, which helps to make better suggestions. These methods are easy to use but often suggest people who are already close in the network, limiting the chance of discovering new connections especially in dating apps where users often want to meet people outside their usual circle.

To better understand user's positions in the network, newer models like DeepWalk **[11]** and node2vec **[2]** learn embeddings by simulating random walks on the graph. These embeddings help represent users in a way that reflects their network structure. However, these models lack depth and are not directly trained to predict new links.

Later, Graph Neural Networks (GNNs) were developed to solve this problem more effectively. Graph Convolutional Networks (GCNs) **[7]** use a node's neighbors to learn useful features, and SEAL **[8]** focuses on learning from the subgraph around each possible connection. Graph Autoencoders (GAEs), introduced by Kipf and Welling **[1]**, are especially good for link prediction because they are trained to rebuild the entire graph. A recent paper by Ma et al. **[6]** shows that a well-tuned GAE can work just as well as newer, more complex models, while using fewer resources.

In addition to network structure, real-world behavior like location history can also help. Cho et al. **[3]** found that people who check in at the same places often become friends, even if they are not already connected in the social network. This supports using location data to improve recommendation systems.

Our project is based on these ideas. We used a GAE model to suggest users who are three hops away in the network and apply a location filter so that only people who have checked in at similar places are recommended. This way, the system suggests users who are both socially distant and behaviorally similar.

3. Methodology

Our dating recommendation system is designed as a multi-step pipeline. First, we construct a user graph using friendship data from the Brightkite dataset. Next, we apply graph-based models to learn embeddings for each user. We then identify candidates who are exactly three hops away from the target user, filter them based on shared check-in locations, and rank them using a scoring method that combines structural and behavioral similarity.

The complete workflow of this system is illustrated in Figure 1 below. It shows how the data flows through each stage from graph construction and embedding generation to multi-level filtering and final user recommendation.

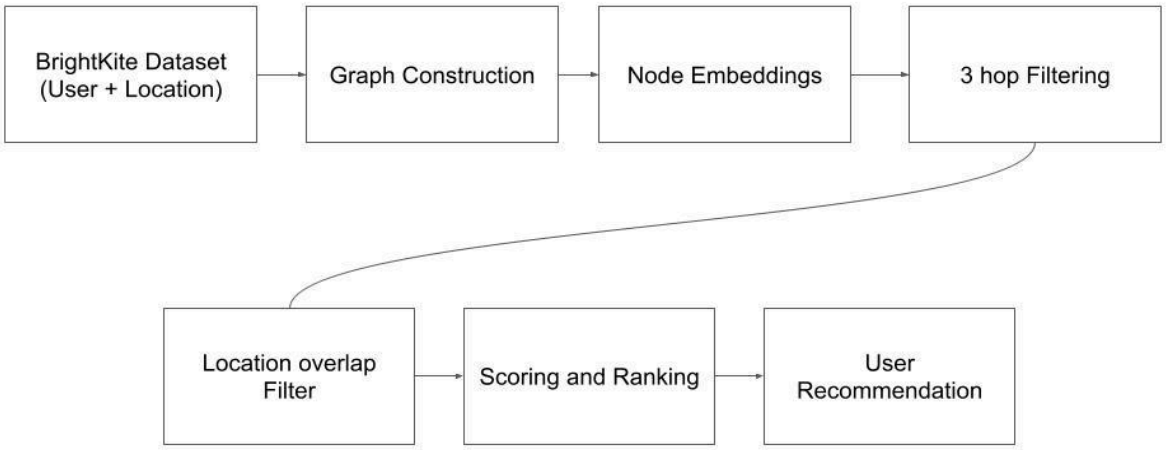

Figure 1: It outlines each stage of the recommendation process, starting from graph construction and embedding generation, followed by 3-hop filtering, location-based filtering, and finally, scoring and ranking to generate user recommendations.

3.1 Graph Construction

We constructed the user graph using the Brightkite social network dataset. In this graph:

- Nodes represent individual users
- Edges represent friendships between users

If two users are connected in the original dataset, we added an undirected edge between them in the graph. The final graph consists of 58,228 nodes and 214,078 edges. Any isolated nodes (users without any friends) were removed to ensure the network was connected and meaningful for modeling.

This graph structure forms the backbone of our recommendation system. It captures the social relationships needed to compute 3-hop connections and supports learning structured embeddings using graph neural networks.

3.2 Location Preprocessing

In addition to social connections, the Brightkite dataset contains check-in records for users, including timestamps and location IDs. We processed this data to build a set of visited locations for each user.
Specifically, we:

- Grouped the check-in data by user ID
- Extracted all unique location IDs each user visited
- Ignored timestamps, focusing only on location presence

These location sets were later used to filter recommendation candidates based on location overlap with the target user. We computed similarity between users using Jaccard similarity, which measures how many locations they have in common compared to their total visited locations.

3.3 GCN Embedding

We first trained a Graph Convolutional Network (GCN) to generate node embeddings. Since the dataset does not include detailed user attributes, we initialized all node features with random values. The GCN learns to update each node's representation by aggregating information from its neighbors.

Our goal was to use these embeddings to predict links between users, that is, to identify users who might be good candidates for connection, even if no edge exists between them. However, in practice, the GCN did not perform well for this task. The cosine similarity between connected and non connected users was nearly identical, indicating that the learned embeddings did not separate true relationships from random ones. This made GCN unsuitable for our recommendation needs.

3.4 Transition to Graph Autoencoder (GAE)

To improve the quality of recommendations, we switched to a Graph Autoencoder (GAE). Like the GCN, GAE uses graph convolution layers to encode the structure of the network into node embeddings. But unlike the GCN, it includes a decoder that tries to reconstruct the original graph.

This means the GAE learns to assign high similarity scores to pairs of users who are connected in the original graph, and low scores to unrelated users. We used the dot product of embeddings to score connections, and trained the model using a binary cross entropy loss between real and predicted edges. Because this task does not require labeled friend suggestions, it fits well with our unsupervised link prediction setup.

3.5 Three-Hop Filtering with Location Overlap

Once we generated the node embeddings, we applied a custom filtering process to narrow down the recommendation pool. Our goal was to find users who are both socially distant and behaviorally relevant. The filtering steps are:

1. Find all users who are exactly three hops away from the target user.
2. Exclude any users who are already direct friends or who share mutual friends with the target user.
3. Keep only the users who have visited at least one common location with the target user.

This creates a group of candidates who are socially new, but not random

3.6 Scoring and Ranking

After filtering, each candidate is ranked using a weighted score based on two types of 11 similarity:

- Embedding similarity: Calculated using cosine similarity between the embeddings of the target user and the candidate.
- Location similarity: Calculated using Jaccard similarity based on their sets of visited locations.

The final score is computed as:

$$\text{Final Score} = \alpha \cdot \text{Embedding Similarity} + (1-\alpha) \cdot \text{Location Similarity}$$

We used $\alpha=0.7$, which gives more importance to structural (embedding-based) similarity. This choice reflects our focus on recommending users who are socially meaningful but still behaviorally relevant.

4. Results

4.1 Link Prediction Performance

We first trained a Graph Convolutional Network (GCN) to generate node embeddings and evaluate potential connections between users. While the model produced visually interpretable clusters, its performance on link prediction was limited. The GCN achieved a ROC-AUC score of 0.63, and the average cosine similarity between true friends (0.99) and random user pairs (0.98) was nearly identical. This narrow gap indicated that the model failed to meaningfully distinguish between connected and unconnected users, suggesting overfitting and poor generalization.

To address this issue, we switched to a Graph Autoencoder (GAE), which is more suitable for unsupervised link prediction tasks. Unlike the GCN, the GAE is explicitly trained to reconstruct the graph structure by maximizing similarity between connected nodes and minimizing it for unrelated pairs. This makes the model inherently aligned with predicting unseen links. The GAE achieved significantly better results, as shown in Table 1:

Metric	GCN	GAE
ROC-AUC	0.63	0.88
F1 Score	0.65	0.78
Average Cosine similarity(Friends)	0.99	0.42
Average Cosine similarity(Random)	0.98	0.20
Cosine Similarity Gap(Derived)	0.01	0.22

Table 1: Performance Comparison of GCN and GAE Across Key Metrics

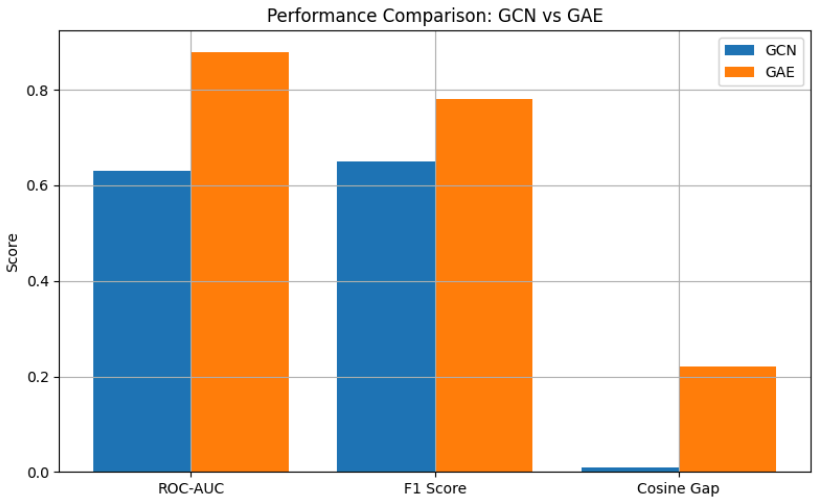

Figure 2: Bar chart comparing GCN and GAE on ROC-AUC, F1 Score, and Cosine Similarity Gap. GAE demonstrates consistently superior performance across all metrics.

These results confirm that the GAE embeddings better capture structural relationships in the graph. The wider cosine similarity gap between friends and non-friends suggests that the GAE learned to meaningfully separate users crucial for generating high quality dating recommendations.

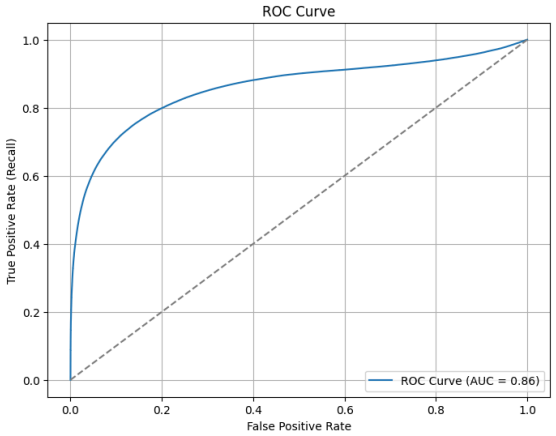

Figure 3: ROC curve for the GAE model. The area under the curve (AUC = 0.88) indicates strong link prediction capability.

4.2 Classification Metrics

To further evaluate performance, we framed the link prediction task as a binary classification problem. User pairs were labeled as either known friends or random non connections. Using a threshold of 0.5 on the decoder's dot product similarity score, the GAE achieved the following metrics:

- Accuracy: 80%
- Precision: 0.89
- Recall: 0.69
- F1 Score: 0.78

These results confirm that the model can reliably identify meaningful links while maintaining a balanced trade off between precision and recall, even in the absence of labeled supervision.

To better understand the nature of these predictions, We also plotted a confusion matrix heatmap, shown below:

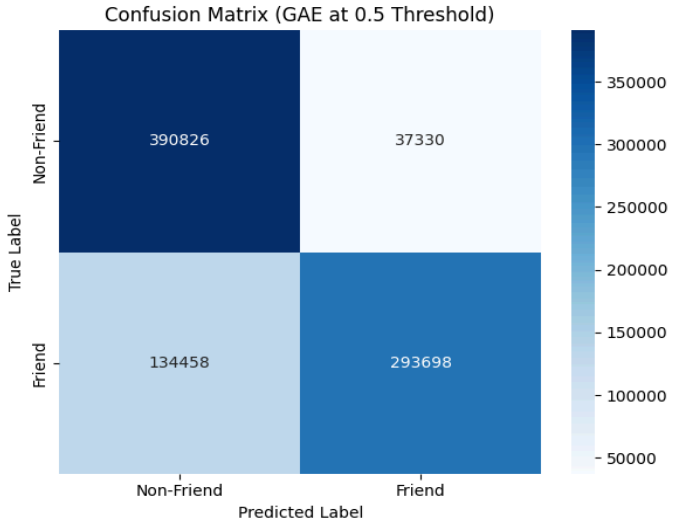

Figure 4: Confusion matrix heatmap showing predictions made by the GAE model at a 0.5 threshold. The rows represent actual labels (Friend or Non-Friend), while the columns represent the model's predictions. The model correctly predicted 390,826 non-friend pairs and 293,698 friend pairs, while 37,330 false positives and 134,458 false negatives were observed.

These results show that while the model is highly precise (relatively few false positives), it still misses a portion of true friendships (moderate number of false negatives). This is typical in sparse graph settings where many potential links are unobserved. However, the overall distribution supports the model's strong link prediction ability and clear separation between connected and unconnected user pairs.

Since this project uses unsupervised learning on a dataset with no labeled friend suggestions, we relied on multiple evaluation metrics to provide a well-rounded view of model performance. Each metric was chosen to capture a different aspect of the system: ROC-AUC reflects general ranking ability, cosine similarity helps assess the quality of learned embeddings, F1 score provides threshold-specific effectiveness, and the confusion matrix helps visualize the distribution of prediction outcomes. Although this may appear extensive, it is necessary to ensure the model behaves reliably across different evaluation perspectives, especially when working without ground-truth supervision.

4.3 Visualization and Filtering

To visualize how the system filters and recommends users, We created a 3-hop ego network for a sample user (User 58205). In the graph, nodes are color-coded based on their distance from the target user:

- Red: Target user
- Green: 1-hop (direct friends)
- Blue: 2-hop users
- Purple: 3-hop users
- Gold: Final recommendations (3-hop users with at least one shared check-in location)

Only users who were exactly three hops away, shared no mutual friends, and had at least one overlapping location were retained as candidates. This visualization helps demonstrate the filtering logic applied before scoring and ranking.

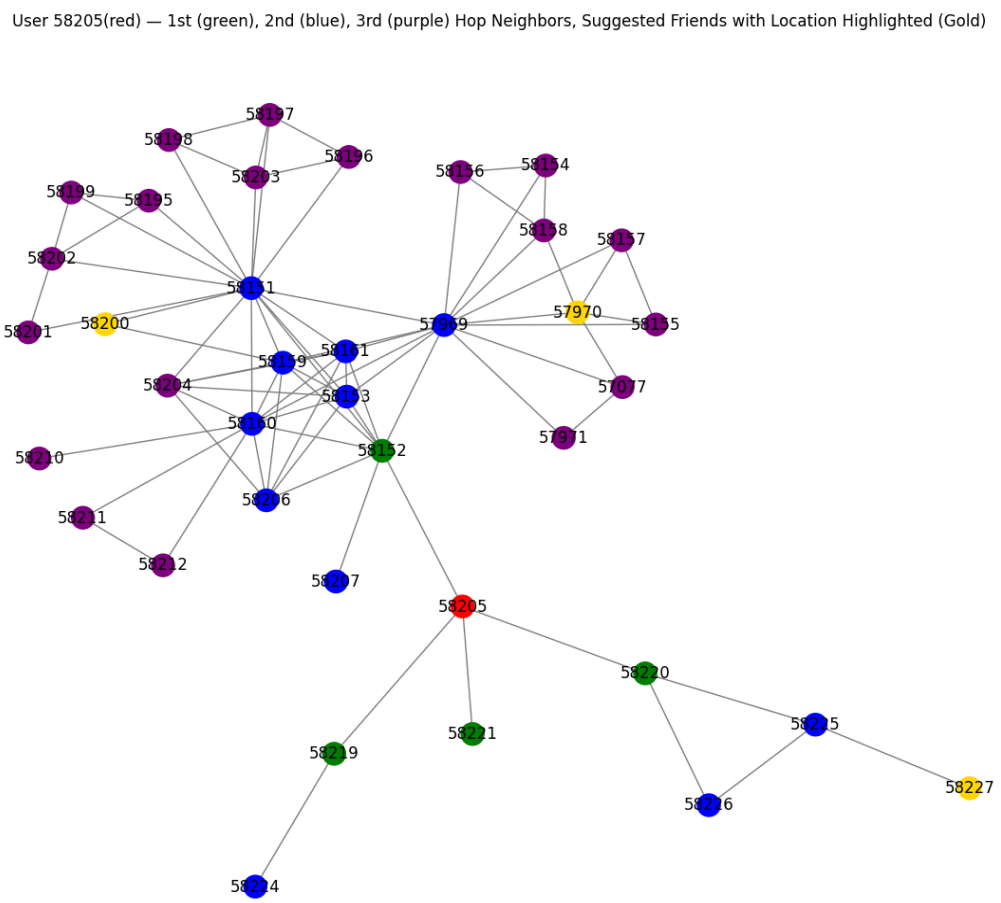

Figure 5: 3-hop ego network of User 58205, with color-coded hop levels and final recommendations shown in gold.

To illustrate the final ranking step, We also generated a card-style output for User 1, displaying the top five suggested users. These candidates are ranked using the weighted final score that combines embedding similarity and location overlap.

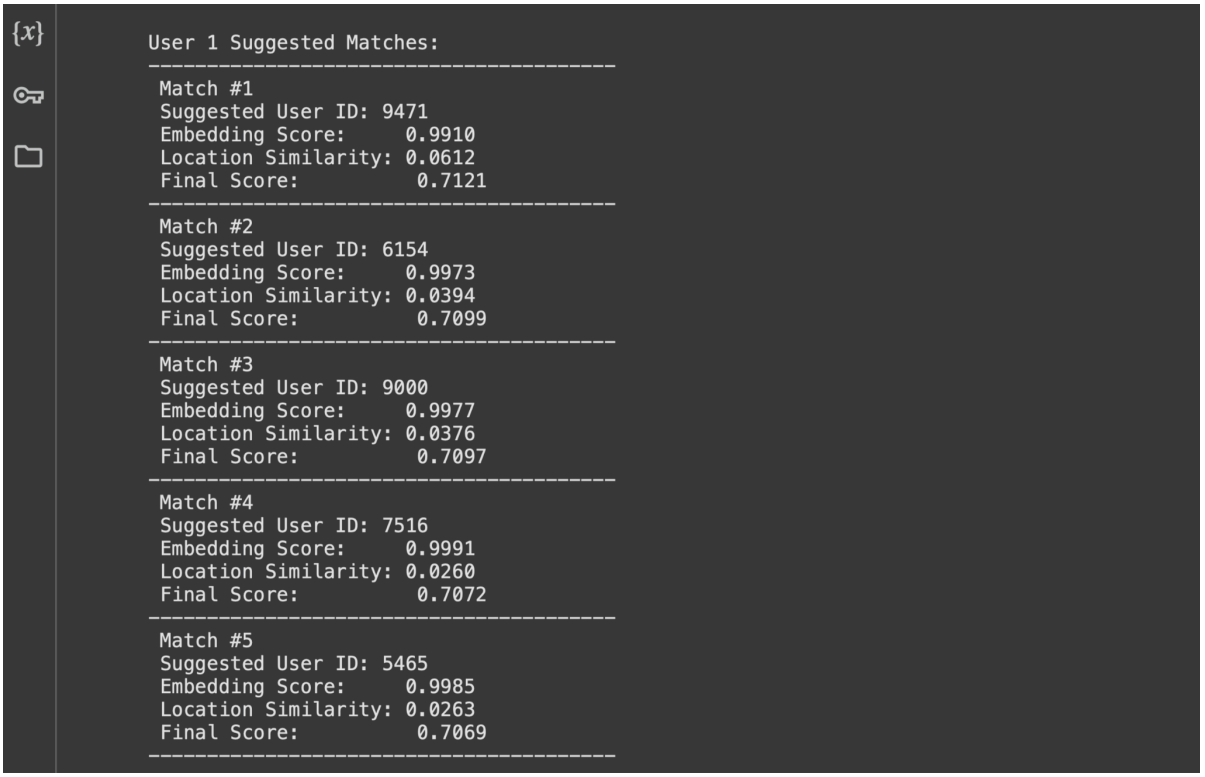

Figure 6: Card-style recommendation output for User 1. The highest-ranked user (User 9471) had a final score of 0.7078, based on an embedding similarity of 0.9849 and a location similarity of 0.0612. The remaining candidates also had high scores, indicating strong confidence in their relevance as recommendations.

4.4 Summary of Model Performance

As shown in Table 1 and Figure 1, the Graph Autoencoder (GAE) consistently outperformed the Graph Convolutional Network (GCN) across all evaluation metrics. The most notable improvement was the cosine similarity gap, which increased from 0.01 (GCN) to 0.22 (GAE), indicating that GAE embeddings were far better at distinguishing true connections from random pairs.

These findings confirm that GAE is more effective for unsupervised link prediction in sparse social graphs. The model's ability to generate meaningful embeddings, even without labeled data, makes it a strong foundation for building behavior-aware friend or dating recommendation systems.

5. Discussion

One of the most impactful decisions in this project was switching from a Graph Convolutional Network (GCN) to a Graph Autoencoder (GAE). While the GCN generated basic structural embeddings, it failed to distinguish well between connected and unconnected

users in an unsupervised setting. The GAE, on the other hand, is designed specifically for link prediction and showed better performance. Its architecture combines a graph convolutional encoder with a dot product decoder aligned naturally with the similarity-based scoring used in this system.

The addition of location overlap as a filtering step added a valuable behavioral dimension to what would otherwise be a purely structural model. This allowed the system to recommend users who were not only socially distant (exactly three hops away) but also behaviorally aligned. Finding the right balance between graph-based similarity and location based relevance was important; too much weight on location risked recommending users with only superficial overlap, while too little would weaken contextual relevance.

A key challenge was the processing time and complexity involved in generating and filtering 3-hop neighborhoods, especially during visualization and ranking. We addressed this by optimizing the graph layout and focusing on smaller ego networks to reduce overhead.

Another limitation involved users with sparse activity or fewer connections. For such users, the system naturally produces fewer or no suggestions. As future work, this limitation could be addressed by introducing collaborative filtering techniques to recommend users based on similarities with others who have richer interaction histories. This would allow the system to extend recommendations even when location or 3-hop graph information is limited.

Although this system was built for dating style recommendations, the core method combining graph structure and location-based filtering can be adapted for other types of social networks as well. It could be used for friend suggestions, community building, or interest-based networking platforms where meaningful but fresh connections are desired.

6. Acknowledgements

I would like to thank Professor Zoran Obradovic for his guidance throughout the semester. I am also grateful to his team for their valuable feedback during my project presentation and to my classmates who provided helpful suggestions during the development and testing phases.

7. References

1. Kipf, T. N., & Welling, M. (2016). *Variational Graph Auto-Encoders*. NIPS Workshop on Bayesian Deep Learning.
   Introduces the Variational Graph Autoencoder (VGAE), a foundational model for unsupervised learning on graph-structured data, particularly effective for link prediction tasks.
2. Grover, A., & Leskovec, J. (2016). *node2vec: Scalable Feature Learning for Networks*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
   Presents node2vec, an algorithmic framework for learning continuous feature representations for nodes in networks, balancing breadth-first and depth-first search strategies.
3. Cho, E., Myers, S. A., & Leskovec, J. (2011). *Friendship and mobility: User movement in location-based social networks*. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1082–1090). ACM.

Shows that shared location history is predictive of social connections, supporting hybrid recommendation methods.
4. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). *Graph Convolutional Neural Networks for Web-Scale Recommender Systems*. In *KDD*.
   Introduces PinSage, a scalable GCN-based recommendation system deployed at Pinterest, demonstrating the practical application of GCNs in large-scale settings.
5. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., & Yin, D. (2019). *Graph Neural Networks for Social Recommendation*. In *The World Wide Web Conference*.
   Proposes GraphRec, a GNN-based framework that effectively models user-item interactions and social relationships for improved recommendation accuracy.
6. Ma, W., Wang, Y., Wang, X., & Zhang, M. (2021). *Reconsidering the Performance of GAE in Link Prediction*. arXiv preprint arXiv:2411.03845.
   Demonstrates that with careful tuning, GAEs can perform as well as more complex models in link prediction tasks.
7. Kipf, T. N., & Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks*. In *ICLR*.
   Introduces the Graph Convolutional Network (GCN), a model that has become a cornerstone in graph-based machine learning tasks, including link prediction and node classification.
8. Zhang, M., & Chen, Y. (2018). *Link Prediction Based on Graph Neural Networks*. In *NeurIPS*.
   Presents SEAL, a framework that leverages GNNs for link prediction by learning from subgraph structures, achieving state-of-the-art results.
9. Liben-Nowell, D., & Kleinberg, J. (2007). *The Link-Prediction Problem for Social Networks*. *Journal of the American Society for Information Science and Technology*, 58(7), 1019–1031.
   A foundational work that formalizes the link prediction problem and evaluates various structural heuristics.
10. Adamic, L. A., & Adar, E. (2003). *Friends and Neighbors on the Web*. *Social Networks*, 25(3), 211–230.
    Introduces the Adamic-Adar score, a heuristic for link prediction based on common neighbors weighted by inverse degree.
11. Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). *DeepWalk: Online Learning of Social Representations*. In *KDD*.
    Proposes DeepWalk, an early method for learning node embeddings through random walks, foundational in graph representation learning.

Supplemental Material

The complete project code, implementation, and evaluation results are available in the Google Colab notebook linked: [Project Code on Google Colab]

This includes data preprocessing, GAE and GCN training, visualization of 3-hop ego networks, and the scoring mechanism for final friend suggestions.