Third-Party Library Recommendations Through Robust Similarity Measures

Abhinav Jamwal[a] and Sandeep Kumar[b]
Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, India

Keywords: Third-Party Library Recommendation, Similarity Measures, Jaccard Similarity, Collaborative Filtering, App-Library Interactions.

Abstract: This research systematically investigates the impact of different similarity measurements on third-party library (TPL) recommendation systems. By assessing the metrics of average precision (MP), average recall (MR), average F1 score (MF), average reciprocal rank (MRR) and average precision (MAP) at different levels of sparsity, the research demonstrates the significant impact of similarity measurements on recommendation performance. Jaccard similarity consistently outperformed the measurements tested and performed better in low-order and high-order app-library interactions. Its ability to reduce the number of sparse data sets and achieve a balance between precision and recall makes it the optimal measurement for the TPL recommendation. Other measurements, such as Manhattan, Minkowski, Cosine, and Dice, exhibited limitations to a certain extent, most importantly under sparse conditions. This research provides a practical understanding of the strengths and weaknesses of similarity measurements, which provides a basis for optimizing the TPL recommendation system in practice.

1 INTRODUCTION

Rapid growth in mobile app development has further intensified competition, requiring developers to quickly publish and update apps to meet changing user demands[1]. Third-party libraries (TPLs) have become part of this process, increasing software quality and reducing development efforts (Li et al., 2021). Studies show that the average Android app on Google Play uses about 11.81 TPLs(He et al., 2020). However, the wide availability of TPLs makes it difficult to choose the most suitable libraries and combinations of libraries (Lamothe and Shang, 2020; Salza et al., 2020). Recommendation systems mitigate this limitation by allowing developers to effectively find suitable TPLs (Nguyen et al., 2020).

Collaborative filtering (CF) has achieved admirable performance in a variety of recommendation tasks (Kangas, 2002). Methods such as LibRec (Thung et al., 2013a) and CrossRec (Nguyen et al., 2020) leverage CF-based similarity approaches to suggest libraries. However, these approaches only rely on low-order interactions and overlook high-order relations between apps and libraries (He et al., 2020). Furthermore, the performance of these systems is highly dependent on the choice of similarity measures, which have a direct influence on the quality of the recommendation.

The role of similarity measures is well studied in machine learning and classification, particularly in K-Nearest Neighbors (KNN), where distance metrics significantly impact performance (Abu Alfeilat et al., 2019). Although optimization techniques exist for classification, their application in TPL recommendation remains underexplored. This paper systematically examines the impact of similarity measures on the TPL recommendation. Building on existing work in classification (Zhang, 2021) and TPL recommendation (Li et al., 2024), we analyze the effects of cosine, Jaccard, and Dice similarity on recommendation performance. Our study aims to:

- Identify the strengths and limitations of different similarity measures in the TPL recommendation.
- Evaluate your impact on low-order and high-order app-library interactions.
- Provide actionable insights to improve the accuracy and efficiency of TPL recommendation sys-

[a] https://orcid.org/0000-0002-0213-3590
[b] https://orcid.org/0000-0002-3250-4866
[1] https://www.statista.com/statistics/289418/

Jamwal, A. and Kumar, S.
Third-Party Library Recommendations Through Robust Similarity Measures.
DOI: 10.5220/0013366600003928
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 635-642
ISBN: 978-989-758-742-9; ISSN: 2184-4895

tems.

Our contributions include a comprehensive analysis of similarity measures and their impact on recommendation quality, which elevates the state of TPL recommender systems.

2 RELATED WORK

Third-party library (TPL) recommendation systems have gained attention for accelerating software development by assisting developers in selecting appropriate libraries. Early research focused on recommending specific APIs or program snippets within TPLs. Zheng et al. (Zheng et al., 2011) proposed an approach for API replacement during software development, while Thung et al. (Thung et al., 2013b) leveraged textual feature requests to suggest relevant API methods.

Beyond API recommendations, researchers have explored app-library usage patterns to recommend entire TPLs. Saied et al. (Saied and Sahraoui, 2016) developed COUPMiner, which combined client- and library-based mining to uncover trends in the use of app-library. Ouni et al. (Ouni et al., 2017) introduced LibFinder, which used semantic similarity between source codes and TPLs for library detection. More recently, Saied et al. (Saied et al., 2018) proposed LibCUP, a multilayer clustering approach for categorizing TPLs based on usage history.

Collaborative filtering (CF) has played a significant role in the recommendation of TPL. LibRec (Thung et al., 2013a) was among the first to apply CF to Java projects, integrating association rule mining. Nguyen et al. (Nguyen et al., 2020) introduced CrossRec, a CF-based approach for open-source software projects, while He et al. (He et al., 2020) developed LibSeek, which incorporated matrix factorization and adaptive weighting for diversified recommendations. However, these approaches primarily relied on low-order interactions, often overlooking high-order dependencies. To address this, GRec (Li et al., 2021) modeled the relationships between the application and the library as graphs, using graph neural networks (GNNs) to capture low- and high-order interactions, significantly enhancing the precision of the recommendation. HGNRec (Li et al., 2024) further refined this approach by decomposing the app-library graph into two homogeneous graphs for efficient aggregation of interaction patterns.

Despite advances in TPL recommendation, the impact of similarity measures on TPL recommendations remains underexplored. This work bridges this gap by systematically evaluating their effects on TPL prediction and recommendation, drawing insights from similarity-based classification techniques.

3 MOTIVATING EXAMPLE

Figure 1 illustrates the role of similarity measures in interactions between the app and the library. The leftmost graph represents an *interaction graph*, where apps (`A1`–`A6`) are connected to libraries (`L1`–`L5`) based on usage, with edges indicating app-library utilization. The middle graph highlights *Similar Apps* (e.g., `A1`, `A2`, and `A5`), identified using measures like Jaccard or Cosine, reflecting shared library usage. The rightmost graph identifies *Similar Libraries* (e.g., `L1`, `L2`, and `L4`), representing frequently used libraries across different apps. These similarity patterns improve the recommendation process by identifying libraries relevant to a given application.

This example demonstrates the importance of similarity measures in capturing both low-order (direct) and high-order (indirect) interactions within app-library graphs:

- Low-order interactions: Libraries directly connected to an app (e.g., `L1` and `L2` for `A1`) are primary recommendation candidates.
- High-order interactions: Libraries connected via multiple hops (e.g., `L4` and `L5`) capture broader patterns of co-usage, providing additional recommendations.

Despite their importance in the TPL recommendation, the impact of similarity measures remains underexplored. This study investigates their role in shaping app-library relationships and improving recommendation effectiveness.

4 METHODOLOGY

The effectiveness of third-party library (TPL) recommendation relies on accurately identifying libraries most likely to be adopted by an application based on app-library interaction patterns. This study integrates K-Nearest Neighbor (KNN) similarity measures with Graph Neural Networks (GNN) to analyze the impact of different distance measures on recommendation performance, addressing the gap in understanding their role in TPL recommendations.

At the core of this methodology is the app-library interaction matrix, a binary representation that indicates whether an application has utilized a particular library. From this matrix, App Similarity Matrices and Library Similarity Matrices are constructed

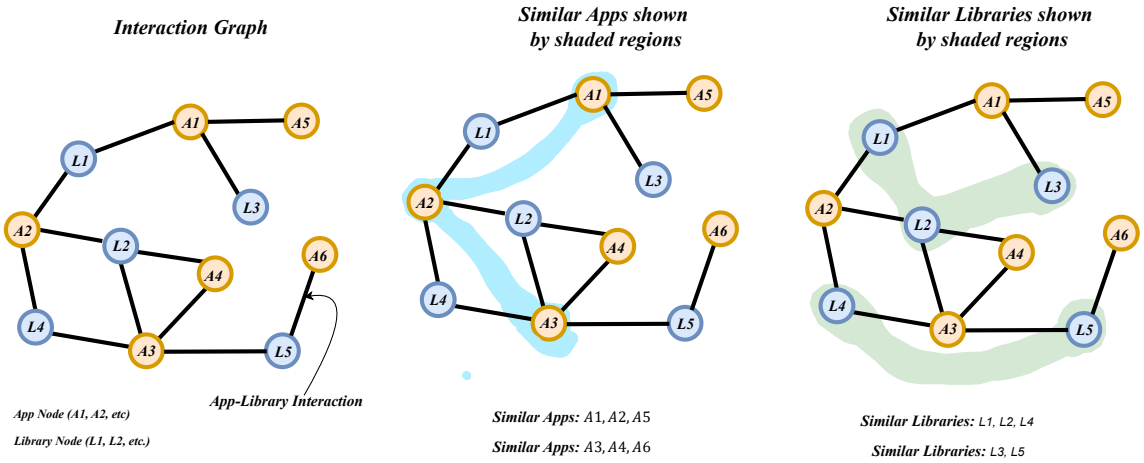


Figure 1: Visualization of App-Library Interactions. The interaction graph (left) shows connections between apps and libraries. Similar apps (middle) and similar libraries (right) are identified using KNN similarity, highlighted by shaded regions.

using Jaccard, Cosine, Dice, Minkowski, and Manhattan similarity measures. As shown in Figure 2, these matrices facilitate the identification of neighbouring applications and libraries based on shared interaction patterns. Recognizing such neighbors is essential to generate accurate and relevant recommendations, which form the foundation of the similarity-driven recommendation process.

4.1 Impact of Similarity Measures

To quantify proximity between apps and libraries, we employ similarity measures such as Jaccard, Cosine, Dice, Minkowski, and Manhattan. These measures construct neighbour matrices that capture interactions between the app and the library, which form the basis for structural analysis, as shown in Figure 2. Jaccard similarity evaluates shared elements between two sets, emphasizing common library usage, while cosine similarity measures the alignment of interaction vectors. The similarity of the slices, a variant of Jaccard, assigns greater weight to the shared interactions. Manhattan distance computes absolute differences, and the Minkowski distance generalizes the Manhattan and Euclidean distances by adjusting a sensitivity parameter. These measures identify app and library neighbors, providing structural information on app-library relationships.

The proposed framework integrates these similarity measures into a GNN-based recommendation pipeline, as shown in Figure 3. Interaction data are preprocessed and split into training and testing sets, with similarity-based neighbour identification enhancing the app-library interaction graph. The GNN propagates information through this augmented graph, leveraging both direct interactions and similarity-based neighbourhoods to learn node embeddings. These embeddings encode the relationship between the application and the library, enabling ranked library recommendations. The framework is evaluated using precision, recall, F1 score, mean average precision (MAP), and mean reciprocal rank (MRR) to systematically assess recommendation quality, ranking effectiveness, and retrieval performance. This approach highlights the role of similarity measures in improving the accuracy of the recommendation and capturing structural dependencies.

5 EXPERIMENTAL SETUP AND RESULTS

This section presents the experimental setup used to evaluate the framework, followed by the results obtained. The evaluation is carried out on the publicly available MALib-Dataset[2], specifically designed for third-party library (TPL) recommendation tasks. The data set consists of 31,432 Android apps and 752 distinct TPLs, represented as nodes in a graph, with 537,011 app-library usage records forming the edges.

Constructed by collecting 61,722 Android applications from Google Play via AndroidZoo[3], the dataset captures app-library interactions, allowing a comprehensive analysis of the use of third-party libraries (TPL). To ensure accuracy, TPL classifications were cross-verified with libraries available in Maven and GitHub, expanding the data set to 827 distinct TPLs and 725,502 app library usage records. With its extensive node and edge scale, the dataset supports

[2]https://github.com/malibdata/MALib-Dataset
[3]https://androzoo.uni.lu

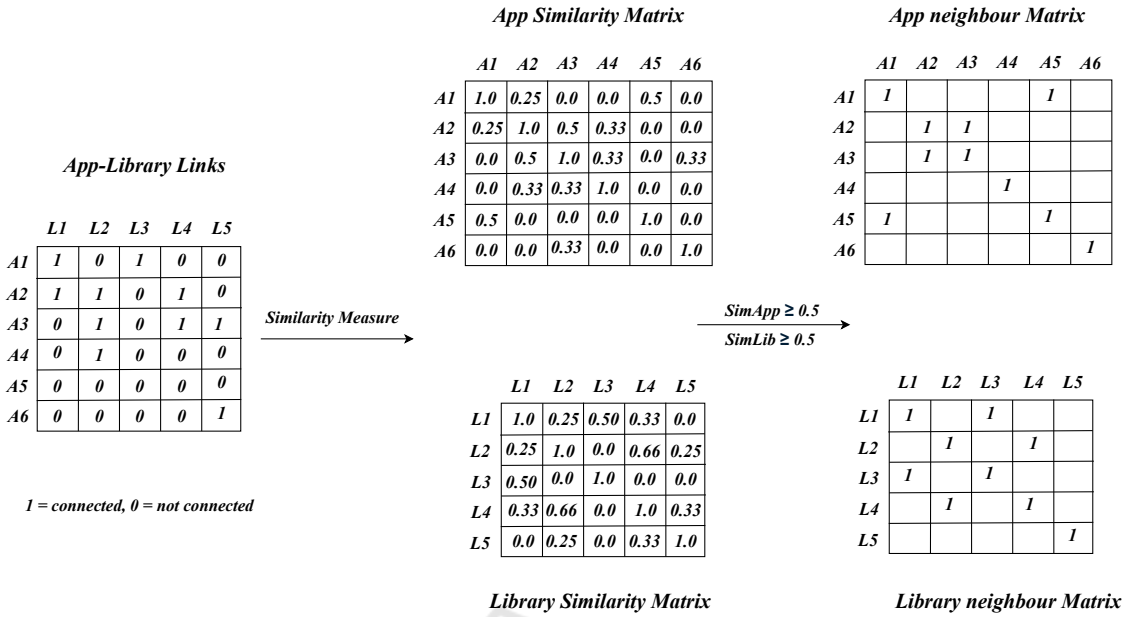


Figure 2: Illustration of identifying neighbouring nodes using similarity measures.

low- and high-order relationship modeling, making it well suited for graph-based recommendation methods. The publicly accessible data set[2] ensures reproducibility and validation by the research community.

5.1 Network Settings

Figure 3 illustrates the architecture of the GNN-based framework, integrating various KNN similarity measures (e.g., Jaccard, Cosine, Dice, Minkowski, Manhattan) to assess their impact on the TPL recommendation. The framework begins with the app-library interaction matrix, where KNN similarity is applied to construct app- and library-neighbor matrices, enhancing the structural information within the graph. For evaluation, a cross-validation approach is employed. Given an app u in the interaction matrix M, the rm libraries ($rm \in \{1,3,5\}$) are randomly removed to create an incomplete representation of the app, and the libraries removed serve as the ground truth. The model is trained on the remaining interactions and the number of selected neighbors k is varied to analyze its effect on performance. The evaluation is carried out using the metrics calculated at $K = [5,10,20]$, ensuring a comprehensive assessment of the similarity measures. This experimental setup enables a systematic evaluation of KNN similarity measures to improve the accuracy of the TPL recommendation.

5.2 Result Analysis

This study investigates the impact of different similarity measures, namely Cosine, Dice, Minkowski, Jaccard, and Manhattan, on TPL recommendation. The evaluation was carried out on datasets with varying levels of sparsity ($rm = 1,3,5$), where rm represents the number of libraries removed from the record of each app during the training phase. Metrics such as mean precision (MP), mean recall (MR), mean F1 score (MF), mean reciprocal rank (MRR) and mean average precision (MAP) were calculated at $K = 5,10,20$. The results are summarized in Table 1, and the trends in these metrics are visualized in Figure 4.

The results reveal that the Jaccard similarity consistently outperformed the other measures in multiple evaluation scenarios. For $rm = 1$, Jaccard achieved an MF of 0.263, an MRR of 0.646, and a MAP of 0.646 in $K = 5$, outperforming the Cosine, Dice, Minkowski, and Manhattan similarity measures. At $K = 10$ and $K = 20$, Jaccard maintained its superiority, demonstrating its robustness in accurately capturing shared interactions between the app and the library even at higher values of K. In contrast, cosine and Dice similarity measures struggled to achieve competitive performance, with lower MF values of 0.223 and 0.226 at $K = 5$, respectively. These results reflect their limitations in handling sparse data scenarios.

For $rm = 3$, the advantage of the Jaccard similarity became even more pronounced. It achieved an MF

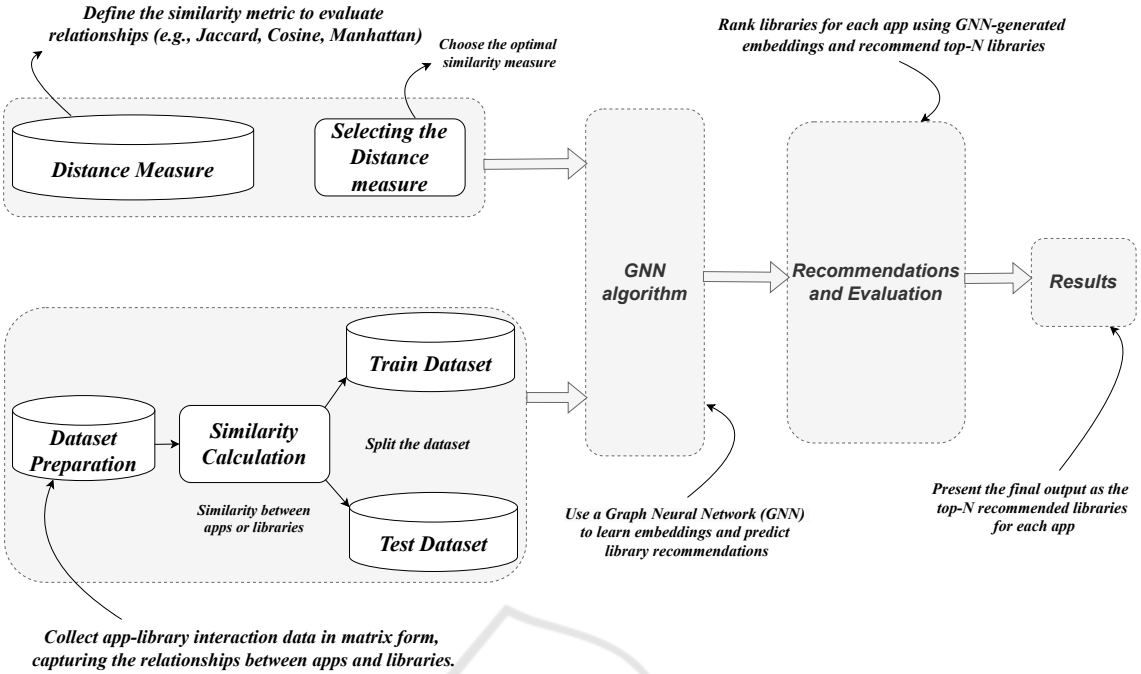


Figure 3: The framework of our experiments designed to evaluate the impact of various distance measures on the performance of the GNN-based recommendation system.

of 0.570 and an MAP of 0.841 at $K = 5$, which differentiates it as the most effective similarity measure in moderate sparsity. Minkowski and Manhattan measures showed competitive performance, with comparable MAP scores of $K = 10$ and $K = 20$. However, Jaccard consistently ranked higher in MF and MRR, highlighting its ability to balance precision and recall while maintaining consistency in ranking. The cosine and Dice measures continued to lag, particularly in recall and MAP, showing their limitations in effectively capturing app-library interactions.

At the highest level of sparsity, $rm = 5$, the Jaccard similarity once again emerged as the leading measure. It achieved an MF of 0.672 and a MAP of 0.879 at $K = 5$, underscoring its robustness even in highly sparse data sets. Manhattan and Minkowski's measures followed closely, delivering competitive results but falling slightly short in recall and ranking-based metrics. The dice and cosine measures struggled to adapt to the sparse setting, with their best MF values reaching only 0.599 and 0.598, respectively. The performance gap between Jaccard and other measures was more pronounced at $K = 20$, where Jaccard achieved superior MAP and MRR scores, demonstrating its effectiveness in retrieving relevant libraries even under challenging conditions.

The trends observed in Figure 4 further validate these findings. The Mean Precision (MP) curves indicate that Jaccard consistently achieved the highest MP, reflecting its ability to effectively prioritize relevant libraries. Similarly, the Mean Recall (MR) curves highlight the superior recall values achieved by the Jaccard and Manhattan measures, particularly at $K = 20$. This demonstrates their ability to capture a higher proportion of relevant libraries. The Mean F1 Score (MF) curves reinforce Jaccard's dominance, as its harmonic balance of precision and recall consistently exceeded that of other measures. The Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP) metrics further emphasize Jaccard's ability to rank relevant libraries higher in the recommendation list, ensuring that developers receive accurate and actionable recommendations.

The loss reduction curve shown in Figure 4(f) provides additional insights into the training dynamics of each similarity measure. Jaccard and Manhattan measures demonstrated the fastest and most stable loss reduction, reflecting their ability to effectively minimize errors and converge to an optimal solution. In contrast, the cosine and Dice measures exhibited slower convergence and less stable loss reduction, indicating challenges in learning high-quality embeddings under sparse conditions. The superior loss reduction performance of Jaccard and Manhattan highlights their efficiency in leveraging the structural and interaction information encoded in the app-library dataset.

The analysis demonstrates the critical role of similarity measures in influencing the quality of TPL

Table 1: Performance Comparison of Different Similarity Measures.

Dataset	Similarity	K = 5					K = 10					K = 20				
		MP	**MR**	**MF**	**MRR**	**MAP**	**MP**	**MR**	**MF**	**MRR**	**MAP**	**MP**	**MR**	**MF**	**MRR**	**MAP**
	Cosine	0.134	0.668	0.223	0.510	0.510	0.076	0.761	0.138	0.523	0.523	0.042	0.843	0.080	0.529	0.529
	Dice	0.136	0.679	0.226	0.524	0.524	0.077	0.773	0.141	0.536	0.536	0.043	0.853	0.081	0.542	0.542
rm=1	Manhattan	0.154	0.768	0.256	0.622	0.622	0.084	0.837	0.152	0.632	0.632	0.044	0.889	0.085	0.635	0.635
	Minkowski	0.154	0.769	0.256	0.625	0.625	0.084	0.840	0.153	0.634	0.634	0.044	0.888	0.085	0.638	0.638
	Jaccard	**0.158**	**0.789**	**0.263**	**0.646**	**0.646**	**0.085**	**0.854**	**0.155**	**0.655**	**0.655**	**0.045**	**0.907**	**0.086**	**0.658**	**0.658**
	Cosine	0.399	0.672	0.500	0.824	0.780	0.231	0.779	0.356	0.828	0.740	0.127	0.854	0.221	0.829	0.704
	Dice	0.400	0.675	0.502	0.826	0.781	0.233	0.784	0.358	0.830	0.739	0.128	0.860	0.222	0.832	0.703
rm=3	Manhattan	0.404	0.681	0.506	0.829	0.785	0.233	0.784	0.359	0.833	0.745	0.127	0.855	0.221	0.834	0.711
	Minkowski	0.406	0.684	0.509	0.827	0.782	0.233	0.785	0.359	0.830	0.744	0.127	0.856	0.221	0.832	0.711
	Jaccard	**0.454**	**0.766**	**0.570**	**0.882**	**0.841**	**0.250**	**0.843**	**0.386**	**0.885**	**0.811**	**0.133**	**0.894**	**0.231**	**0.886**	**0.785**
	Cosine	0.590	0.597	0.593	0.899	0.850	0.364	0.736	0.487	0.900	0.795	0.205	0.827	0.328	0.901	0.746
	Dice	0.595	0.603	0.599	0.894	0.848	0.365	0.739	0.489	0.896	0.795	0.205	0.830	0.329	0.896	0.746
rm=5	Manhattan	0.567	0.575	0.571	0.874	0.825	0.357	0.722	0.478	0.877	0.768	0.202	0.818	0.324	0.877	0.719
	Minkowski	0.566	0.573	0.569	0.872	0.824	0.356	0.720	0.476	0.875	0.767	0.202	0.817	0.324	0.876	0.718
	Jaccard	**0.668**	**0.676**	**0.672**	**0.916**	**0.879**	**0.393**	**0.795**	**0.526**	**0.918**	**0.834**	**0.214**	**0.865**	**0.343**	**0.918**	**0.796**

recommendations. Among the evaluated measures, the Jaccard similarity consistently emerged as the most effective, excelling in all metrics and K-values. Its robustness and reliability in capturing both low-order and high-order interactions make it the preferred choice for TPL recommendation tasks. Although Manhattan and Minkowski's measures offered competitive alternatives, their performance lagged behind Jaccard's. The findings emphasize the importance of selecting an appropriate similarity measure to ensure accurate and efficient recommendations, particularly in sparse data scenarios.

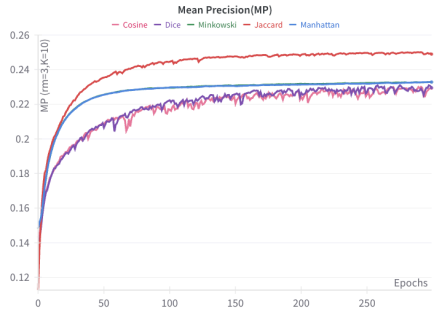
6 DISCUSSION

This study underscores the critical role of similarity measures in third-party library (TPL) recommendation systems. Among the measures evaluated, the Jaccard similarity consistently outperformed others in all metrics and values of K, as shown in Table 1 and Figure 4. Its ability to capture low-order (direct) and high-order (multi-hop) interactions makes it particularly effective in sparse datasets. The high mean F1 score (MF), the mean reciprocal rank (MRR), and the mean average precision (MAP) further highlight its balanced precision-recall trade-off and ranking effectiveness.

Manhattan and Minkowski measures performed well in moderately sparse scenarios ($rm = 3$) but showed reduced effectiveness in highly sparse settings ($rm = 5$). Cosine and Dice similarity, relying on vector alignment and weighted intersections, exhibited consistent limitations across all sparsity levels, particularly in capturing complex app-library interactions. Jaccard's efficiency in learning optimal embeddings and its faster convergence during training reinforce its suitability for the TPL recommendation. These findings emphasize the importance of selecting appropriate similarity measures to improve the precision and efficiency of recommendations.

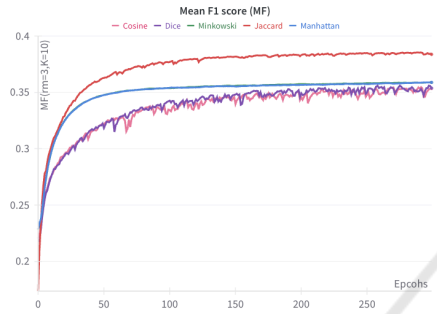
7 CONCLUSION

This study evaluated the impact of similarity measures on TPL recommendation, focusing on mean precision (MP), mean recall (MR), mean F1 score (MF), mean reciprocal rank (MRR) and mean average precision (MAP) at varying sparsity levels. The results show that Jaccard similarity is the most effective, capturing both low-order (direct) and high-order (multihop) app-library interactions. Its emphasis on shared interactions and balanced union enables superior performance in all metrics. Although Manhattan and Minkowski performed well in moderately sparse scenarios, they were outperformed by Jaccard in highly sparse datasets. The cosine and Dice measures struggled with sparse interaction matrices, resulting in lower performance. These insights highlight the strengths and limitations of different similarity measures, which guide the optimization of TPL recommendation systems. Future work may explore additional similarity measures and hybrid approaches to enhance adaptability across diverse datasets.

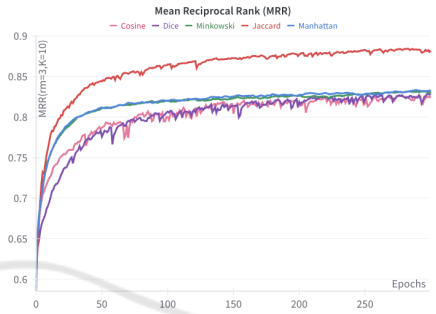


(a) Mean Precision (MP) for Different Similarity Measures.

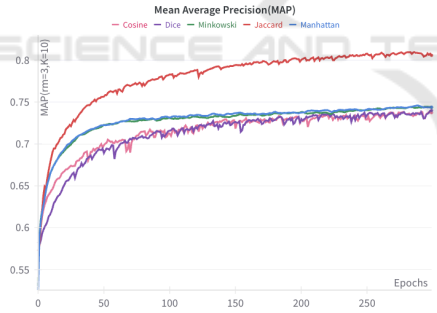
(b) Mean Recall (MR) for Different Similarity Measures.



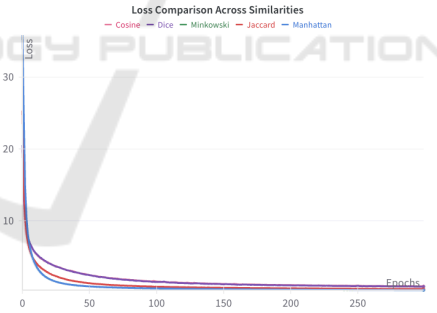
(c) Mean F1 Score (MF) for Different Similarity Measures.



(d) Mean Reciprocal Rank (MRR) for Different Similarity Measures.



(e) Mean Average Precision (MAP) for Different Similarity Measures.



(f) Overall Loss Reduction Across Similarity Measures.

Figure 4: Comparison of trends across different similarity measures, highlighting their performance and loss.

REFERENCES

Abu Alfeilat, H. A., Hassanat, A. B., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Eyal Salman, H. S., and Prasath, V. S. (2019). Effects of distance measure choice on k-nearest neighbor classifier performance: a review. *Big data*, 7(4):221–248.

He, Q., Li, B., Chen, F., Grundy, J., Xia, X., and Yang, Y. (2020). Diversified third-party library prediction for mobile app development. *IEEE Transactions on Software Engineering*, 48(1):150–165.

Kangas, S. (2002). Collaborative filtering and recommendation systems. *VTT information technology*, pages 18–20.

Lamothe, M. and Shang, W. (2020). When apis are intentionally bypassed: An exploratory study of api workarounds. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 912–924.

Li, B., He, Q., Chen, F., Xia, X., Li, L., Grundy, J., and

Yang, Y. (2021). Embedding app-library graph for neural third party library recommendation. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 466–477.

Li, D., Gao, Y., Wang, Z., Qiu, H., Liu, P., Xiong, Z., and Zhang, Z. (2024). Homogeneous graph neural networks for third-party library recommendation. *Information Processing & Management*, 61(6):103831.

Nguyen, P. T., Di Rocco, J., Di Ruscio, D., and Di Penta, M. (2020). Crossrec: Supporting software developers by recommending third-party libraries. *Journal of Systems and Software*, 161:110460.

Ouni, A., Kula, R. G., Kessentini, M., Ishio, T., German, D. M., and Inoue, K. (2017). Search-based software library recommendation using multi-objective optimization. *Information and Software Technology*, 83:55–75.

Saied, M. A., Ouni, A., Sahraoui, H., Kula, R. G., Inoue, K., and Lo, D. (2018). Improving reusability of software libraries through usage pattern mining. *Journal of Systems and Software*, 145:164–179.

Saied, M. A. and Sahraoui, H. (2016). A cooperative approach for combining client-based and library-based api usage pattern mining. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pages 1–10. IEEE.

Salza, P., Palomba, F., Di Nucci, D., De Lucia, A., and Ferrucci, F. (2020). Third-party libraries in mobile apps: When, how, and why developers update them. *Empirical Software Engineering*, 25:2341–2377.

Thung, F., Lo, D., and Lawall, J. (2013a). Automated library recommendation. In *2013 20th Working conference on reverse engineering (WCRE)*, pages 182–191. IEEE.

Thung, F., Wang, S., Lo, D., and Lawall, J. (2013b). Automatic recommendation of api methods from feature requests. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 290–300. IEEE.

Zhang, S. (2021). Challenges in knn classification. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4663–4675.

Zheng, W., Zhang, Q., and Lyu, M. (2011). Cross-library api recommendation using web search engines. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 480–483.