Supplementary Material for AtgBuilder: Feature-Assisted Graph Learning for Activity Transition Graph Construction with Seed Supervision

ANONYMOUS AUTHOR(S)

1 OVERVIEW

This document provides supplementary details referenced in the main paper, including: (1) the full LLM prompt template for summary generation; (2) additional ablation details on the GCN vs. GIN comparison; and (3) additional results on negative sampling strategies.

2 FULL PROMPT TEMPLATE FOR LLM-BASED SUMMARY GENERATION

This section provides the exact prompt templates used to generate one-sentence functionality summaries for activities and widgets. In the templates below, placeholders (e.g., `[MASK_ID]`, `[MASK_STRUCTURE]`) are instantiated with the corresponding metadata extracted for each activity/widget.

```
activity_prompt_template: |
  As an Android app tester, given an Activity's info:
  - Activity id: [MASK_ID]
  - Activity name: [MASK_NAME]
  - Detailed structure: [MASK_STRUCTURE]

  Succinctly and accurately summarize the core purpose of this Activity in one sentence.
  Your Response must follow the requirements:
  1. In English and in one concise sentence (<= 30 English words)
  2. Accurately reflects the main function/purpose
  3. No extra info/explanations
  4. Return as a pure JSON object wrapped in triple backticks (code block markers):
  ```{
    "activity_id": "[MASK_ID]",
    "activity_name": "[MASK_NAME]",
    "purpose": "YOUR_ANSWER"
  }```

retry_prompt_template: |
  The previous response did not meet the requirements. Please try again to summarize the core purpose of the
  ↪  Activity in one sentence, following the specified format and constraints:
  - Activity id: [MASK_ID]
  - Activity name: [MASK_NAME]
  - Detailed structure: [MASK_STRUCTURE]

  Succinctly and accurately summarize the core purpose of this Activity in one sentence.
  Your Response must follow the requirements:
  1. In English and in one concise sentence (<= 30 English words)
  2. Accurately reflects the main function/purpose
  3. No extra info/explanations
  4. Return as a pure JSON object wrapped in triple backticks (code block markers):
  ```{
    "activity_id": "[MASK_ID]",
    "activity_name": "[MASK_NAME]",
    "purpose": "YOUR_ANSWER"
  }```

widget_prompt_template: |
  As an Android app tester, given a Widget's info:
  - Widget id: [MASK_ID]
  - Widget type: [MASK_TYPE]
  - Widget structure: [MASK_CONTENT]

  Succinctly and accurately summarize the core purpose/function of this Widget in one sentence.
```

Author's address: Anonymous Author(s).

```
  Your Response must follow the requirements:
  1. In English and in one concise sentence (<= 30 English words)
  2. Accurately reflects the main function/purpose
  3. No extra info/explanations
  4. Return as a pure JSON object wrapped in triple backticks (code block markers):
  ```{
    "widget_id": "[MASK_ID]",
    "widget_type": "[MASK_TYPE]",
    "purpose": "YOUR_ANSWER"
  }```

retry_widget_prompt_template: |
  The previous response did not meet the requirements. Please try again to summarize the core purpose of the
  ↪  Activity in one sentence, following the specified format and constraints:
  - Widget id: [MASK_ID]
  - Widget type: [MASK_TYPE]
  - Widget structure: [MASK_CONTENT]

  Succinctly and accurately summarize the core purpose/function of this Widget in one sentence.
  Your Response must follow the requirements:
  1. In English and in one concise sentence (<= 30 English words)
  2. Accurately reflects the main function/purpose
  3. No extra info/explanations
  4. Return as a pure JSON object wrapped in triple backticks (code block markers):
  ```{
    "widget_id": "[MASK_ID]",
    "widget_type": "[MASK_TYPE]",
    "purpose": "YOUR_ANSWER"
  }```
```

These prompts are used uniformly across apps to encourage consistent, concise summaries for subsequent embedding and feature fusion.

3 ADDITIONAL COMPARISON DETAILS: GCN VS. GIN ENCODER

This section provides the encoder comparison removed from the main paper due to page limits. Table 1 presents the results of the two GNN encoders (GCN and GIN), under two thresholds (@0.5 and @t^*). Based on the results, we have the following observations:

- **GIN consistently outperformed GCN across both thresholds.** Under @0.5, GIN attained a higher F1-score (0.9072) than GCN (0.8555), with gains in both precision and recall. Under @t^*, GIN reached 0.9158 F1-score, while GCN remained around 0.855. This trend was as expected, because GIN has greater discriminative ability than GCN in distinguishing graph structures [4]. Unlike GCN, which relies on linear neighborhood smoothing [1], GIN applies an MLP-based update after aggregation [4], which helps preserve discriminative signals and mitigates against over-smoothing [2, 3]. Transitions in ATGs are typically hub-dominated: (1) Many-to-one navigation means that many activities converge to a small set of high in-degree hub activities (e.g., "back" flows to `HomeActivity`); and (2) one-to-many navigation means that a hub activity can have many outgoing transitions to diverse functional targets (e.g., a `HomeActivity` linking to multiple feature pages). Over-smoothing in GCN could blur subtle differences among activities connected to the same hubs, reducing link-prediction separability.
- **GIN yielded a higher validation-selected threshold than GCN.** The selected t^* for GIN was higher (≈ 0.4535) than that for GCN (≈ 0.3930). This indicates that GIN could use a stricter decision threshold when converting predicted probabilities into binary transition predictions. Such a stricter threshold was consistent with better separation between the predicted probabilities of transition and non-transition edges: GIN could filter out more low-confidence edges (improving precision) while losing fewer true transitions (preserving recall) than GCN.

Table 1. RQ1.1: Effect of encoder selection.

ID	Encoder	@0.5			@t^*			
		Precision	Recall	F1-score	Precision	Recall	F1-score	t^* Value
E1	GCN	$0.8376_{\pm 0.0087}$	$0.8744_{\pm 0.0189}$	$0.8555_{\pm 0.0136}$	$0.8271_{\pm 0.0025}$	$0.8859_{\pm 0.0243}$	$0.8551_{\pm 0.0120}$	$0.3930_{\pm 0.0778}$
E2	GIN	$0.9057_{\pm 0.0121}$	$0.9139_{\pm 0.0508}$	$0.9072_{\pm 0.0321}$	$0.8995_{\pm 0.0131}$	$0.9330_{\pm 0.0230}$	$0.9158_{\pm 0.0177}$	$0.4535_{\pm 0.0423}$

Table 2. RQ1.5: Effect of negative-sampling strategies.

ID	Sampling Strategy	@0.5			@t^*			
		Precision	Recall	F1-score	Precision	Recall	F1-score	t^* Value
S1	No Sampling	$0.7614_{\pm 0.1406}$	$0.8234_{\pm 0.0380}$	$0.7883_{\pm 0.0964}$	$0.7628_{\pm 0.1377}$	$0.8194_{\pm 0.0452}$	$0.7876_{\pm 0.0973}$	$0.5241_{\pm 0.0512}$
S2	Random Sampling	$0.9199_{\pm 0.0181}$	$0.9425_{\pm 0.0202}$	$0.9310_{\pm 0.0191}$	$0.9115_{\pm 0.0220}$	$0.9543_{\pm 0.0143}$	$0.9323_{\pm 0.0184}$	$0.4166_{\pm 0.0253}$
S3	Hard Sampling	$0.9097_{\pm 0.0194}$	$0.9372_{\pm 0.0122}$	$0.9232_{\pm 0.0159}$	$0.9019_{\pm 0.0190}$	$0.9478_{\pm 0.0122}$	$0.9243_{\pm 0.0158}$	$0.4272_{\pm 0.0143}$

Summary of Answers to GCN and GIN comparison: Selecting an appropriate encoder for AtgBuilder (e.g., GIN) that effectively handles hub-dominated navigation graphs could improve the performance. Accordingly, the prediction-threshold selection should be treated as part of the model configuration, and should be tuned for the encoder (using a validation-selected t^*), rather than using a fixed value.

4 ADDITIONAL COMPARISON DETAILS: NEGATIVE SAMPLING STRATEGIES

This section reports detailed results for negative-sampling strategies referenced in the main paper. Table 2 presents the results related to how different negative-sampling strategies affect the AtgBuilderperformance. Based on these results, we have the following observations:

- **Balancing positives and negatives is necessary for stable learning.** The performance without sampling (S1) was substantially worse, and unstable, across seeds. This was as expected, because transition prediction across all activity pairs is a highly imbalanced problem: There are far more non-transition pairs than true transitions. As a result, training with all the negative samples can overwhelm the positive signal, and bias the model towards predicting no transition.
- **Random 1:1 sampling achieved the best overall performance.** AtgBuilder achieved the best results with random 1:1 positive-negative sampling (S2): F1-score 0.9310 under @0.5, and 0.9323 under @t^*. This suggests that balancing the positive and negative training samples led to better-separated and more stable score distributions, making the predicted probabilities more reliable for decision-making.
- **Hard negatives were competitive, but did not consistently improve over random sampling.** Hard sampling (S3) performed slightly worse than random sampling, with F1-scores of 0.9232 and 0.9243 under @0.5 and @t^*, respectively. A possible explanation for this is that same-source hard negatives are harder to distinguish from true transitions and thus more sensitive to unreliable static-seed supervision. Because many activities exhibit similar outgoing navigation patterns (e.g., a settings page linking to multiple sub-pages), negatives that share the same source can be hard to distinguish from mislabeled positives. As a result, the model may become more conservative and show higher variance across runs.

Summary of Answers to Different Negative Sampling Strategies: Negative sampling was essential for ATG-link prediction under extreme class imbalance. Random 1:1 sampling provided the most reliable overall performance and stability, while hard negatives were not consistently beneficial under noisy static supervision. We therefore recommend random balanced sampling as the default, and using hard negatives cautiously.

REFERENCES

[1] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[2] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*. 3538–3545.

[3] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*.

[4] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*.