Z-REx: Human-Interpretable GNN Explanations for Real Estate Recommendations

Kunal Mukherjee *
kunal.mukherjee@utdallas.edu
The University of Texas at Dallas
Richardson, Texas, USA

Zachary Harrison
zacharyha@zillowgroup.com
Zillow Group, Inc
Seattle, Washington, USA

Saeid Balaneshin
saeidb@zillowgroup.com
Zillow Group, Inc
Seattle, Washington, USA

ABSTRACT

Transparency and interpretability are crucial for enhancing customer confidence and user engagement, especially when dealing with black-box Machine Learning (ML)-based recommendation systems. Modern recommendation systems leverage Graph Neural Network (GNN) due to their ability to produce high-quality recommendations in terms of both relevance and diversity. Therefore, the explainability of GNN is especially important for Link Prediction (LP) tasks since recommending relevant items can be viewed as predicting links between users and items. GNN explainability has been a well-studied field, but existing methods primarily focus on node or graph-level tasks, leaving a gap in LP explanation techniques.

This work introduces Z-REx, a GNN explanation framework designed explicitly for heterogeneous link prediction tasks. Z-REx utilizes structural and attribute perturbation to identify critical substructures and important features while reducing the search space by leveraging domain-specific knowledge. In our experimentation, we show the efficacy of Z-REx in generating contextually relevant and human-interpretable explanations for ZiGNN, a GNN-based recommendation engine, using a real-world real-estate dataset from Zillow Group, Inc. We compare against State-of-The-Art (SOTA) GNN explainers to show Z-REx outperforms them by 61% in Fidelity metric by producing superior human-interpretable explanations.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Model Transparency, Model Explanation, Graph Neural Networks, Link Prediction, Recommendation

1 INTRODUCTION

As user interactions in housing marketplaces grow in volume and complexity, personalized recommendation systems are becoming indispensable for improving user experience and satisfaction. Recent advancements in data modeling and recommendation algorithms enable the creation of detailed interaction graphs, that capture nuanced relationships between users and items (*e.g.,* property listings and cities). Leveraging the inherent structure of these graphs provides fine-grained insights into user behaviors and preferences, forming a strong foundation for advanced recommendation systems. Building on these insights, various learning-based recommendation systems have been developed and deployed to effectively utilize this data, significantly improving personalization and engagement rates across multiple domains.

In particular, GNNs have emerged as a powerful tool for modeling relational data in recommendation systems. GNNs leverage graph-structured data, where nodes represent entities (such as users, listings, and cities) and edges denote interactions (such as user views, saves, and tours). Through a message-passing mechanism, GNNs aggregate information from neighboring nodes and edges, enabling them to capture complex, multi-hop relationships within the graph. This capability makes GNNs particularly effective for personalized recommendations, as they can identify nuanced patterns in user preferences and item associations.

While GNN-based recommendation models achieve impressive accuracy, their black-box nature poses significant challenges regarding trust and transparency. Users and system administrators often require explanations for recommendations, particularly in high-value transactions or critical decision-making scenarios. Current GNN explanation techniques focus on node and graph-level tasks whereas recommendation systems model the recommendation problem as Link Prediction (LP) tasks, as recommending relevant items to users can be framed as predicting links between users and relevant items. Therefore, there is a notable gap in the literature concerning the explanation of LP tasks.

Existing GNN explanation techniques [1]–[3] primarily focus on generating explanations by either learning a mask to select an edge-induced subgraph or searching for the most informative subgraph. While these methods are effective for node-level and graph-level tasks, they face significant challenges in link-level prediction contexts. For instance, approaches like [1], [2] often yield disconnected edges or subgraphs, making the resulting explanations challenging to interpret about the predicted link. Furthermore, [3] suffers from scalability issues as the graph size increases exponentially, leading to computational challenges due to the combinatorial explosion of possible subgraphs. This is particularly problematic since enumerating all subgraph types requires checking for isomorphism, which results in exponential time complexity as the graph's vertices and edges grow. Customizing these techniques for sparse datasets and heterogeneous graphs introduces further difficulties. Existing methods emphasize dense local subsets of the dataset and are primarily designed for homogeneous graphs, limiting their applicability to more sparse and diverse graph structures.

*This work was done while he was interning at Zillow Group, Inc., during summer '24.

KDD '25 MLoG-GenAI, August 06, 2025, Toronto, ON, Canada

Explaining GNNs for LP introduces three unique challenges (also noted by [4]): *(1)* accurate interpretation of substructures in the presence of sparse relations, *(2)* scalability, and *(3)* heterogeneity. To address these challenges, we formulate LP explanation as an instance-level, post-hoc task that generates interpretable ground-truth aware explanations by identifying the most important feature subset and critical subgraphs. Our approach offers both interpretable and scalable explanations while also accounting for the heterogeneity of real-world recommendation systems. This provides a critical advancement in improving the transparency and trustworthiness of GNN-based recommendation systems.

We propose Z-REx, an instance-level GNN-based recommendation explanation framework for LP tasks using entire heterogeneous graphs. Z-REx leverages both graph structures and domain knowledge to identify critical structural relationships to provide explanations with better contextual relevance and alignment with ground truth. Our approach involves a two-step collaborative process where we perturb: *(1)* the features to identify the most important subset, and *(2)* the graph to determine the key subgraphs.

Our study emphasizes the importance of decoupling the decision-making process using both structural and entire feature sets, and instead focuses on using a subset of features and subgraphs to enhance the interpretability of recommendations. This allows us to create context-aware explanations that align closely with the interaction patterns found in recommendation systems. We stress that our framework, Z-REx, is designed to provide interpretable insights into GNN decisions, not to improve recommendation accuracy.

To demonstrate the effectiveness of Z-REx, we measure the quality of explanation for recommendations made by ZiGNN, a GNN-based recommendation engine for heterogeneous interaction graphs. We use a real-world real-estate dataset from Zillow Group, Inc. The dataset is collected from a large-scale recommendation platform, where user interactions are logged across a diverse set of listings and cities over an extended period. Finally, we compare it against SOTA GNN explainers, evaluating its ability to provide relevant explanations in recommendation tasks. Our evaluation shows that Z-REx outperforms existing explainers in terms of providing actionable and interpretable insights for both users and system administrators. This work highlights the potential for developing domain-specific features to close the gap between high-performing recommendation systems and their need for transparency.

In summary, the key contributions of our work are:

- To the best of our knowledge, this is the first end-to-end framework that integrates recommendation and instance-level explanation using whole graph structure without disintegrating into paths or subgraphs.
- We conduct an extensive analysis of the real-world housing market to uncover structural heuristics that reduce the explanation search space and improve the relevance of recommendation explanations.
- We propose a collaborative approach combining feature and structural perturbation, which enhances both the accuracy and diversity of recommendations and is validated through a real-world case study.
- Z-REx outperforms PaGE-Link by 29%, GNNExplainer by 85%, and SubgraphX by 70% in the Fidelity metric, demonstrating its superior explanation accuracy.

2 BACKGROUND AND RELATED WORKS

GNN-based Recommendation Framework. Graph Neural Networks (GNNs) have emerged as a powerful approach for enhancing recommender systems. Over the past few years, several studies have applied GNNs directly to user-item bipartite graphs, yielding significant improvements in both effectiveness and efficiency [5]–[9]. One common challenge in this approach lies in capturing higher-order connectivity between nodes. Multi-GCCF [10] and DGCF [11] address this by introducing artificial edges that connect two-hop neighbors (*e.g.,* user-user and item-item graphs) to introduce proximity information into the user-item interaction.

Node representations are computed layer-by-layer in GNNs, where the overall user and item representations are critical for downstream tasks like recommendation prediction. The most common practice is to adopt the final-layer embeddings as the ultimate representations [12], [13]. For a comprehensive survey of GNN-based recommender systems, readers are referred to [14].

NGCF [9] enhances feature interactions between users and items using an element-wise product operation. NIA-GCN [8], on the other hand, introduces pairwise neighborhood aggregation to better capture neighbor relationships. Inspired by GraphSAGE [15], [10], [12], [13] utilize a concatenation operation followed by non-linear transformations to update node representations. Conversely, LightGCN [16] and LR-GCCF [7] simplify the aggregation by removing non-linearities, which enhances both performance and computational efficiency.

GNN-based Explainers. Recent research in GNN explainers [1]–[3] has advanced in identifying key nodes, edges, or subgraphs in GNNs. They are categorized into white-box and black-box explainers. White-box methods, *e.g.,* GNNExplainer [1] and PGExplainer [2], access GNN internals, including model weights and gradients. Conversely, black-box methods like SubgraphX [3] operate on model inputs and outputs, reducing coupling between the explanation framework and model architecture. GNN explainers encounter exponentially increasing computation time with graph size growth, hindering the interpretability in real-world graphs.

GNN-based Recommendation Explainers. The rise of GNN-based recommender systems has created a pressing need for explainability in these recommendation systems [17]. Explainable recommender systems aim to not only deliver accurate predictions but also provide transparent and persuasive justifications for recommendations [17]–[20]. Prior work on explainable recommender systems adopted these strategies [17] of designing intrinsically explainable models with interpretable logic [21], [22] and using post hoc models that generate explanations for the predictions of black-box models [23]. However, these methods face two key challenges: *(1)* representing explainable information often requires node attributes and influential subgraphs identification, and *(2)* reasoning for recommendations relies on domain knowledge [17].

Several explainable AI (XAI) approaches have been proposed, focusing on node or graph classification tasks [2], [24]–[26]. These methods commonly provide factual explanations in the form of

subgraphs deemed relevant for a particular prediction analogous to feature-based XAI methods like LIME [27] and SHAP [28]. [4] was the first work to do a path-based graph neural network explanation for heterogeneous link prediction tasks with the primary limitation that the explanation depends on the ego-graph constructed around the explanation node. Therefore, the explanation technique cannot provide relevant explanations for larger graphs with multiple node attributes (*e.g.,* ego-graph size increases exponentially) and large graph diameter. Our approach bridges the gap between general XAI solutions and the unique need for explainable recommendations, addressing transparency challenges and the complexity of graph-structured user-item relationships.

3 PROBLEM STATEMENT

In this section, we formally define the problem of providing interpretable explanations for the link prediction task in GNN-based recommendation systems using whole heterogeneous graphs. The key challenge is to provide human-interpretable insights that align with the ground-truth so that user confidence and user trust is confirmed. Therefore, we aim to identify the most important feature subset and critical subgraphs used for recommendation by a GNN-based recommendation engine using a real-world heterogeneous real-estate dataset (*e.g.,* Zillow Group, Inc).

Recommendation Task. The *primary task* of the recommendation system discussed here is to predict the likelihood of a link between a user $u \in \mathcal{V}_u$ and a city $l \in \mathcal{V}_c$ based on observed interactions and the graph structure. This is formalized as a link prediction problem: $\hat{y}_{uc} = f_\theta(u, c, \mathcal{G})$, where f_θ is the link prediction model parameterized by θ, and $\hat{y}_{uc}$ is the predicted probability of an interaction between user u and city c.

4 PRELIMINARIES

We outline the construction of interaction graphs used for recommendation and define the ground-truth data leveraged for training and evaluating our models.

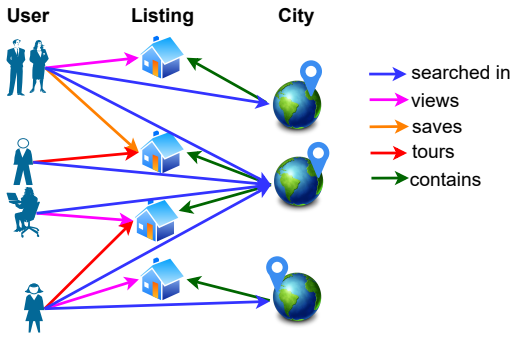

Figure 1: Interaction graph from a real-estate dataset.

4.1 Interaction Graph

The user interaction data in real-estate is inherently relational, comprising of entities such as users, listings, and cities, along with various interaction types like `views`, `saves`, and `tours` as shown in Figure 1 and listed in Table 1 (more details in §A.3). We focus primarily on three entities in this study since location is the most important factor in the home-buying process [29]. All searches on Zillow Group, Inc website [30] correspond to a location, and we consider the city to be one of the entities, in addition to the interactions between users and listings. Since every listing belongs to a city, a special `contains` edge is added between the listing and the city. These entities and interactions are best modeled as a heterogeneous interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where:

- $\mathcal{V}$ represents the set of nodes (*e.g.,* user, listing, and city)
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges that encode interactions (*e.g.,* user $\rightarrow$ `views` $\rightarrow$ listing, city $\rightarrow$ `contains` $\rightarrow$ listing).

The construction of the interaction graph $\mathcal{G}$ involves aggregating data from multiple sources, such as user behavior logs, listing metadata, and geographical information. Each edge $e \in \mathcal{E}$ is associated with a type $\tau(e)$, representing the nature of the relationship between two nodes. Formally, we define a heterogeneous interaction graph as: $\mathcal{G} = (\mathcal{V}_u \cup \mathcal{V}_l \cup \mathcal{V}_c, \mathcal{E}_{ul} \cup \mathcal{E}_{cl} \cup \mathcal{E}_{uc})$, where $\mathcal{V}_u$, $\mathcal{V}_l$, and $\mathcal{V}_c$ represent the sets of user, listing, and city nodes, respectively. $\mathcal{E}_{ul}$ corresponds to interactions between users and listings, $\mathcal{E}_{cl}$ captures relationships between cities and listings, and $\mathcal{E}_{uc}$ captures relationships between users and cities.

Src Node Type	Dst Node Type	Edge Type
User	Listing	`views`, `saves`, and `tours`
User	City	`searched in`
City	Listing	`contains`

Table 1: Description of relationships in the heterogeneous graph.

4.2 Ground Truth

The ground truth for training and evaluating is derived from historical user interactions with listings on the platform. Specifically, the interaction logs provide labels for whether a user u has engaged with a listing l (*e.g.,* viewed, saved, and toured), resulting in positive examples for link prediction. We infer the user to city interaction based on the listing interaction as listings are all associated with a city.

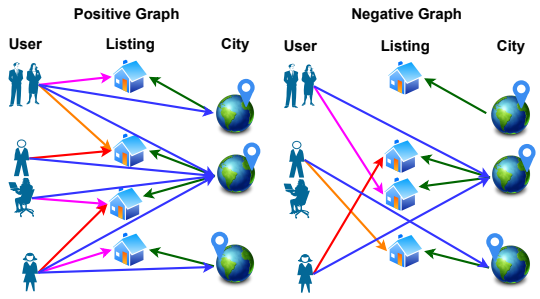

Figure 2: Negative graph construction from positive graph.

Negative examples are defined by selecting edges between users and cities (or listing) where no interaction has been recorded, assuming that non-interaction implies a lack of interest. These are weak negative edges, as there is no explicit evidence that the user dislikes the item. However, for our use case in real-estate, we argue that a user's lack of interaction and interest are closely aligned, given the context where the items are either listings or cities. Thus, incorporating weak negative edges to create a negative graph helps the recommendation model learn finer distinctions about the city (or listing) preferences and understand why users do not interact with them. A negative graph is generated by sampling negative edges from user-listing and user-city pairs as shown in Figure 2.

We generate ground truth explanations by identifying the subgraphs most relevant for each prediction and comparing their node features to identify their similarity. These subgraphs serve as interpretable justifications for why a specific user u is likely to engage with listing l, thus providing a transparent recommendation system.

Let, $\mathcal{Y} \subseteq \mathcal{V}_u \times \mathcal{V}_l$ be the set of observed interactions (positive examples), and $\mathcal{Y}' \subseteq \mathcal{V}_u \times \mathcal{V}_l$ be set of sampled negative examples. The training set $\mathcal{T}$ is constructed as $\mathcal{T} = \{(u, l, y_{ul}) \mid (u, l) \in \mathcal{Y} \cup \mathcal{Y}'$, where $y_{ul} = 1$ if $(u, l) \in \mathcal{Y}$ and $y_{ul} = 0$ if $(u, l) \in \mathcal{Y}'$.

5 Z-REX OVERVIEW

The workflow of Z-REx is illustrated in Figure 3 and algorithm is described in Algorithm 1 where Z-REx first performs feature perturbation to find the important features and then performs graph structural perturbations to identify the important subgraph structures. We *focus* on user-city due to the problem definition in §3, but this method works for any relationship.

5.1 Feature Perturbation.

To interpret the GNN-based recommendation system, we first analyze the influence of target node features using feature perturbation. Consider a user u and their recommended city c_t with their embeddings, h_u and h_{c_t}, where cosine similarity represents the predicted affinity score that the GNN uses to rank city c_t for user u:

$$\text{sim}(\mathbf{h}_u, \mathbf{h}_{c_t}) = \frac{\mathbf{h}_u \cdot \mathbf{h}_{c_t}}{\|\mathbf{h}_u\| \|\mathbf{h}_{c_t}\|}. \tag{1}$$

The feature perturbation involves the following steps:

(1) **Feature Perturbation:** Perturb the target city's features $\mathbf{x}_{c_t}$ one by one by zeroing them out, and the indices of the features to be zeroed out are in $\mathcal{F}$. The perturbed features $\bar{\mathbf{x}}_{c_t}$ are defined as:

$$\bar{\mathbf{x}}_{c_t}[i] = \begin{cases} 0, & \text{if } i \in \mathcal{F}, \\ \mathbf{x}_{c_t}[i], & \text{otherwise.} \end{cases} \tag{2}$$

(2) **Compare Performances:** Compare the change in nDCG@K $\Delta\text{nDCG}(\mathcal{F})$ due to the perturbed features $\bar{\mathbf{x}}_{c_t}$:

$$\Delta\text{nDCG@K}(\mathcal{F}) = \text{nDCG@K}(\bar{\mathbf{x}}_{c_t}) \; - \; \text{nDCG@K}(\mathbf{x}_{c_t}). \tag{3}$$

The ZIGNN is re-evaluated using the perturbed features, and the resulting performance degradation is measured using the ranking metric of nDCG@K (Normalized Discounted Cumulative Gain). A significant drop in the metric indicates the importance of the perturbed feature for the recommendation.

5.2 Structural Perturbation.

While feature perturbation focuses on node-level characteristics, structural perturbation analyzes the impact of graph topology on the model's predictions. Thus, accounting for the whole graph context in each recommendation explanation. Please note that structural perturbation happens with only the subset of features identified in the previous step. Feature perturbation identifies node attributes that have a significant influence on the recommendation outcomes, while structural perturbation uncovers graph edges and relationships that are critical to the model's predictions.

Specifically, we study how the presence or absence of edges in the graph influences the recommendation similarity. The procedure is as follows:

(1) **Graph Transformation**: From the heterogeneous graph, a user-city graph $\mathcal{G}_h$ is created by collapsing all user-city relationships and removing intermediate nodes (e.g., listings). A k-hop subgraph $\mathcal{G}_u^k$ centered around a target user u is then extracted, focusing on both direct and indirect relationships with city nodes.

(2) **Identify Co-clicked Cities**: For the subgraph $\mathcal{G}_u^k$, we identify pairs of cities (c_i, c_j) that share a common predecessor user u_p. These pairs are added as new edges, representing co-click relationships, and their contributions to the model predictions are evaluated. By *co-clicked cities*, we mean cities in which the current user clicked listings, as well as cities other users clicked listings in, where they share at least one common city with the current user. A co-clicked city indicates user groups with similar preferences.

(3) **Edge Removal and Similarity Change**: To assess the importance of structural connections, we iteratively remove identified edges and recompute the similarity between the user embedding $\mathbf{h}_u$ and the target city embedding $\mathbf{h}_{c_t'}$. The change in similarity Δsim after edge removal is defined as:

$$\Delta\text{sim} = \text{sim}(\mathbf{h}_u, \mathbf{h}_{c_t'}) - \text{sim}(\mathbf{h}_u, \mathbf{h}_{c_t}). \tag{4}$$

Edges with the highest absolute Δsim values are identified as critical contributors to the recommendation. These edges represent strong graph relationships that drive user preferences for specific cities. The hyperparameters that influence the structural perturbations are: *(1)* k (hop distance) as it limits the number of edges perturbed to efficiently evaluate while focusing on the most impactful connections and increasing k captures more indirect relationships but may introduce noise, and *(2)* edge removal strategy as prioritizing edges based on shared predecessors ensures that only influential connections are analyzed.

5.3 Graph Size vs. Resource Overhead

We provide real estate-specific recommendation explanations by leveraging the insight that co-clicked cities have a significant impact on recommendations. Co-clicked cities restrict the search space for structural perturbations, which in turn lead to finding the optimal number of edges required to change the recommendations. We acknowledge that iterative perturbation can be a resource-intensive process. But, our experiments show that real estate interaction is sparse in nature (as seen in Table 7), since people view the most number of listings, but only save a few, and tour even fewer listings.

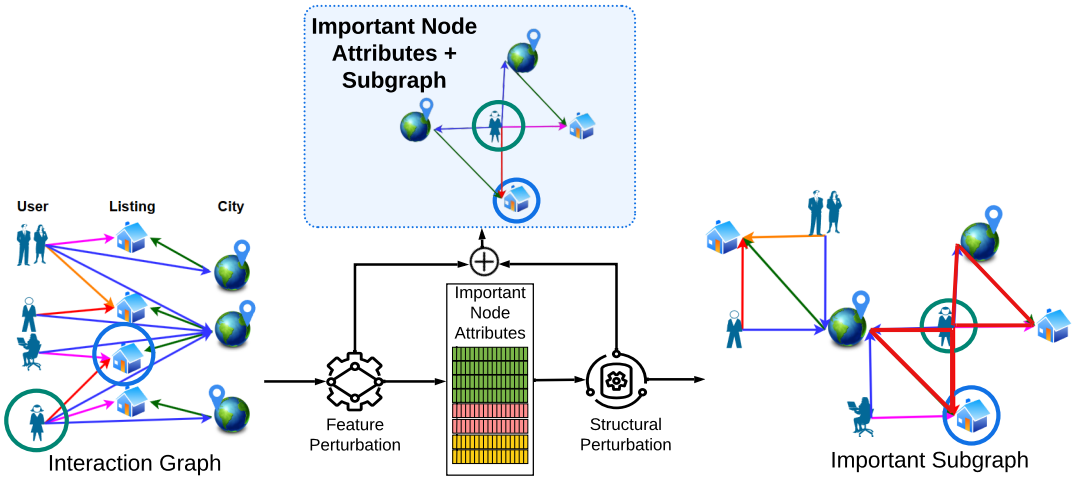

Figure 3: Z-REx overview.

Therefore, the interaction graph from the real-world data is also sparse (as seen in the 30-days sub-section of Table 7), so large-scale real-world deployments are feasible.

As seen in Table 8 in evalaution, training one epoch on a 30-day interaction graph containing 29.8M edges requires 10.89 sec, only an order of magnitude slower than the 0.88 sec required for a 3-day snapshot containing 1.2M edges, confirming that additional temporal coverage does not explode computation and inference time stays almost the same. The results demonstrate that Z-REx delivers timely and faithful explanations at a production scale.

5.4 Generalizability of Z-REx

To adapt the Z-REx for other recommendation tasks, the edge perturbation step needs modification since for real estate, we are using domain knowledge and are focusing on co-clicked edges. For other domains, these edges might not exist, or the assumption we made might not be accurate. Since the structure influences the GNN's message passing step, we think that for other domains, we can translate the insights to structural constraints, we can update §5.2 Structural Perturbation, and decrease the search space from all the potential edges to a smaller list of potentially important edges.

6 EVALUATION

To comprehensively evaluate, Z-REx we investigate the following research questions using the methodology §6.1 and dataset §6.2.

- **RQ1: Recommendation Accuracy.** Can ZIGNN recommend relevant regions? (§6.3)?
- **RQ2: Explanation Accuracy.** Can Z-REx explain ZIGNN's recommendations ? (§6.4)?
- **RQ3: Comparison with SOTA GNN Explainers.** How do the explanations of ZIGNN compare against those of SOTA GNN explainers (GNNExplainer [1], PGExplainer [2], and SubgraphX [3]) (§6.5)?

6.1 Methodology

The data processing of ZIGNN is described in Algorithm 2, which begins by extracting features containing user activities (*e.g.,* `views` and `saves`), user and listing attributes, and their geographical regions. During preprocessing, missing values are handled systematically to ensure data integrity. Outliers, which can distort analysis and model training, are addressed by replacing extreme values (detected using Z-scores) with the column mean. These steps ensure a clean and reliable dataset. Additionally, numerical features are normalized using z-score normalization, creating a consistent data representation. These preprocessing steps create a robust and standardized dataset, ready for graph-based modeling in the recommendation system.

After preprocessing, the user, listing, and city data are used to construct a heterogeneous graph. Each entity (*e.g.,* user, listing, and city) is represented as a unique node type, and edges capture interactions between these nodes. For instance, user-to-listing edges represent activities such as `views` or `saves`, while city-to-listing edges denote listings contained within a city. Node features are assigned and this comprehensive graph representation forms the backbone of the recommendation system, enabling the ZIGNN to understand complex relationships.

The training pipeline first constructs the negative graph to ensure a balanced training dataset by mimicking real-world scenarios where users do not interact with all items. During evaluation, the ZIGNN's effectiveness was tested using node embeddings derived from the trained model. These embeddings were normalized to enable cosine similarity-based retrieval, crucial for recommendation tasks. For specific canonical edge types (*e.g.,* 'user', `views`, 'city'), embeddings were computed separately for source (*e.g.,* user) and destination nodes (*e.g.,* city). The recommendation performance of ZIGNN is measured using nDCG@K, which evaluates the ranking quality of recommendations and prioritizes highly relevant items. Metrics like nDCG@K was used to evaluate recommendation quality, providing quantitative insights.

Metric. Z-REx explains the recommendations generated by ZIGNN, and for this experimentation, we *focus* on user-to-city `views` relationships, but this works for any relationships. First, we identify an *important* subset of features by measuring the *change in nDCG@K score* when individual features are zeroed out. Then, selecting those that produce a negative impact on nDCG, meaning those features were important for predicting relevant recommendations. To align with the recommendation explainability task, we redefine the traditional Fidelity metric [3]—originally based on *change in prediction confidence*—as the *change in nDCG@K score* due to perturbations. The explanation performance of Z-REx is then evaluated based on the drop in nDCG@K score and the change in cosine similarity (Δsim) between user and city embeddings caused by structural perturbations and by only using the *important* features.

Baseline. We compare Z-REx against GNNExplainer [1], SubgraphX [3] and PaGE-Link [4], demonstrating superior explanation quality as evidenced by higher fidelity to the ground truth. We also compare ZIGNN against two baselines to validate its effectiveness: *(1)* random recommendations where recommendations are generated randomly without considering user preferences, and *(2)* histogram recommendations where a histogram-based method that creates recommendations from user-item interaction summary. ZIGNN outperforms both the baselines in recommendation accuracy as demonstrated by higher nDCG@K.

6.2 Dataset

The dataset was collected over a three-day and 30-day period from the state of Washington in the USA. The dataset statistics are shown in Table 5 and Table 6, and more details in §A.4. The dataset is split into training, validation, and test sets, ensuring that the test set contains previously unseen pairs, which are evaluated on the following day to avoid any data leakage. The 30-day setting naturally reduces temporal leakage between training and evaluation interactions, further strengthening the case for Z-REx as a robust, scalable, and domain-aware explainer for real-estate recommender systems. Its ability to combine high fidelity with diversified recommendations makes it the only method that meets practical requirements for accuracy, interpretability, and user engagement.

To note, currently there are no public real estate datasets that contain user-to-listing interactions or the rich attributes we are using. Therefore, we collected and used our own dataset, that accounts for seasonal trends. In the 3-days training data (refer to Table 7), expected trends are seen such as view events are the most popular event, followed by save and then toured because users tend to view multiple items before saving them and finally touring the item. The average interaction of the user to listing for view event for the 25th quantile is 1.0 and the 75th quantile is 8.0, the save event for the 25th quantile is 2.7 and the 75th quantile is 3.0, and the tour event for the 25th quantile is 3.2 and the 75th quantile is 4.0. Therefore, from the average interaction of different events statistics tells us that the graph generated from the dataset will contain more sparse connections than dense connections.

The experiments utilize a heterogeneous graph built from user, listing, and city interaction data (shown in Table 1), where node types encompass multiple attributes of varying data types (shown in Table 4); interested readers can find more details in §A.3. Certain feature values can have valid but outlier values, such as commercial properties with over a thousand bedrooms and bathrooms. In contrast, residential real estate has an average number of bedrooms of 4 and bathrooms of 3.

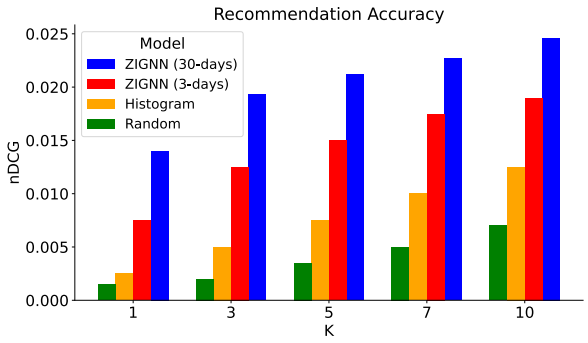

Figure 4: Performance of ZIGNN against different baselines.

6.3 Recommendation Accuracy of ZIGNN

To investigate **RQ1**, we compared the performance between the ZIGNN (3-days) and the baseline recommendation models to reveal ZIGNN consistently and substantially outperforms the rest as seen in Figure 4. ZIGNN consistently surpasses the histogram model across all values of K, with the performance gap widening as K increases. Notably, at $K = 10$, the ZIGNN achieves an nDCG score of 0.019, a 52% improvement over the histogram model's 0.0125, demonstrating ZIGNN's ability to produce more accurate and relevant recommendations as the recommendation set grows.

This trend is further emphasized by the steep growth curve of the GNN's performance compared to the relatively linear increase observed for the histogram model, showcasing the ZIGNN's adaptability. ZIGNN's structural representation learning captures nuanced relationships in the data that the histogram model, with its simpler statistical approach, fails to address. At $K = 1$, where only the top recommendation matters, ZIGNN outperforms the histogram model by a remarkable 200% (0.0075 vs. 0.0025), highlighting its capacity to prioritize the most relevant results effectively. Comparing ZIGNN performance using the extended 30-day dataset and 3-day dataset, the 30-day model yields consistent improvement gains ranging from 30% at $K = 10$ to nearly 90% at $K = 1$, highlighting the benefit of incorporating a longer 30-day interaction window where the user behavior is captured with higher relevance.

6.4 Explanation Quality of Z-REx

For investigating **RQ2**, we first consider the feature perturbation quality and show that we can identify the top ten important city features for the recommendation by measuring the difference between the original nDCG and the perturbed nDCG when a feature is zeroed out. When the difference is positive, the feature was important for predicting the recommendation, as shown in Figure 5, and features with negative differences mean they were confusing the model (more details in §A.5).

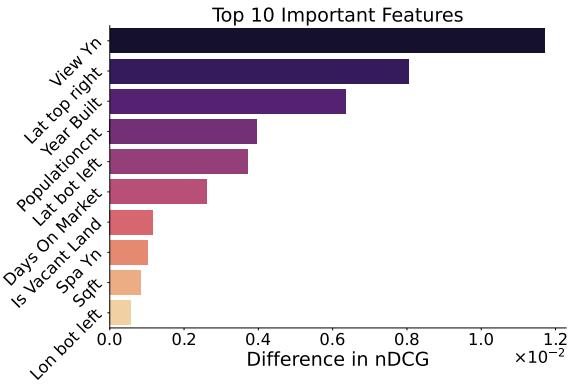

Figure 5: Impact of zeroing out features to find important features.

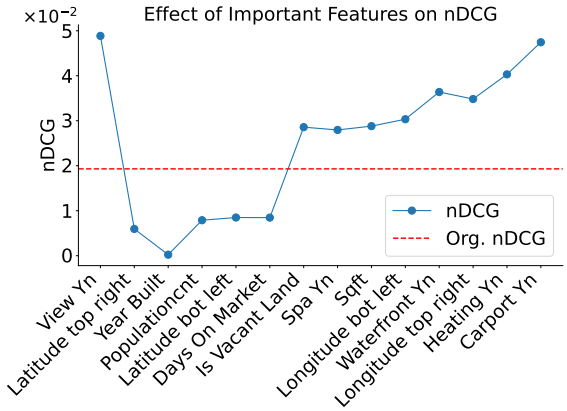

Figure 6: Impact of sequentially adding important features on nDCG.

In particular, these important features can be grouped: *(1)* geographic - latitude (top right and bottom left) and longitude (bottom left), *(2)* boolean - view, spa, and vacant land, and *(3)* numeric - year, population count, days on the market, and square feet. As illustrated in Figure 6, using only the important features yield better performance than using all the features, demonstrating that Z-REx effectively identifies the important features needed for user-city recommendation. These top features will be the only features used for measuring the structural perturbation quality.

The important city features align with domain knowledge: a city is typically characterized by its geographic location and aggregate properties (*e.g.*, population count, the average year of houses, and average days on the market), which are different from the features usually used to characterize a listing (*e.g.*, the number of bathrooms and bedrooms). Aggregate properties offer a holistic snapshot of a city's dynamics and freshness. Interestingly, cities with a younger infrastructure experience rapid growth, indicating a strong interplay between user preference and given recommendations.

In Figure 5, the boolean feature `view` yields the best recommendation performance. Moreover, combining the top fourteen features results in performance similar to the single boolean feature. Incorporating additional features is necessary because relying solely on one boolean feature would compromise the model's generalizability and robustness. Furthermore, considering a diverse set of features, Z-REx does not overfit its explanation to a single scenario.

Change in	**Z-REx**	PaGE-Link [4]	GNNExplainer [1]	SubgraphX [3]
3-days (Training)				
nDCG (% decr.)	**94%**	81% (*-13%*)	21% (*-73%*)	47% (*-47%*)
cosine similarity	**-0.10**	-0.07 (*-0.03*)	-0.02 (*-0.08*)	-0.04 (*-0.06*)
30-days (Training)				
nDCG (% decr.)	**92%**	63% (*-29%*)	9% (*-85%*)	22% (*-70%*)
cosine similarity	**-0.09**	-0.05 (*-0.04*)	-0.01 (*-0.08*)	-0.02 (*-0.07*)

Table 2: Quantitative evaluation of Z-REx against GNN explainers.

Table 2 quantitatively evaluates the performance of Z-REx against SOTA GNN explainers, where Z-REx outperforms PaGE-Link, GNNExplainer, and SubgraphX in both nDCG and cosine similarity metrics, demonstrating its superior explanation accuracy compared with the ground-truth. In the ground-truth, the edges in the real-world dataset are treated as positive, and other edges are treated as negative. It is seen that Z-REx achieves the most significant reduction in nDCG (94%), indicating superior fidelity in explanation degradation. Additionally, it achieves the largest decrease in cosine similarity (-0.10), highlighting its effectiveness in finding the important subgraphs that alter ZIGNN's recommendations.

Our month-long (30-days) evaluation confirms that Z-REx delivers markedly stronger explanations than all baselines while optimally scaling to a larger interaction graph. With a 30-days training window, Z-REx preserves 92% of the original nDCG, almost identical to its 3-day performance (94%), and maintains the largest embedding shift (-0.09), indicating that it highlights influential edges without collapsing recommendation diversity.

The stability of Z-REx across time windows suggests that its structural perturbation strategy generalizes rather than over-fitting to short-term interaction noise. The two-point drop in nDCG retention and the 0.01 improvement in cosine similarity from 3 to 30 days imply that extending the past horizon yields the same explanatory fidelity with slightly more conservative embedding movements, an attractive trade-off for production systems.

6.5 SOTA GNN Explainers vs. Z-REx

We compared Z-REx against a specialized GNN-based recommendation explainer PaGE-Link [4] and two general GNN explainers, GNNExplainer [1] and SubgraphX [3] in Table 2 to answer **RQ3**. Z-REx performed the best, followed by PaGE-Link and general purpose GNN explainers performed the worst which is expected since Z-REx and PaGE-Link have been specifically designed for LP tasks. In contrast, the recommendation-specific PaGE-Link retains only 63% of nDCG and induces a much smaller change in cosine similarity, while general-purpose explainers fall to 22% (SubgraphX) and 7% (GNNExplainer). Competing methods degrade sharply because their mask-optimization (GNNExplainer) or Monte-Carlo subgraph search (SubgraphX) faces an exponentially expanding candidate space as graph size grows.

Since, PaGE-Link creates the k-hop ego-graph by limiting the consideration of relevant features and subgraphs. Therefore, PaGE-Link performance degrades sharply because they compute the k-hop neighborhood around the target link, but there is no guarantee that the k-hop relationship will include relevant links. This limitation is reflected by -0.04 less decrease is cosine similarity. Z-REx avoids this pitfall by exploiting domain knowledge: the number of co-clicked cities grows sub-linearly, so the search space for meaningful perturbations remains tractable even on month-long graphs. Compared to general purpose GNN explainer, *i.e.,* GNNExplainer and SubgraphX, Z-REx demonstrates average of -78% more reduction in nDCG and -0.08 more cosine similarity increase, indicating that it more effectively identifies the features and subgraphs the GNN relies on to make recommendations.

Hyperparameter	Values Tested	nDCG@1 Scores	Best Score
Neighborhood Size (k)	4, 8, 16	0.0070, 0.0075, 0.0071	0.0075 (at k=8)
Edge Removal Strategy	PRI, HID, HBC	0.0070, 0.0077, 0.0075	0.0077 (at HID)
Negative Graph Size	1, 3, 5	0.0072, 0.0074, 0.0075	0.0075 (at 5)
Learning Rate	10^{-3}, 10^{-2}, 10^{-1}	0.0072, 0.0074, 0.0069	0.0074 (at 10^{-2})
Output Feature Dim.	128, 256, 512	0.0071, 0.0075, 0.0081	0.0081 (at 512)

Table 3: Hyperparameter sensitivity summary. Each row varies one hyperparameter while holding the others constant; the rightmost column reports the highest nDCG@1 observed for that setting.

6.6 Hyperparameter Sensitivity Study

Table 3 shows the hyperparameter sensitivity tuning summary where we test different neighborhood sizes (K), edge removal strategy, size of negative graph, learning rate and output feature dimensions. The size of the input feature dimension is governed by the number of features for each entity type, so we did not incorporate that. The three different edge removal strategies we used are: *(1)* PRI (prioritize Recent Interaction) which removes edges with more recent interactions first (reduces the strength of fresh user-item relationships); *(2)* HID (High In-Degree) which removes edges to nodes with the highest in-degree (breaking these highly connected links disrupts potential shared neighborhoods); and *(3)* HBC (High Betweenness Centrality) which removes edges with high betweenness centrality, targeting the links acting as bridges in the hub cities in the interaction graphs.

Increasing the neighborhood size (K) from 4 to 8 improves nDCG@1, but going further to 16 diminishes it, indicating that while broader neighborhoods capture more context, too large a neighborhood may dilute relevant signals. HID (high in-degree) as the edge removal strategy yields the best score among the tested strategies, suggesting that selectively removing overly connected edges can help avoid excessive overlap and improve the model's focus on more distinct interactions. A larger negative graph size, up to a point, seems beneficial (optimal at size 5 in this table) because it gives the model a more balanced signal to distinguish relevant versus irrelevant connections. Tuning the learning rate reveals that a rate of 1e-2 provides a stable convergence path. Finally, a higher output feature dimension (512) significantly boosts nDCG@1, demonstrating that a richer embedding space helps the model capture more nuanced relational patterns but at the cost of resource consumption.

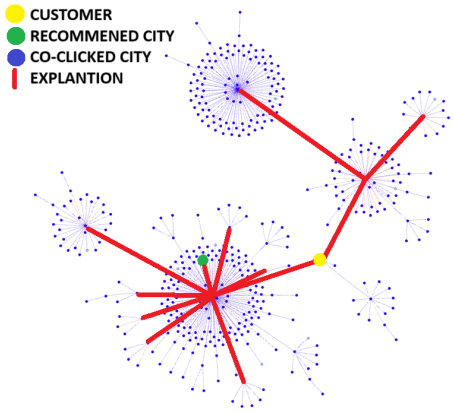

Figure 7: Z-REx's explanation of a recommended city #1.

7 CASE STUDY

In the case study, we selected a real user (*i.e.,* customer) and, based on their interactions, generated a city recommendation along with explanations. To enhance visualization, we present a simplified graph (Figure 7) focusing on the customer (yellow node), co-clicked cities (blue nodes), and the recommended city (green node). A blue edge will exist between the co-clicked city nodes, the recommended city is the green node, and the red edges are the edges identified as important for recommendations.

Figure 7 illustrates that the user has direct connections to a limited number of co-clicked cities. However, one of these cities is highly connected to other cities, including the recommended city. These highly connected cities act as information hubs, facilitating the flow of information between the user and the recommended city. This information flow occurs through two mechanisms: *(1)* shared preferences, users who co-click on listings in the hub city exhibit similar preferences to the target user, making the recommended city more relevant, and *(2)* indirect connections, the hub city connects the user to a broader network of cities with relevant listings, increasing the likelihood of discovering the recommended city. Consequently, Z-REx identifies the edges connecting the user to these hub cities as critical to the recommendation, as their removal significantly impacts the information flow and potentially disrupts the discovery of relevant recommendations.

8 CONCLUSION

We introduced Z-REx, which aims to bridge a critical gap in explainable ML for recommendation systems by providing human-interpretable insights into GNN-based recommendation systems to increase user confidence and customer engagement. Z-REx offers contextually relevant and human-interpretable explanations through structural and attribute perturbation. Z-REx outperforms PaGE-Link by 29%, GNNExplainer by 85%, and SubgraphX by 70% in the Fidelity metric, demonstrating its superior explanation accuracy compared with the ground-truth. Specifically, ZIGNN provides whole graph instance-level explanations for GNN models in the context of the heterogeneous link prediction (LP) task by identifying a subset of node features and subgraphs that are most influential for GNN's-based recommendation system.

REFERENCES

[1] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Neural Information Processing Systems*, 2019.

[2] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *Neural Information Processing Systems*, 2020.

[3] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International Conference on Machine Learning*, PMLR, 2021.

[4] S. Zhang, J. Zhang, X. Song, S. Adeshina, D. Zheng, C. Faloutsos, and Y. Sun, "Page-link: Path-based graph neural network explanation for heterogeneous link prediction," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 3784–3793.

[5] Y. Wang, Y. Zhao, Y. Zhang, and T. Derr, "Collaboration-aware graph convolutional network for recommender systems," in *Proceedings of the ACM web conference*, 2023.

[6] K. Mukherjee, J. Wiedemeier, T. Wang, M. Kim, F. Chen, M. Kantarcioglu, and K. Jee, "Explaining provenance-based gnn detectors with graph structural features," *arXiv preprint arXiv:2306.00934*, 2023.

[7] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 27–34.

[8] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 1289–1298.

[9] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[10] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph convolution collaborative filtering," in *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2019, pp. 1306–1311.

[11] Z. Liu, L. Meng, F. Jiang, J. Zhang, and P. S. Yu, "Deoscillated graph collaborative filtering," *arXiv preprint 2011.02100*, 2020.

[12] C. Li, K. Jia, D. Shen, C.-J. R. Shi, and H. Yang, "Hierarchical representation learning for bipartite graphs.," in *IJCAI*, vol. 19, 2019, pp. 2873–2879.

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[14] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[16] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[17] Y. Zhang, X. Chen, *et al.*, "Explainable recommendation: A survey and new perspectives," *Foundations and Trends® in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020.

[18] Z. Lyu, Y. Wu, J. Lai, M. Yang, C. Li, and W. Zhou, "Knowledge enhanced graph neural networks for explainable recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4954–4968, 2022.

[19] R. Sinha and K. Swearingen, "The role of transparency in recommender systems," in *CHI'02 extended abstracts on Human factors in computing systems*, 2002, pp. 830–831.

[20] X. Wang, K. Liu, D. Wang, L. Wu, Y. Fu, and X. Xie, "Multi-level recommendation reasoning over knowledge graphs with reinforcement learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2098–2108.

[21] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 83–92.

[22] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015.

[23] G. Peake and J. Wang, "Explanation mining: Post hoc interpretability of latent factor models for recommendation systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2060–2069.

[24] A. Duval and F. D. Malliaros, "Graphsvx: Shapley value explanations for graph neural networks," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, Springer, 2021, pp. 302–318.

[25] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 6666–6679.

[26] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[27] M. T. Ribeiro, S. Singh, and C. Guestrin, """ why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[28] S. Lundberg, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[29] Z. Harrison and A. Khazane, "Taxonomic recommendations of real estate properties with textual attribute information," in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 479–481.

[30] *Zillow group, inc website*, https:www.zillow.com.

A APPENDIX

A.1 ZIGNN Architecture

ZIGNN is a Graph Neural Network-based recommendation engine designed to operate on heterogeneous graphs $\mathcal{G}$ built using real-world real-estate (*e.g.*, Zillow Group, Inc.) interaction data. ZIGNN not only models user-listing (or city) interactions for recommendation but also captures contextual information such as city-level influences and item similarities. The ability to leverage heterogeneous relationships improves the recommendation quality by allowing the system to reason about multi-hop dependencies and indirectly related entities.

ZIGNN utilized the Zillow Group, Inc. dataset for modeling and evaluation purposes, demonstrating its effectiveness in a real-world recommendation system. However, the architecture of ZIGNN is generic and can be adapted to various platforms across different domains. ZIGNN can be extended to use cases where the context is critical in influencing user decisions, such as (user, item, context/category), where the user and item types will dictate the node types, and the context interaction between the user and item will for the edges. For instance, it can model different interactions: (user, food, restaurant) in the food industry, (user, post, topic) in social media, (patient, doctor, hospital) in healthcare, or (user, job, company) in the job search industry. By leveraging the interaction graph creation and behavioral learning flexibility, ZIGNN can cater to a wide range of recommendation scenarios, enhancing user satisfaction and specific context-aware decision-making across multiple domains.

The ZIGNN consists of the following components:

(1) Node Embedding Layer: Each node $v \in \mathcal{V}$ is mapped to a dense vector representation $\mathbf{h}_v \in \mathbb{R}^d$ using an embedding layer. The initial embeddings are learned from the node features and are iteratively updated during the training process.
(2) Message Passing Mechanism: ZIGNN employs a multi-hop message passing mechanism, where each node $v \in \mathcal{V}$ aggregates information from its neighbors $\mathcal{N}(v)$ through a learnable function. The node update rule for t-th layer is defined as $\mathbf{h}_v^{(t+1)} = \text{AGG}\left(\mathbf{h}_v^{(t)}, \left\{\mathbf{h}_u^{(t)} : u \in \mathcal{N}(v)\right\}\right)$, where AGG is an aggregation function such as sum, mean, or attention-based pooling. This allows the model to capture higher-order dependencies between nodes.
(3) Link Prediction: The final node embeddings $\mathbf{h}_u$ and $\mathbf{h}_l$ for users and listings are fed into a scoring function to compute the likelihood of a link using $\hat{y}_{ul} = \sigma(\mathbf{h}_u^T W \mathbf{h}_l)$, where σ is the sigmoid function and W is a learnable weight matrix.

ZIGNN is composed of three key components: a projection layer for aligning feature dimensions across different node types, a Relational Graph Convolutional Network (RGCN) for learning node embeddings while considering multi-relational graph structures, and a dot product predictor layer for scoring edge relationships. The projection layer standardizes feature dimensions by applying type-specific linear transformations. This ensures compatibility with the RGCN, which processes node and edge type information through multiple layers of convolution, enabling the model to capture complex interactions between heterogeneous entities.

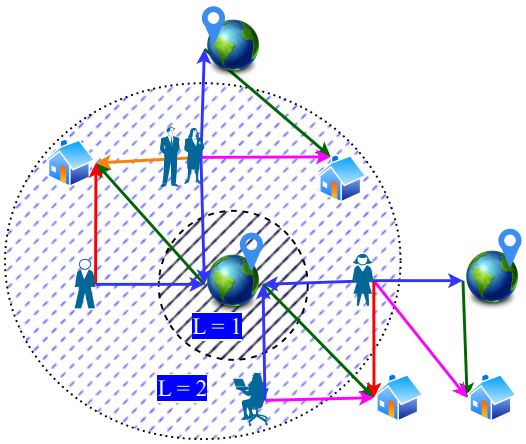

Figure 8: Interaction graph diameter is two so we need at most two GCN layers to incorporate information from the farthest nodes.

The RGCN implementation employs a two-layer design, where each layer performs graph convolutions over different relationship types. We purposefully selected two-layer RGCN because the graph diameter of $\mathcal{G}$ is at most two as shown in Figure 8. Since, the message passing step happens in parallel, the numbers of times the message passing steps need to be completed for the information to travel from the farthest part of the graph is two.

For each relationship, the model constructs a separate Graph Convolutional layer and applies type-specific linear transformations to incorporate edge-specific features into the convolution process. Additionally, self-loop embeddings are refined using residual connections for each node type, ensuring that the node's initial features are preserved alongside learned representations. This structure allows the RGCN to aggregate information across the graph, dynamically updating node embeddings while addressing the unique characteristics of heterogeneous relationships.

The dot product predictor layer scores edges by computing the dot product between node embeddings at the source and target ends of an edge for each edge type. During this process, the predictor assigns scores to edges by using the learned node features (h) from the RGCN, which encapsulate the structural and relational context of each node in the graph. For positive edges, which represent actual relationships observed in the graph, the scores reflect the strength of these connections as encoded in the node embeddings. Conversely, for negative edges, which are artificially sampled to represent non-existent or unlikely relationships, the scores typically indicate weaker or negligible associations. This distinction between positive and negative edge scores is critical for the model to learn meaningful embeddings and effectively differentiate between real and spurious relationships, forming the basis for accurate recommendation tasks.

The implementation is built using Deep Graph Library (DGL) for graph operations and PyTorch for neural network components. FAISS is integrated to perform efficient nearest-neighbor searches for recommendation evaluation.

Algorithm 1 Z-REx

Require: target user u, target city c_t, heterogeneous interaction graph G_{hetero}, trained GNN ZiGNN, rank cut-off K, city-feature F_{all}, hop distance k

function Z-REx($u, c_t, F_{\text{all}}, k$)
  $ranked\_features \leftarrow$ FeaturePerturb(u, c_t, F_{all})
  $ranked\_edges \leftarrow$ StructuralPerturb(u, c_t, k)
  return *ranked_features, ranked_edges*
end function
function FeaturePerturb(u, c_t, F_{all})
  $baseline\_ndcg \leftarrow$ NDCG(u) ▷ original score
  $ranked\_features \leftarrow \{\}$
  for all $f \in F_{\text{all}}$ **do**
    $\bar{x}_{c_t} \leftarrow$ ZeroOutFeature(c_t, f)
    $\Delta ndcg \leftarrow \text{NDCG}(u, \bar{x}_{c_t}) - baseline\_ndcg$ ▷ Eq.(3)
    $ranked\_features \leftarrow ranked\_features \cup (f, \Delta ndcg)$
  end for
  Sort(*ranked_features*) by $\Delta ndcg$ (descending)
  return *ranked_features*
end function
function StructurePerturb(u, c_t, k)
  $G_h \leftarrow$ CollapseToUserCityGraph(G_{hetero}, c_t)
  $G_k \leftarrow$ KHopSubgraph(G_h, u, k)
  AddCo-ClickEdges(G_k)
  $base\_sim \leftarrow \cos(\text{ZiGNN.embed}(u), \text{ZiGNN.embed}(c_t))$
  $ranked\_edges \leftarrow \{\}$
  for all edge e incident to u or c_t in G_k **do**
    temporarily remove e
    $new\_sim \leftarrow \cos(\text{ZiGNN.embed}(u), \text{ZiGNN.embed}(c_t))$
    $\Delta sim \leftarrow new\_sim - base\_sim$ ▷ Eq.(4)
    $ranked\_edges \leftarrow ranked\_edges \cup (e, |\Delta sim|)$
    restore e
  end for
  Sort(*ranked_edges*) by $|\Delta sim|$ (descending)
  return *ranked_edges*
end function

During training, both positive and negative edge scores are calculated for each edge type by applying the dot-product predictor to pairs of node embeddings, enabling the model to distinguish between observed and unobserved relationships. We use the Adam optimizer with weighted decay to optimize margin-based loss, where the positive edge scores are encouraged to exceed negative edge scores by at least a margin of 1, penalizing cases where this condition is not met. The modular design supports inference by directly returning node embeddings when negative edges are not provided. This architecture ensures the model can efficiently learn and generalize from multi-relational graph data for edge classification tasks.

A.2 Z-REx Algorithm

Algorithm 1 describes the Z-REx algorithm and Algorithm 2 describes the algorithm to generate the heterogeneous interaction graph, G_{hetero} from clickstream events. Algorithm 2 converting raw interaction and regional metadata CSVs into attribute-normalized

Algorithm 2 Generate Heterogeneous Interaction Graph

Require: CSVs (*i.e., events_file*, *regions_file*), normalization method *norm_method*

function Preprocess($events\_file, regions\_file$)
  $events \leftarrow$ ReadCSV($events\_file$)
  $regions \leftarrow$ ReadCSV($regions\_file$)
  $U_L \leftarrow$ BuildUserListingEdges($events$)
  $regions' \leftarrow regions[id \in U_L.listing\_id]$
  $U_C \leftarrow$ BuildUserCityEdges($U_L, regions'$)
  $C_L \leftarrow$ BuildCityListingEdges($regions'$)
  $users \leftarrow$ Unique($events.user\_id, Xavier\_Init$)
  $listings \leftarrow$ CleanNumeric($regions', norm\_method$)
  $cities \leftarrow$ AggregateCity($regions', norm\_method$)
  $G_{hetero} \leftarrow$ dgl.heterograph(U_L, U_C, C_L)
  return G_{hetero}
end function

heterogeneous graph, G_{hetero}, that connects the users, listings, and cities through user-listing, user-city, and city-listing edges.

Using G_{hetero}, Algorithm 1 delivers fine-grained model transparency: for a given user-city recommendation, it *(1)* ranks city-level features by zeroing them out and measuring the resulting $\Delta nDCG@K$, and *(2)* ranks graph edges in the K-hop neighbourhood by perturbing edge and observing the change in the cosine similarity between the ZiGNN embeddings of the user and target city. The joint output—two ordered lists of high-impact features and structural subgraphs, exposes the concrete evidence that the GNN relied upon.

Node Type	Attribute	Attribute Type
User	session id	Numeric
Listing	bedrooms, bathrooms, year built, sq. ft, price binned sq. ft, binned price, price per bedroom days on market, floors	Numeric
Listing	Waterfront, heating, basement, fireplace Cooling, view, vacant, spa, carport Pool, new construction	Boolean
Listing	Latitude top/bottom left/ right Longitude top/bottom left/right	Geographic
City	population count, avg. all the listing features for listings per city	Numeric
City	avg. all the boolean features for listings per city	Boolean
City	avg. all the geographic features for listings per city	Geographic

Table 4: Attributes and their types for different node types.

A.3 Node and Edge Details

As described in the preliminaries §4.1, there are three different types of nodes: user, listing, and city. There are different relationships or edges between them, as shown in Table 1. The most popular relationship between the user and the listing is the `views` relationship, since users view multiple listings before narrowing down their search by saving the listing and finally touring the listing. Between user and city, `searched in` relationship exists, and between city and listing `contains` relationship exists.

Table 4 outlines the attribute schema for the different node types used in our model. The User node is characterized by a numeric session identifier. The Listing node includes a diverse set of features: numeric attributes (*e.g.,* bedrooms, bathrooms, price, etc.), boolean attributes (*e.g.,* waterfront, heating, etc.), and detailed geographic attributes (latitude and longitude for the property boundaries). Meanwhile, the city node aggregates listing data by averaging numeric, boolean, and geographic attributes and adding population count.

	Date	City	Listing	User
3-days Training	5/17-5/20	449	55k	393k
30-days Training	4/20-5/20	456	97k	1.3M
Testing	5/27-5/30	452	53k	405k
Evaluation	5/31	448	45k	203k

Table 5: Number of entities (*e.g.,* nodes) in different datasets.

	Date	`views`	`saves`	`tours`	`contains`
3-days Training	5/17-5/20	789k	169k	234k	55k
30-days Training	4/20-5/20	26M	2.2M	1.6M	97k
Testing	5/27-5/30	773k	138k	197k	53k
Evaluation	5/31	714k	41k	56k	45k

Table 6: Number of relationships (*e.g.,* edges) in different datasets.

	`views`	`saves`	`tours`
	3-days		
mean	7.97	2.74	3.24
25^{th} quantile	1.00	1.00	1.00
median	3.00	2.00	2.00
75^{th} quantile	8.00	3.00	4.00
	30-days		
mean	21.90	5.65	6.72
25^{th} quantile	2.00	1.00	1.00
median	5.00	2.00	3.00
75^{th} quantile	18.00	6.00	7.00

Table 7: Average number of user-city interactions.

A.4 Dataset Statistics

The dataset statistics as shown in Table 5 and Table 6 shows that the dataset is divided into four segments: 3-day training (May 17-20), month-long training (April 20 - May 20), testing (May 27-30), and evaluation (May 31)—providing a temporal snapshot of various metrics. The 3-day training dataset, spanning May 17th to May 20th, includes 55k listings viewed by 393k users across 449 cities, resulting in 789k `views`, 169k `saves`, and 234k `tours`. The `contains` relationship matches the number of listings since each listing must belong to a city. The testing dataset exhibits similar characteristics from May 27th to May 30th with slightly different values (53k listings and 405k users). Notably, the evaluation dataset on May 31st, while covering a similar number of cities (448), shows a smaller number of listings (45k), users (203k), saves (41k), and tours (56k) in comparison to training and testing datasets since there is a decrease in user engagement during the evaluation period which is just one day after the testing period. The `views` are the largest interactions since users view the listing the most, and only a few of them convert to `saves`, and fewer convert to `tours`.

As illustrated in Table 7, user engagement with listings scales markedly with the observation window. Over the 7-day horizon, users perform a mean of 7.97 views, 2.74 saves, and 3.24 tour requests per listing, yet the corresponding medians (3, 2, 2) reveal a highly right-skewed distribution in which a small subset of users interactions form the majority of the engagement dataset. Extending the window to 30 days amplifies this effect: mean interactions roughly triple for views and more than double for both saves and tours, while the medians increase more modestly (to 5, 2, 3), indicating that additional time primarily benefits heavier-engagement cohorts. The widening gap between the 25^{th} and 75^{th} percentiles across all metrics further shows the importance of the growing dispersion, suggesting that long-term recommendation strategies should explicitly account for this heterogeneity rather than relying on aggregate averages alone.

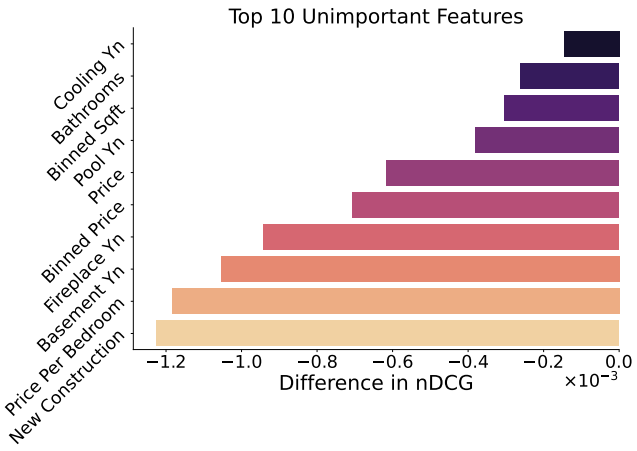

Figure 9: Impact of zeroing out features to find unimportant features.

A.5 Unimportant Features

The unimportant features, as shown in Figure 9, are identified by measuring the performance increase when the features are zeroed out, indicating that the feature was detrimental to the ZiGNN 's predictive performance. Therefore, these are unimportant features and should be removed from the model to enhance accuracy. It is interesting to see that the numeric unimportant features are usually used to identify listing. Understandably, aggregating those listing features is not helpful to ZiGNN when compared against city-specific features, such as population count, average year built, and geographic-based features.

Table 8: ZiGNN Runtime performance on different dataset windows.

	Scenario	# of Edges	Time (s)
3-days			
Training	per epoch	1.2M	0.88
Evaluation	per user	248k	5.14
30-days			
Training	per epoch	29.8M	10.89
Evaluation	per user	248k	5.04

A.6 Runtime Analysis: 3 vs 30 Day Dataset

Table 8 shows that enlarging the interaction window from 3 days (1.2M edges) to 30 days (29.8M edges) raises the cost of one training epoch from 0.88s to only 10.89s. Despite a 25 times increase in graph size, the runtime grows by just one order of magnitude, confirming that our training pipeline scales sub-linearly with temporal coverage. Inference remains virtually size-independent: generating an explanation for a single user takes 5.14s on the 3-day graph versus 5.04s on the 30-day graph. Together, these results demonstrate that ZiGNN delivers timely and accurate explanations at production scale.