ExplorAct: Context-Aware Next Action Recommendations for Interactive Data Exploration

Dinuka Manohara de Zoysa
The University of Melbourne
Melbourne, Australia
bdezoysa@student.unimelb.edu.au

James Bailey
The University of Melbourne
Melbourne, Australia
baileyj@unimelb.edu.au

Renata Borovica-Gajic
The University of Melbourne
Melbourne, Australia
renata.borovica@unimelb.edu.au

Abstract

Modern data analysis platforms, such as Tableau, Microsoft Power BI, Google Looker Studio, Kibana, and Splunk, have democratized data exploration by enabling users to interact with data through intuitive visual interfaces, eliminating the need for proficiency in query languages like SQL. These platforms allow both experts and non-experts to perform high-level operations and incrementally construct complex analysis workflows. As the volume and complexity of data grow, assisting users in navigating these workflows becomes increasingly important. One promising direction is to provide intelligent next-action recommendations that guide users through meaningful and efficient exploration paths.

In this paper, we present **ExplorAct**, a context-aware next-action recommendation framework that leverages historical session logs to predict and suggest relevant next steps during data exploration. Unlike existing approaches that suffer from scalability issues due to log-size-dependent retrieval, **ExplorAct** achieves constant-time inference by employing a deep learning architecture that models both the structural and sequential aspects of exploration sessions. Through extensive experiments on four real-world datasets, we show that **ExplorAct** consistently outperforms state-of-the-art (SOTA) baselines across three core recommendation tasks, while maintaining stable and low-latency inference regardless of log size.

CCS Concepts

• **Information systems → Recommender systems**; **Query suggestion**; *Data analytics*; • **Mathematics of computing → Exploratory data analysis**.

Keywords

Interactive Data Analysis, Action Recommendation, Graph Isomorphism Networks, Gated Recurrent Units, Evidence Fusion

ACM Reference Format:
Dinuka Manohara de Zoysa, James Bailey, and Renata Borovica-Gajic. 2025. ExplorAct: Context-Aware Next Action Recommendations for Interactive Data Exploration. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746252.3761257

1 Introduction

Modern interactive data analysis (IDA) platforms empower users to explore and gain insights from complex datasets. However, as these platforms grow in complexity, users — particularly non-experts —face increasing challenges in navigating their workflows efficiently [2]. To mitigate this, action recommendation systems have emerged as promising tools to guide users by suggesting relevant next steps during exploratory analysis. Practical recommendations not only accelerate the discovery process but also enhance user understanding and decision-making by reducing cognitive and operational overhead.

Consider the case of scientists analyzing datasets from multiple experimental trials to identify anomalies or validate theoretical patterns. Recommendations based on previous exploration sessions, either their own or those of peers, can foster collaboration and knowledge transfer [14, 17, 28]. Such suggestions serve dual roles: encouraging novel exploration while grounding actions in previously successful analytical strategies. The result is a more efficient, informed, and user-friendly analysis experience.

Recent work has formalized the next-action prediction problem in IDA platforms, particularly within multi-dataset environments [28]. Other efforts have targeted related challenges, such as estimating the "interestingness" of query results [36], or generating exploration notebooks that encapsulate analytical intent [5, 25], which often rely on prior knowledge of the analysis objective and are less suited to open-ended exploratory scenarios. Additional lines of research focus on explaining user behaviour via interestingness-based rationales [10].

A common abstraction used in state-of-the-art (SOTA) next-action recommendation systems is the *analysis tree*: a hierarchical representation of a user's exploration session, where nodes denote actions and edges encode transitions or dependencies [28]. From these, a *context tree* is derived to represent the user's current state. This structured representation offers advantages over flat sequences by preserving branching behaviors typical in exploratory workflows. Despite this, existing systems face two key limitations (depicted in Figure 1):

① **Under-utilization of Tree Structures:** Prior work largely applies simple similarity-based retrieval over context trees, primarily using k-nearest neighbors with tree edit distances [28, 36]. Such an approach fails to exploit the rich, hierarchical, and feature-annotated nature of these trees, leaving substantial latent information untapped.

② **Neglect of Sequential Patterns:** While tree structures model the structural context of a session, they often overlook the inherently sequential nature of exploration. The temporal progression of user actions carries important signals for predicting future steps, which are under-exploited in current approaches.

To address these gaps, we propose **ExplorAct**, a deep learning framework for context-aware next-action recommendation in modern IDA platforms. Our key idea is to combine the expressive power of graph-based representations with sequential modeling, capturing both the structural context (via context trees) and temporal dynamics (via action sequences).

Similar to the prior work [28], we focus on three core recommendation tasks:
① **Action-Type Recommendation:** Predicting the next type of analytical action a user is likely to perform (e.g., filter, projection).
② **Column Recommendation:** Suggesting the most relevant column(s) for the upcoming action.
③ **Action-Type and Column Pairing:** Jointly predicting action-type and associated column, a more fine-grained and practical recommendation setting.

This work makes the following key contributions:

- **Problem Formulation:** We formulate next-action recommendation as a candidate probability prediction task. This novel formulation enables us to train on new candidates incrementally without restructuring the model architecture.
- **Novel Deep Learning Architecture:** We introduce a hybrid model combining Graph Isomorphism Networks (GIN) to embed structural context and Gated Recurrent Units (GRU) to capture sequential dependencies, viewing context trees as a sequence of evolving user states. We also provide two variants of input encoding strategies.
- **Evidence Fusion via Dempster-Shafer Theory (DST):** We propose an evidence fusion mechanism to improve recommendation robustness across tasks. Each model, trained for a specific task, acts as an independent evidence source with its own uncertainty. We use DST to systematically combine these sources, yielding more reliable joint action-type and column recommendations.
- **Comprehensive Experimental Validation:** Through extensive experiments on real-world data sets, we demonstrate consistent improvements of up to **16.98%** (**11.5%** avg.), **22.22%** (**17.02%** avg.), and **20.94%** (**13.69%** avg.) of Recall@3 for the three recommendation tasks, over the SOTA respectively. Column recommendation and joint action-type, column recommendation tasks show improvements up to **34.38%** (**22.49%** avg.), and **13.87%** (**11.54%** avg.) of Mean Reciprocal Rank, respectively. Our model also achieves superior generalization and sample efficiency. Notably, the inference cost remains constant with respect to session log size, a key advantage over existing kNN-based methods, which scale poorly due to log-size-dependent retrieval operations. The inference time depends only on the maximum context size of a context tree sequence. We publish our code to assist future research.

The remainder of this paper is organised as follows. Section 2 defines the problem formally. Section 3 presents our proposed framework in detail. Section 4 reports empirical results, followed by a discussion. Section 5 reviews related work and background. We conclude in Section 6.

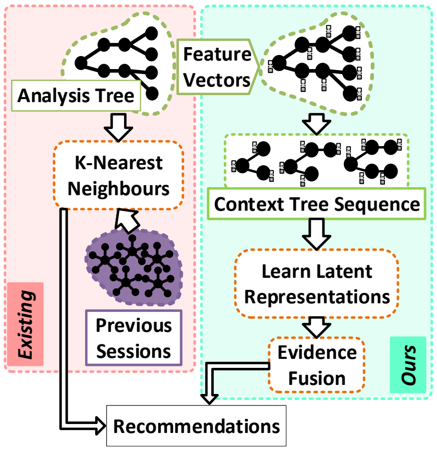


Figure 1: Existing SOTA vs. ExplorAct

Table 1: Summary of notations.

Symbol	Definition
$\mapsto$	Mapping operator
A	The set of attributes (columns)
$a \in A$	An attribute/column a
O_i	The set of rows (records) in the i-th dataset
$D_i = (O_i, A)$	The i-th dataset
C	The set of all possible columns
$D = \{D_i \mid i = 1, 2, \ldots\}$	The set of datasets with matching attributes (same columns, different rows)
$\mathcal{D} = (\bigoplus_{i=1}^{m} O_i, A)$	The concatenated ($\bigoplus$) dataset.
$a(D_i)$	The multiset of values for attribute a in D_i
$a(\mathcal{D})$	The multiset of values for attribute a in $\mathcal{D}$
$\Delta_r = (o_i, c)$ $o_i \subseteq O_i,\ c \subseteq C$	The result of an action α_s performed on dataset D_i; $r \in \mathbb{Z}^+$ indicates step index
$a(\Delta_r)$	The multiset of values for attribute a in Δ_r
$\zeta \in Z_*$	Z_* is the set of candidates for a recommendation task, and ζ is one such candidate

2 Preliminaries and Problem Definition

To motivate and formalize our recommendation approach, we begin by introducing key concepts that capture user interactions during data exploration. We first define the structure of *analysis actions*, which serve as the atomic units of user behavior. Next, we describe how sequences of these actions form *analysis trees*, which represent the evolving state of a user's exploration session. From these trees, we extract *context trees*, substructures that encapsulate the recent history leading to a given action, and use them as the primary input to our learning model. Finally, we formally define the recommendation objective. The notations used throughout the paper are summarized in Table 1.

Throughout the paper, we use a running example in which users explore datasets containing information about age, country, city, employment, and COVID-19 vaccination status of surveyed populations. The datasets contain the columns: `Id`, `Age`, `Country`, `City`, `Employed`, and `Vaccinated`.

We define analysis actions (Def. 2.1), analysis trees (Def. 2.2), and context trees (Def. 2.3) as follows.

Definition 2.1 (An Analysis Action). An analysis action α on a dataset $D_i \in D$ is a tuple $\alpha = (\tau, a, \kappa, \Omega, \Lambda)$ where, $a \in A, \tau \in \mathcal{T}, \kappa \in \{\emptyset\} \cup \mathcal{K}, \omega \in \{\emptyset\} \cup \Omega, \Lambda \in \{\emptyset\} \cup \{(a, \gamma) \mid a \in A, \text{ and } \gamma \in \Gamma\}$.

The set $\mathcal{T} = \{\texttt{Projection}, \texttt{Filter}, \texttt{Group}, \texttt{Sort}\}$ includes all possible action types. $\mathcal{K}$ includes operators such as $=, \leq, >$, while Ω contains literal values used in filtering operations (e.g., filtering rows where `Country` equals `"France"`). The set $\Gamma = \{\texttt{count}, \texttt{min}, \texttt{max}, \texttt{sum}, \texttt{avg}\}$ specifies aggregation functions for column operations (e.g., `avg(Age)`, `count(City)`). The component Λ pairs columns with these aggregation functions. Note that the action type and column are always present, while the other components are optional.

An example of an action is $(\texttt{Filter}, \texttt{Country}, =, \texttt{"France"})$. Henceforth, we refer to analysis actions simply as *actions*.

The recommendation task involves suggesting possible components of an action (e.g., column) or combinations of components (e.g., action type and column).

Definition 2.2 (Analysis Tree). Given a dataset $D_i \in D$, let $\Psi = (u_0, U_\Psi, E_\Psi)$ be an analysis tree, where:
$u_0 \mapsto \Delta_0 \mapsto D_i$ is the root node; $U_\Psi = \{u_0, \dots, u_r, \dots\}$ is the finite set of nodes, ordered such that $0 < \cdots < r < \dots$; E_Ψ is the set of directed edges; Each node u_r maps to an action result Δ_r i.e. $u_r \mapsto \Delta_r$; A directed edge $e_s = (u_r, u_s) \in E_\Psi$ exists if an action α_s is performed on Δ_r resulting in Δ_s. Thus, each edge $e_s \mapsto \alpha_s$.

Continuing the example, Figure 2a illustrates an analysis tree representing a user's exploration session.

Definition 2.3 (δ-Context Tree). Let $\Psi = (u_0, U_\Psi, E_\Psi)$ be an analysis tree. The δ-context tree of an action α_r is defined as the minimal (i.e., smallest connected) sub-tree of Ψ that includes the δ most recent nodes preceding the edge $e_r \mapsto \alpha_r$. Formally, it contains the nodes $\{u_{r-\delta}, u_{r-\delta+1}, \dots, u_{r-1}\} \subseteq U_\Psi$. The corresponding context tree is denoted $\psi = (U_\psi, E_\psi)$. Note that a δ-context tree exists for α_r if and only if $r - \delta \geq 0$.

Figure 2b illustrates the 3-context tree for action α_7 of the analysis tree shown in Figure 2a.

2.1 Problem Definition

Intuitively, our goal is to estimate the likelihood that a given context tree sequence leads to a particular action. Each action corresponds to a candidate, which we represent as an object. We recommend the top-n most probable actions based on this likelihood.

For instance, in recommending columns, each candidate represents a different column. As described in the next section, we represent these candidates using a graph structure called the *Candidate Representative Graph.*

Definition 2.4 (Objective). Given a context tree (Def. 2.3) sequence $[\psi_1, \dots, \psi_T]$ and an object ϕ (e.g. a graph, a vector, an image etc.) representing a candidate $\zeta \in Z$, find a function $f(\cdot, \cdot)$ that estimates the likelihood that the given sequence leads to the action represented by ϕ, i.e.,

$$f([\psi_1, \dots, \psi_T], \phi) \in [0, 1].$$

Figure 3 shows a simple context tree sequence based on the analysis tree (Fig. 2a).

3 ExplorAct Framework

To support intelligent recommendations during data exploration, we propose *ExplorAct*, a framework designed to predict the subsequent analysis action a user is likely to take. ExplorAct leverages both the contextual properties of recent user interactions and the sequential behavior inherent in exploratory tasks. The framework integrates tree-based and sequence-based modelling with uncertainty reasoning to provide accurate and interpretable recommendations. We now describe the core components of ExplorAct, including its recommendation tasks, overall architecture, and key design choices.

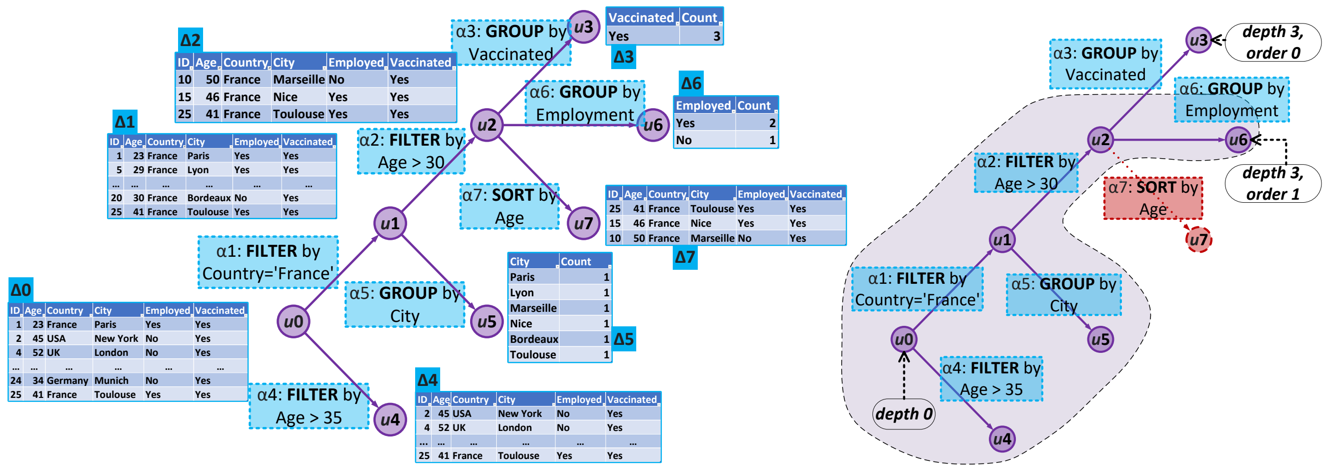
The action-type recommendation task is denoted as τ-rec, the column recommendation task as a-rec, and the combined action-type and column recommendation task as (τ, a)-rec. The user's choice of action-type and the corresponding column are key factors that determine their analysis trajectory.

3.1 Framework Overview

A high-level overview of the ExplorAct framework is shown in Figure 4. The pipeline comprises four core stages: candidate representative object generation, training, evidence fusion, and recommendation.

- **Candidate Representative Object Generation:** Since the problem is modeled as a candidate probability prediction task, effectively representing candidate-specific information is critical. To this end, we introduce a simple yet effective method called *candidate representative graphs.* These graphs capture candidates' characteristics, aiding the model in predicting actions from context sequences.
- **Training:** We formulate two types of context tree sequences to serve as our model's input. These sequences are used in supervised training tasks across all three recommendation types (τ-rec, a-rec, and (τ, a)-rec). The model is trained to learn sequence-based patterns that lead to specific candidates.
- **Evidence Fusion:** Once the models for the individual tasks are trained, we treat each as a source of uncertain evidence. We use Dempster-Shafer Theory (DST) to fuse their outputs, generating improved predictions for the (τ, a)-rec task. DST enables us to combine the strengths of each model while accounting for uncertainty in their predictions.
- **Recommendation:** Based on the fused candidate probabilities, we select the top-n candidate actions to recommend to the user as potential next steps.

ExplorAct is designed to leverage the hierarchical and contextual information in context trees and the temporal patterns in sequences of analysis actions. We hypothesize that Graph Isomorphism Networks with Edge-features (GINE)[21, 40], a state-of-the-art Graph Neural Network (GNN) model, can effectively extract rich, latent



Δ0

ID	Age	Country	City	Employed	Vaccinated
1	23	France	Paris	Yes	Yes
2	45	USA	New York	No	Yes
4	52	UK	London	No	Yes
...	...	...	...	...	...
24	34	Germany	Munich	No	Yes
25	41	France	Toulouse	Yes	Yes

Δ1

ID	Age	Country	City	Employed	Vaccinated
1	23	France	Paris	Yes	Yes
5	29	France	Lyon	Yes	Yes
...	...	...	...	...	...
20	30	France	Bordeaux	No	Yes
25	41	France	Toulouse	Yes	Yes

Δ2

ID	Age	Country	City	Employed	Vaccinated
10	50	France	Marseille	No	Yes
15	46	France	Nice	Yes	Yes
25	41	France	Toulouse	Yes	Yes

Vaccinated	Count
Yes	3

Δ3

Δ6

Employed	Count
Yes	2
No	1

ID	Age	Country	City	Employed	Vaccinated
25	41	France	Toulouse	Yes	Yes
15	46	France	Nice	Yes	Yes
10	50	France	Marseille	No	Yes

Δ7

City	Count
Paris	1
Lyon	1
Marseille	1
Nice	1
Bordeaux	1
Toulouse	1

Δ5

Δ4

ID	Age	Country	City	Employed	Vaccinated
2	45	USA	New York	No	Yes
4	52	UK	London	No	Yes
...	...	...	...	...	...
25	41	France	Toulouse	Yes	Yes

(a) An example analysis tree. Node u_0 represents the initial dataset Δ_0. An action α_j on an action result Δ_i represented by node u_i leads to the action result Δ_j represented by node u_j. See Def. 2.2.

(b) The 3-context tree for action α_7 from the analysis tree presented in Figure 2a (Def. 2.3). Positional identity assignments of nodes 3 and 6 are shown (Sec. 3.3.2).

Figure 2: Analysis Tree and Context Tree

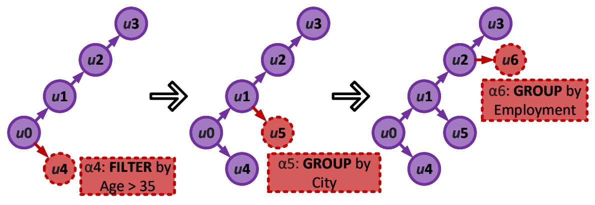


Figure 3: Context Tree Sequence for Action Sequence $[\alpha_4, \alpha_5, \alpha_6]$.

representations from context trees. We select GINE due to its demonstrated performance on graph isomorphism tasks, which are essential for capturing nuanced structural variations in user interaction histories. Section 3.2 describes how we construct feature vectors for the nodes and edges of the analysis trees to be consumed by the GINE layers.

To model sequential patterns, we employ Gated Recurrent Units (GRUs)[6]. We opt for GRUs instead of Long Short-Term Memory (LSTM)[18, 20] models due to their lower computational cost[7] and the relatively short lengths of analysis action sequences (typically up to 25 steps). This combination enables efficient and accurate modeling of user exploration behavior.

3.2 Node and Edge Feature Vectors

To be processed by GINEs, feature vectors are assigned to nodes and edges of analysis trees, with features inherited by context trees. ExplorAct employs a simple method for node feature vector generation. First, generate feature vectors for each column of the action result. Then, concatenate those vectors to form the action result vector. Finally, dimensionality is reduced to produce node feature vectors. For each column, including those from aggregations (`count`, `min`, `max`, `sum`, `avg`), we compute a probability vector over bins, unique items, or text templates.

We need to define new columns based on analysis actions originating from aggregation functions. For instance, applying a `count` on the `City` column when grouping by `Country` creates a `"City-count"` column. Columns generated from aggregation functions on different groupings, like `"Age-avg"` when grouping by `Country` or `City`, are considered similar. Finding all possible columns involves applying aggregation functions on the original columns, constrained by their data types.

Numerical columns are allowed with any of the five aggregation functions. Nominal categorical and text columns are limited to the `count` function, while ordinal categorical columns can use `count`, `min`, and `max` functions. An example of potential columns are `["Age-avg", "City-count", "Employed-count", . . . ]`.

3.2.1 Numerical Columns. Original numerical columns, their aggregated forms, and count aggregations remain numerical. We discretize numerical columns into equal-width bins. The bins for any original column or its emergent `avg`, `min`, and `max` columns are defined on the range $[\min a(\mathcal{D}), \max a(\mathcal{D}) + \epsilon)$. For emergent `count` columns, the range is $\left[0, \max_{i=1}^{m} |a(D_i)| + \epsilon\right)$. $\left[\min_{i=1}^{m} \sum_{\substack{x\in a(D_i)\\ x<0}} x,\ \left(\max_{i=1}^{m} \sum_{\substack{x\in a(D_i)\\ x\geq 0}} x\right) + \epsilon\right)$ is the range for emergent `sum` columns. Positive increment choices, $\epsilon > 0$, are arbitrary. A range is divided into b bins. Then the probability vector of a numerical column a of an action result Δ_r is

$$\mathbf{P}_a = \left[P(B_1^a \mid a(\Delta_r)), P(B_2^a \mid a(\Delta_r)), \ldots, P(B_b^a \mid a(\Delta_r))\right]$$

with B_i^a denoting the i-th bin.

3.2.2 Categorical Columns. Original categorical columns and `min`-`max` aggregations of ordinal columns are treated as categorical. We can form a probability vector for the unique column values using the action result Δ_r and the categorical column a. (q_i denotes a unique value, and $|Q|$ is the count of unique values.)

$$\mathbf{P}_a = \left[P(q_1 \mid a(\Delta_r)), P(q_2 \mid a(\Delta_r)), \ldots, P(q_{|Q|} \mid a(\Delta_r))\right]$$

3.2.3 Text Columns. We extract a finite set of common patterns, or templates, from text values using a rule-based algorithm with regular expressions, though other methods can be used. The calculated probability vector for a text column a in an action result Δ_r is

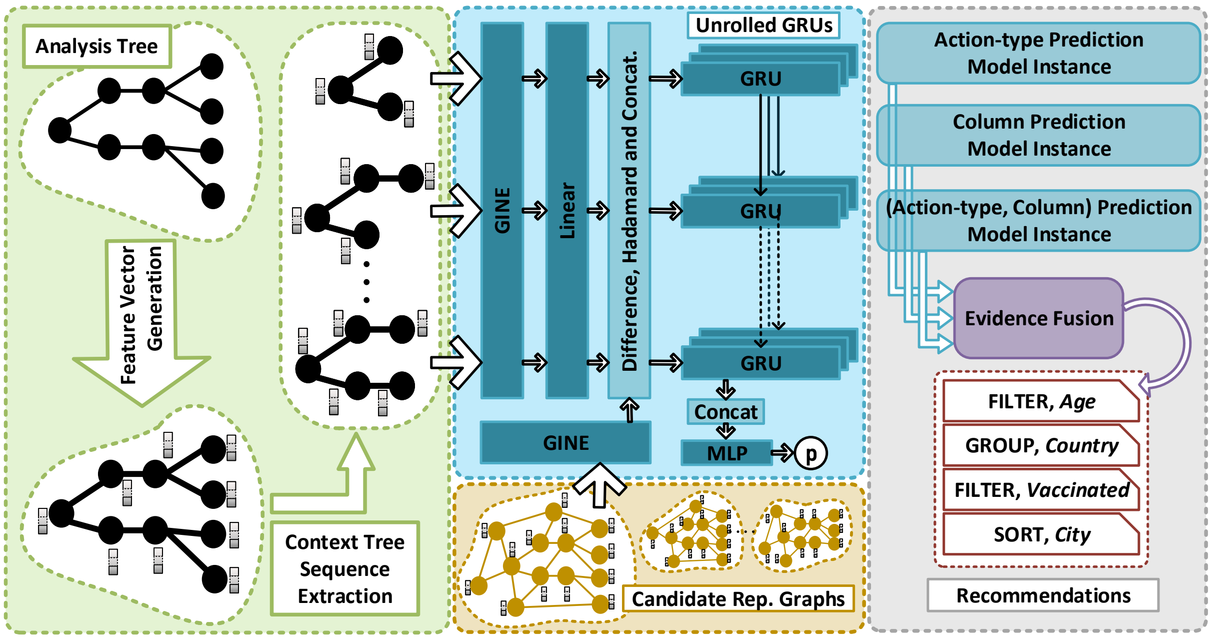

Analysis Tree
Feature Vector Generation
Context Tree Sequence Extraction
GINE
Linear
Difference, Hadamard and Concat.
Unrolled GRUs
GRU
GRU
GRU
Concat
MLP
p
GINE
Candidate Rep. Graphs
Action-type Prediction Model Instance
Column Prediction Model Instance
(Action-type, Column) Prediction Model Instance
Evidence Fusion
FILTER, Age
GROUP, Country
FILTER, Vaccinated
SORT, City
Recommendations

Figure 4: ExplorAct Overview. Data-flow of the system is shown by the arrows.

denoted as

$$\mathbf{P}_a = \left[P(t_1 \mid a(\Delta_r)), P(t_2 \mid a(\Delta_r)), \ldots, P(t_{|T|} \mid a(\Delta_r))\right]$$

, where t_j represents a template and $|T|$ the number of templates.

3.2.4 *Node Feature Vector.*

For a node u_r representing an action result Δ_r in an analysis tree, the feature vector is given by $v(\Delta_r) = \bigoplus_{a \in C} \mathbf{P}_a$. The feature dimension d_v can be large. Hence, we apply principal component analysis (PCA) to reduce its dimension by selecting the top-$d_{v'}$ components, forming the final node feature vector $v(u_r) = v'(\Delta_r)$.

3.2.5 *Edge Feature Vectors.*

For each edge in a context tree, the edge feature vector v_α is formed by concatenating a one-hot encoded action-type with a one-hot encoded column.

3.3 Candidate Representative Graph

Our objective is to evaluate a context tree sequence's probability leading to an action represented by a candidate object (2.4). Therefore, we construct Candidate Representative Graphs as model-understandable representative objects (ϕ) of candidates in recommendation tasks. The construction involves: (1) Separating context trees into candidate sets, (2) Assigning positional identities to context tree nodes, and (3) Defining the nodes, edges, and feature vectors of the Candidate Representative Graph.

3.3.1 *Context Tree Separation.*

Consider an extracted list of δ-context trees for specific actions. Candidate sets are defined based on the recommendation task, separating the trees accordingly. For an example, assume that the context trees $[\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6]$ lead to actions,

$$[(\texttt{Filter}, \texttt{Age}, >, 30), (\texttt{Group}, \texttt{Vaccinated}, \emptyset, \emptyset, \{(\texttt{Id, count})\}),$$
$$(\texttt{Filter}, \texttt{Age}, >, 35), (\texttt{Group}, \texttt{City}, \emptyset, \emptyset, \{(\text{Id}, \text{count})\}),$$
$$(\texttt{Group}, \texttt{Employed}, \emptyset, \emptyset, \{(\texttt{Id, count})\}), (\texttt{Sort}, \texttt{Age})]$$

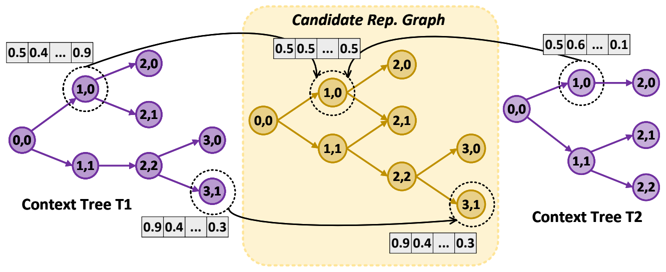


Figure 5: Construction of a Candidate Representative Graph.

For τ-rec, the candidates are {Filter, Group, Sort}; for a-rec, {Age, Vaccinated, City}; and for (τ, a)-rec,{(Filter, Age), (Group, Vaccinated), (Group, Employed), (Group, City),(Sort, Age)}. For τ-rec, trees are divided as {Filter:$[\psi_1, \psi_3]$, Group:$[\psi_2, \psi_4, \psi_5]$, Sort:$[\psi_6]$}. We denote candidate sets for tasks τ-rec, a-rec and (τ, a)-rec as $Z_{\mathcal{T}}$, Z_A, and $Z_{\mathcal{T}\times A}$.

3.3.2 *Positional Identity Assignment.*

Positional identity assignment labels nodes in a context tree to indicate their location. The label ρ_u is used for a node u. The node's position is determined by its depth $\mathbf{d}$ in the tree and its sequential index $\mathbf{i}_d$ at that depth. $\mathbf{i}_d$ is based on the order of the node's action result. Thus, positional identity is a pair $\rho_u = (\mathbf{d}, \mathbf{i}_d)$. In the 3-context tree (Figure 2b) of action α_7, nodes at depth 3 are $[u_3, u_6]$, representing $[\Delta_3, \Delta_6]$ action results. Indices 0 and 1 are assigned to u_3 and u_6 based on Δ_3 occurring before Δ_6. Hence, positional identities for u_3 and u_6 are $(3, 0)$ and $(3, 1)$.

3.3.3 *The Candidate Representative Graph.*

A representative graph (Def. 3.1) captures collective structure, node, and edge representations of a given set of context trees.

Definition 3.1 (Candidate Representative Graph). Consider a collection of context trees wherein nodes are attributed with positional identities. Denote $\{\rho_1, \rho_2, \ldots, \rho_n\}$ as the resultant set of unique positional identities. Consequently, a representative graph is defined as a graph $G = (U_G, E_G)$, characterized by $U_G = \{u_{\rho_1}, u_{\rho_1}, \ldots, u_{\rho_n}\}$ with $u_{\rho_i} \mapsto \rho_i$ and $\exists(u_{\rho_i}, u_{\rho_j}) \in E_G$ (signifying a directed edge) if and only if there exists at least one edge from a node possessing the ρ_i positional identity to a node possessing the ρ_j positional identity within any of the context trees in the given set.

A representative graph is a connected, directed acyclic graph, but not necessarily a tree. Its node and edge feature vectors are calculated by averaging feature vectors from context trees with matching settings as cited in Def. 3.1. For instance, a node's feature vector with a $(3, 1)$ positional identity label in a G results from averaging all similar nodes in context trees with $(3, 1)$ positional identity. Likewise, an edge's feature vector in G, from a node with $(3, 1)$ to $(4, 3)$ positional identities, is the average of all similar edges in context trees, transitioning from nodes with $(3, 1)$ to $(4, 3)$ positional identities (See figure 5).

3.4 Context Tree Sequence Formulation

We propose two different context tree sequence formulations, each constructed by considering distinct perspectives that lead to the next action. They are described as follows.

3.4.1 State-Perspective Context Tree Sequence: We define this as the sequence of context trees corresponding to a series of actions, constrained by a specific context size δ. If an action lacks a δ-context tree, the largest available context tree is used instead (e.g., if the maximum context tree for an action is 5, the 5-context tree is selected when needing an 8-context tree). This sequence encapsulates the user's state at each action step.

3.4.2 Multi-Perspective Context Tree Sequence: We retrieve context trees of sizes from 1 to δ for a given action. By ordering these trees in descending size, we create this sequence, which presents information from oldest to newest, offering multiple perspectives on the action.

These formulations provide a comprehensive approach to modeling user actions, enhancing the accuracy and relevance of our recommendations.

3.5 Model Function

This section outlines our model functions. We define $\mathbf{Latent}(\cdot)$ for processing context trees and graphs, where BN is Batch Normalization, σ is the ReLU activation, and $\sigma_{\mathcal{L}}$ is the Leaky ReLU activation. The $\mathbf{Latent}(\cdot)$ updates node features at layer l, $(h_u^{(l)})$ using edge features (h_{uv}) with the GINE update rule,

$$h_u^{(l)} = \sigma\left(\mathrm{BN}\left(\mathrm{MLP}\left((1+\epsilon)\, h_u^{(l-1)} + \sum_{v \in N(u)} \sigma\left(h_v^{(l-1)} + W \cdot h_{uv}\right)\right)\right)\right) \tag{1}$$

The MLP is a multi-layer perceptron. Global add pooling is applied, pooling vectors from each GINE layer are concatenated, then passed through a linear projection and a Leaky ReLU to produce the final output.

$$h_G^{(l)} = \sum_{u \in G} h_u^{(l)} \qquad h_G^{\oplus} = \bigoplus_{l=1}^{L} h_G^{(l)} \qquad h_G = \sigma_{\mathcal{L}}\left(W_p \cdot h_G^{\oplus} + b_p\right) \tag{2}$$

Given a sequence of context trees, and a candidate representative graph G_ζ, we use a dedicated $\mathbf{Latent}(\cdot)$ for context trees and another $\mathbf{Latent}(\cdot)$ for processing G_ζ.

$$x'_\psi = \mathbf{Latent}_\Psi(\psi) \qquad x_\zeta = \mathbf{Latent}_G\left(G_\zeta\right) \tag{3}$$

For each x'_ψ, compute the Hadamard product $(\odot)$, find its difference with x_ζ, and then concatenate them.

$$x_\psi = x'_\psi \oplus \left(x'_\psi - x_\zeta\right) \oplus \left(x'_\psi \odot x_\zeta\right) \tag{4}$$

For a context tree sequence $[\psi_1, \ldots, \psi_t, \ldots, \psi_T]$, we can obtain the concatenated hidden representations, $[x_{\psi_1}, \ldots, x_{\psi_t}, \ldots, x_{\psi_T}]$. Then, for an input vector x_{ψ_t} at step t, and the previous hidden state h_{t-1}, the GRU updates are defined as,

$$z_t = Sigmoid(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \qquad \text{(Update gate)}$$
$$r_t = Sigmoid(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \qquad \text{(Reset gate)}$$
$$h'_t = \tanh(W_h \cdot x_t + U_h \cdot (r_t \odot h_{t-1}) + b_h) \qquad \text{(Prospective hid. state)}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t \qquad \text{(New hid. state)}$$

In a stacked GRU with L layers, let the hidden states from each GRU layer at the final step T be $[h_T^{(1)}, \ldots, h_T^{(L)}]$. We concatenate the final hidden states and it is concatenated with the latent representation x_ζ of G_ζ. Finally, h undergoes a linear layer followed by Sigmoid function to estimate the likelihood of ψ_δ leading to the next action of candidate ζ.

$$h_T^{\oplus} = \bigoplus_{l=1}^{L} h_T^{(l)} \quad h = x_\zeta \oplus h_T^{\oplus} \quad z = W \cdot h + b \quad p = Sigmoid(z) \in [0, 1] \tag{5}$$

W_* and U_* are learnable weight matrices, b_* are learnable bias vectors.

3.6 Training

Given a set of analysis trees and a context size δ, training proceeds as follows:

- Context tree sequences are generated using a selected formulation (see Section 3.4).
- Candidate representative graphs are constructed from context trees in the training sequences.
- Each sequence is paired with each candidate graph G_ζ, for $\zeta \in Z_*$.
- A binary label $y \in \{0, 1\}$ is assigned: 1 if G_ζ matches the true candidate of ψ_δ; 0 otherwise.

Model training uses backpropagation with binary cross-entropy loss:

$$Loss = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i)\right]$$

where p_i is the predicted probability and y_i the ground truth label.

3.7 Recommendation Inference

Given a test sequence of context trees (constructed according to one of the defined formulations), we begin by generating pairs $([\psi_1, \ldots, \psi_T], G_\zeta)$ for all candidates $\zeta \in Z_*$ obtained from the training set. For each of these pairs, we compute the model's output. By selecting the top-n values among these outputs, we identify the top-n most likely candidates corresponding to the given input sequence. These predicted candidates directly correspond to the top-n next action recommendations.

3.8 Evidence Fusion

Dempster-Shafer Theory (DST) is an evidence-based reasoning framework[8, 41] that accounts for all possible outcomes in a problem space. It is beneficial when handling uncertainty, conflicting information, and imprecise data without the need for strict priors or assumptions required by methods like Bayesian inference[15, 19, 33, 34, 42]. When presented with multiple sources of evidence, one can combine them using Dempster's rule of combination[35]. It has been extensively utilized in computer vision[9, 31, 32], sensor data fusion[1], and multi-modal recommendation[39].

Let Θ denote the *frame of discernment*, a finite set of mutually exclusive and exhaustive hypotheses. DST defines a *basic belief assignment* (BBA), or mass function, as, $m : 2^{\Theta} \longrightarrow [0,1]$ such that $m(\emptyset) = 0$ and $\sum_{X \subseteq \Theta} m(X) = 1$. Each value $m(X)$ represents the amount of belief assigned exactly to the subset $X \subseteq \Theta$. From the mass function m, two key functions are derived; the *belief function* $Bel(X) = \sum_{Y \subseteq X} m(Y)$, representing the total belief that supports X and the *plausibility function* $Pl(X) = \sum_{Y \cap X \neq \emptyset} m(Y)$, which measures how much belief could potentially support X. Given two independent mass functions m_1 and m_2, Dempster's rule of combination defines a new mass function m as:

$$m(W) = \frac{\sum_{X \cap Y = W} m_1(X) \cdot m_2(Y)}{1 - \sum_{X \cap Y = \emptyset} m_1(X) \cdot m_2(Y)} \quad \text{for, } Z \neq \emptyset$$

To refine recommendation confidence, we perform evidence fusion over model outputs. We begin by normalizing predicted likelihoods into probabilities, dividing each by the total likelihood across all candidates. Then, we apply *Dempster's combination rule* to fuse belief masses associated with their respective frames of discernment.

$$\Theta_\tau = \{\emptyset, \text{Filter}, \text{Projection}, \text{Group}, \text{Sort}\}, \quad \Theta_a = \{\emptyset, a_1, \ldots, a_{|A|}\}$$

This results in probability mass assignments over the joint space $\Theta_\tau \times \Theta_a$, which are then further combined with the mass function from the model that predicts the joint action-type and column pairs, represented by $\Theta_{(\tau,a)}$. Finally, we compute *belief values* $Bel(\cdot)$ over these fused outcomes. The resulting belief scores are used to determine the top-n most probable action-type and column pairs, which form the final set of recommendations.

4 Experimental Evaluation

Our experiments are designed to address the following questions:

- Do our models achieve higher recommendation accuracy compared to the state-of-the-art (SOTA) approach?
- Does evidence fusion improve the accuracy of action-type and column-pair recommendations?
- Do our models achieve constant, sub-second inference times?

4.1 Experimental Setup

We evaluate our models using the REACT-IDA next-action recommendation benchmark [16], which consists of four datasets. The benchmark includes session logs from 56 cybersecurity analysts exploring four network traffic datasets provided by the Honeynet Project [37] to find security events (e.g. hacking, malware infiltrations etc.). This benchmark is recognized as the de-facto in evaluating recommender systems for next action recommendations in modern IDA platforms [29].

Table 2: Number of cases per context tree size.

Context Size	3	4	5	6	7	8
Total Cases	1185	918	714	552	435	350

To simulate realistic exploratory behavior, sessions were user-separated and randomly shuffled, mimicking concurrent dataset analysis by multiple users. These shuffled sessions were split into five balanced folds, ensuring that all sessions from a single user appear in only one fold, thereby avoiding bias from repeated patterns. We generated three such scenarios using different random seeds. We report the averaged accuracy values with their standard deviations. The number of recorded sessions per dataset is 158, 120, 96, and 80, respectively. We consider the context sizes ranging from 3 to 8 and their distribution is shown in Table 2.

Experiments were conducted on a computing cluster node equipped with an 80GB NVIDIA A100 GPU, 8GB of RAM, and a 4-core Intel(R) Xeon(R) Gold 6326 CPU (2.90 GHz). The implementation code is available at GitHub[1].

4.2 Baselines and Models

We compare our models against the following baselines:

- *Basic GRU*: A sequence-based model that takes one-hot encoded action sequences and predicts the next action as a multi-class classification task.
- *Basic GINE*: A graph-based model that processes δ-context trees to predict the next action. Node features are one-hot encodings of node numbers within their respective analysis trees.
- *REACT*: The current SOTA method for next-action recommendation in IDA platforms. It employs a k-nearest neighbor algorithm based on a custom tree edit distance, which incorporates specialized node and edge similarity functions [28].

We train two versions of our model for τ-rec and a-rec tasks to account for different sequence representations, resulting in four model variations for the (τ, a)-rec task. The model variants are:

- *EA-SP*: *ExplorAct* models based on *State-Perspective Context Tree Sequences.*
- *EF-SP*: *EA-SP* models enhanced with Dempster-Shafer Theory (DST) evidence fusion (EF). Applicable only to (τ, a)-rec.
- *EA-MP*: *ExplorAct* models based on *Multi-Perspective Context Tree Sequences.*
- *EF-MP*: *EA-MP* models with DST-based evidence fusion. Applicable only to (τ, a)-rec.

4.3 Implementation Details

All models were implemented in Python using PyTorch, PyTorch Geometric, and scikit-learn. Training was conducted on four folds, with the fifth used for testing, rotating folds across for each scenario.

For REACT, we used the official implementation and distance functions provided in their public GitHub repository [16]. We did not set a custom distance threshold to ensure coverage across all test cases. The frequency threshold was set to 0.1, as recommended in the original paper. For a given training set, we divide it into 5-folds to find optimal k for the kNN algorithm of REACT by choosing the k value that gives the best average accuracy across those folds. k

[1]https://github.com/DinukaManohara/exploract

Table 3: R@1 values for τ-rec task (the higher the better). The subscripted values are standard deviations.

δ	3	4	5	6	7	8
GRU	$.44_{.06}$	$.43_{.05}$	$.43_{.07}$	$.46_{.05}$	$.46_{.06}$	$.45_{.08}$
GINE	$.61_{.05}$	$.59_{.04}$	$.56_{.05}$	$.56_{.06}$	$\underline{.56}_{.06}$	$.55_{.06}$
REACT	$.61_{.06}$	$.59_{.06}$	$.56_{.07}$	$.53_{.08}$	$.54_{.08}$	$.54_{.06}$
EA-SP	$\underline{.62}_{.05}$	$\underline{.60}_{.07}$	$\underline{.59}_{.07}$	$\underline{.57}_{.08}$	$.56_{.09}$	$\underline{.56}_{.10}$
EA-MP	$\mathbf{.64}_{.06}$	$\mathbf{.63}_{.05}$	$\mathbf{.62}_{.06}$	$\mathbf{.62}_{.06}$	$\mathbf{.61}_{.06}$	$\mathbf{.63}_{.07}$
Improvement	4.92%	6.78%	10.71%	16.98%	12.96%	16.67%

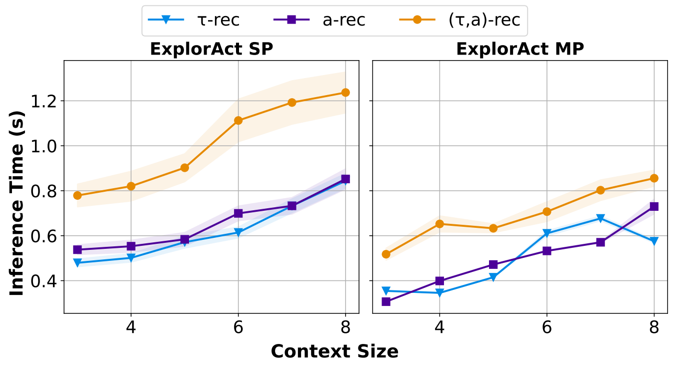


Figure 6: ExplorAct inference times on CPU.

values are selected from the integer value range $\left[\left\lfloor\frac{\sqrt{size}}{2}\right\rfloor, \left\lceil\frac{3\sqrt{size}}{2}\right\rceil\right]$, where *size* is the size of the training set.

All deep learning models were trained using the Adam optimizer with a learning rate of 1×10^{-4} and weight decay of 1×10^{-7}. Model architecture parameters, such as the number of hidden layers and dimensionality, were kept consistent across all methods for fairness. Each model instance was run five times with different training seeds (this is different from scenario seeds).

4.4 Evaluation Framework

We evaluate the recommendations from the models against the real next actions taken in the analysis sessions and adopt standard evaluation metrics commonly used in recommender systems:
① **Recall@n (R@n)**: Measures whether the true next action appears within the top-n recommended actions.
② **Mean Reciprocal Rank (MRR)**: Evaluates the recommendation ranking quality, assigning higher weight to correct actions ranked earlier.

We assess performance on three tasks: τ-rec (recommend action type), a-rec (recommend column pairs), and (τ, a)-rec (joint recommendation). Due to the small number of candidates in τ-rec (four candidates), we use R@1. For a-rec and (τ, a)-rec, we report both R@3 and MRR. Finally, we measure inference latency on the CPU to assess the practical feasibility of deploying this system in real-world settings.

4.5 Results Comparison

We conduct a comprehensive comparison of ExplorAct and other methods across three recommendation tasks, with results showing its superior performance in realistic scenarios derived from benchmark session logs.
τ-rec and a-rec Performance: Both ExplorAct models consistently outperform naive methods and REACT across all context

Table 4: R@3 and MRR values for a-rec task (higher the better). The first row of each model contains R@3 values, whereas the second contains MRR values. Subscripted values are standard deviations.

δ	3	4	5	6	7	8
GRU	$.47_{.04}$	$.46_{.05}$	$.46_{.05}$	$.47_{.05}$	$.47_{.05}$	$.50_{.06}$
	$.26_{.03}$	$.27_{.03}$	$.28_{.03}$	$.28_{.03}$	$.28_{.04}$	$.30_{.04}$
GINE	$.60_{.04}$	$.59_{.04}$	$.57_{.04}$	$.55_{.04}$	$.54_{.05}$	$.54_{.05}$
	$.42_{.04}$	$.40_{.03}$	$.38_{.03}$	$.36_{.03}$	$.35_{.04}$	$.36_{.04}$
REACT	$.62_{.05}$	$.61_{.06}$	$.57_{.06}$	$.54_{.06}$	$.53_{.04}$	$.53_{.06}$
	$.43_{.04}$	$.40_{.03}$	$.39_{.04}$	$.36_{.04}$	$.32_{.04}$	$.32_{.03}$
EA-SP	$\underline{.63}_{.05}$	$\underline{.62}_{.05}$	$\underline{.61}_{.05}$	$\underline{.59}_{.06}$	$\underline{.56}_{.05}$	$\underline{.57}_{.08}$
	$\underline{.43}_{.03}$	$\underline{.42}_{.04}$	$\underline{.40}_{.04}$	$\underline{.38}_{.04}$	$\underline{.37}_{.05}$	$\underline{.38}_{.06}$
EA-MP	$\mathbf{.69}_{.04}$	$\mathbf{.68}_{.04}$	$\mathbf{.67}_{.05}$	$\mathbf{.66}_{.05}$	$\mathbf{.64}_{.05}$	$\mathbf{.63}_{.06}$
	$\mathbf{.48}_{.03}$	$\mathbf{.47}_{.03}$	$\mathbf{.46}_{.04}$	$\mathbf{.44}_{.04}$	$\mathbf{.43}_{.04}$	$\mathbf{.42}_{.05}$
Improvement	11.29%	11.48%	17.54%	22.22%	20.75%	18.87%
	11.63%	17.5%	17.95%	22.22%	34.38%	31.25%

Table 5: R@3 and MRR values for (τ, a)-rec task (higher the better). The first row of each model contains R@3 values, whereas the second contains MRR values. Subscripted values are standard deviations.

δ	3	4	5	6	7	8
GRU	$.30_{.03}$	$.28_{.04}$	$.30_{.05}$	$.30_{.05}$	$.30_{.05}$	$.32_{.05}$
	$.19_{.02}$	$.19_{.03}$	$.20_{.03}$	$.20_{.03}$	$.20_{.04}$	$.22_{.04}$
GINE	$.42_{.04}$	$.40_{.04}$	$.38_{.04}$	$.35_{.04}$	$.34_{.05}$	$.35_{.05}$
	$.28_{.03}$	$.27_{.03}$	$.25_{.03}$	$.23_{.03}$	$.22_{.03}$	$.24_{.03}$
REACT	$.42_{.04}$	$.40_{.05}$	$\underline{.40}_{.05}$	$.37_{.06}$	$.33_{.05}$	$.34_{.05}$
	$.30_{.03}$	$.27_{.03}$	$\underline{.27}_{.03}$	$\underline{.26}_{.05}$	$.24_{.04}$	$.23_{.04}$
EA-SP	$.43_{.05}$	$.41_{.05}$	$.39_{.05}$	$\underline{.37}_{.05}$	$\underline{.37}_{.06}$	$\underline{.38}_{.05}$
	$.28_{.04}$	$.28_{.04}$	$.27_{.03}$	$.25_{.03}$	$.24_{.04}$	$\underline{.26}_{.04}$
EA-MP	$.39_{.05}$	$.38_{.04}$	$.39_{.04}$	$.36_{.05}$	$.35_{.05}$	$.36_{.05}$
	$.25_{.03}$	$.24_{.03}$	$.25_{.03}$	$.23_{.04}$	$\underline{.24}_{.03}$	$.25_{.04}$
EF-SP	$\underline{.44}_{.05}$	$\underline{.41}_{.05}$	$.39_{.07}$	$.34_{.08}$	$.34_{.06}$	$.34_{.09}$
	$\underline{.30}_{.03}$	$\underline{.28}_{.04}$	$.26_{.05}$	$.22_{.05}$	$.23_{.04}$	$.23_{.06}$
EF-MP	$\mathbf{.48}_{.04}$	$\mathbf{.44}_{.04}$	$\mathbf{.43}_{.06}$	$\mathbf{.42}_{.06}$	$\mathbf{.39}_{.07}$	$\mathbf{.40}_{.04}$
	$\mathbf{.33}_{.03}$	$\mathbf{.31}_{.03}$	$\mathbf{.30}_{.03}$	$\mathbf{.28}_{.04}$	$\mathbf{.27}_{.04}$	$\mathbf{.27}_{.03}$
Improvement	12.95%	10.88%	7.72%	12.9%	20.94%	16.76%
	11.51%	13.2%	9.13%	8.54%	13.87%	13.04%

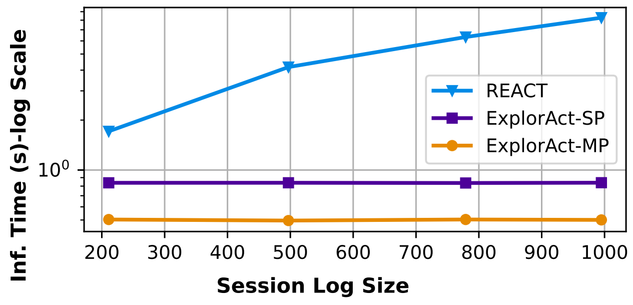


Figure 7: Varying inference times against the session log size (Only for context size 3 is depicted. Other context sizes are omitted for brevity, and they behave the same).

sizes. ExplorAct MP has 3.23-12.5% R@1 improvement in τ-rec and 9.52–14.29% R@3 with 10.53-16.22% MRR improvement in a-rec over ExplorAct SP, indicating the added value of multi-perspective context sequences for next action-type prediction. ExplorAct MP demonstrates 4.92–16.98% R@1 improvement in τ-rec (Table 3) and 11.29–22.22% R@3 with 11.63-34.38% MRR improvement in a-rec (Table 4) over the SOTA.

(τ, a)-rec Performance: ExplorAct MP achieves R@3 and MRR comparable to REACT. However, with evidence fusion (DST models), ExplorAct consistently outperforms REACT across all context sizes and scenarios. Notably, ExplorAct MP with evidence fusion surpasses REACT by 7.72-20.94% R@3 and 8.54-13.87% MRR, highlighting the impact of evidence fusion on performance (Table 5).

Effect of DST: ExplorAct MP with evidence fusion (*EF-MP*) shows 10.62-20.73% R@3 improvement and 7.52-31.53% MRR improvement over its non-evidence fused counterpart (*EA-MP*) across all test configurations.

Inference Times: Our experiments demonstrate that ExplorAct MP achieves sub-second average inference times across all three recommendation tasks. ExplorAct SP has sub-second average inference times for τ-rec and a-rec while achieving around $1.2s$ at maximum for (τ, a)-rec inference (Figure 6). In their user testing, REACT measured $40s$ between consecutive queries, justifying the inference times. However, in the information retrieval community, the threshold where the users notice the delay in responses is $1s$ [3], strengthening the need for sub-second recommendations. Furthermore, this demonstrates that the inference time depends on the maximum context size of a context tree sequence - inference time increases with the context size. In contrast, REACT's recommendation time depends on the session log size. This log-size dependent inference time behaviour of REACT is also revealed by our experiments (presented in Figure 7). [2]

4.6 Discussion

REACT relies on explicit similarity search, employing tree edit distance and k-nearest neighbors to retrieve context trees that resemble the current session. In contrast, *ExplorAct* learns latent representations of context trees using Graph Isomorphism Networks (GINs), eliminating the need for computationally expensive similarity calculations.

By framing the next-action recommendation task as a candidate probability prediction problem, each candidate action is associated with a representative object, namely a *Candidate Representative Graph—ExplorAct* supports a unified model architecture across different tasks. This design obviates the need for task-specific output layers, allowing for seamless incremental training. As a result, adding support for new candidates requires only the construction of corresponding representative graphs and training on the paired sequences, rather than retraining the entire model from scratch. Furthermore, as context size increases, the number of available training cases decreases (see Table 2). Despite this reduction, *ExplorAct* consistently achieves the most substantial accuracy improvements over REACT, indicating superior generalization and sample efficiency compared to the current state-of-the-art.

[2]It is worth noting that the inference times of our REACT implementation are slightly higher than those reported in the original paper due to the use of external functions for distance computations. However, our analysis focuses on trend behavior rather than absolute performance values. This however, has no impact on the accuracy reported.

5 Related Work

The recommendation of subsequent actions in data exploration can be broadly categorized into two areas: i) SQL/OLAP-based recommendations, and ii) recommendations within modern Interactive Data Analysis (IDA) platforms.

SQL/OLAP-based recommendations. Recommender systems designed for SQL/OLAP environments typically propose complete or partial SQL queries [17, 23, 24, 38], query templates [24], specific query predicates [17, 27], or complete sessions composed of multiple SQL queries [14]. Some systems also suggest data items that may capture user interest [13, 26] or predict future accesses to facilitate data prefetching [43]. Identifying interesting regions of data is often handled by active learning frameworks, commonly referred to as *explore-by-example* [11, 12, 22, 30]. In such systems, recommendations are iteratively refined based on user feedback on prior suggestions.

Modern IDA platforms. Within modern IDA platforms, REACT [28] formalizes the problem setting addressed in this paper and introduces the *analysis tree session benchmark* for evaluating next-action recommenders. Another approach [36] builds on REACT's methodology to predict interestingness measures that best characterize the result sets of analysis actions. A closely related line of work involves generating a complete sequence of exploratory operations in a Python notebook, based on patterns derived from previously authored notebooks [5]. This approach assumes prior knowledge of the types of insights users typically aim to uncover.

More recently, the *explainability* of analysis actions has been explored using interestingness measures [10]. Additionally, some user interfaces [4, 25] incorporate deep reinforcement learning to assist users in exploring datasets within modern IDA platforms.

6 Conclusion and Future Work

This paper addressed the challenge of recommending next analytical actions in multi-dataset settings on modern IDA platforms. We introduced ExplorAct, the first GIN-based recommender for context trees, capturing both hierarchical and sequential aspects of analysis sessions. Unlike prior retrieval-based methods like REACT, our approach models user behavior using context tree sequences and combines multiple prediction models for improved accuracy.

By applying GNN architectures, we uncover latent representations of user states—advancing beyond prior similarity-based use of context trees. Experiments on real-world datasets show our models outperform the state of the art on τ-rec, a-rec, and (τ, a)-rec by up to 16.98% R@1, 22.22% R@3 (34.38% MRR), and 20.94% (13.87% MRR), respectively, all while maintaining sub-second inference times. Our formulation as a candidate probability prediction problem also allows for efficient incremental training.

Future work includes exploring positional encoding for identifying key context trees, leveraging generative models for candidate representative graphs, and extending our method to predict operators and predicate values, which involve complex semantics and large combinatorial spaces.

Acknowledgments

We gratefully acknowledge support from the Australian Research Council Discovery Early Career Researcher Award DE230100366, and L'Oreal-UNESCO For Women in Science 2023 Fellowship.

7 GenAI Usage Disclosure

We utilized the grammar checking tool Grammarly and its AI capabilities to improve clarity of our texts.

References

[1] Michael Aeberhard, Sascha Paul, Nico Kaempchen, and Torsten Bertram. 2011. Object existence probability fusion using dempster-shafer theory in a high-level sensor data fusion architecture. In *2011 IEEE Intelligent Vehicles Symposium (IV)*. 770–775. doi:10.1109/IVS.2011.5940430

[2] Mohammed Alhamadi, Omar Alghamdi, Sarah Clinch, and Markel Vigo. 2022. Data Quality, Mismatched Expectations, and Moving Requirements: The Challenges of User-Centred Dashboard Design. In *Nordic Human-Computer Interaction Conference* (Aarhus, Denmark) *(NordiCHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 11, 14 pages. doi:10.1145/3546155.3546708

[3] Ioannis Arapakis, Xiao Bai, and B. Barla Cambazoglu. 2014. Impact of response latency on user behavior in web search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Gold Coast, Queensland, Australia) *(SIGIR '14)*. Association for Computing Machinery, New York, NY, USA, 103–112. doi:10.1145/2600428.2609627

[4] Ori Bar El, Tova Milo, and Amit Somech. 2019. ATENA: An Autonomous System for Data Exploration Based on Deep Reinforcement Learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, Beijing China, 2873–2876. doi:10.1145/3357384.3357845

[5] Ori Bar El, Tova Milo, and Amit Somech. 2020. Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, Portland OR USA, 1527–1537. doi:10.1145/3318464.3389779

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). Association for Computational Linguistics, Doha, Qatar, 1724–1734. doi:10.3115/v1/D14-1179

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv* abs/1412.3555 (2014).

[8] A. P. Dempster. 2018. A Generalization of Bayesian Inference. *Journal of the Royal Statistical Society: Series B (Methodological)* 30, 2 (Dec. 2018), 205–232. doi:10.1111/j.2517-6161.1968.tb00722.x _eprint: https://academic.oup.com/jrsssb/article-pdf/30/2/205/49095334/jrsssb_30_2_205.pdf.

[9] Lucas Deregnaucourt, Alexis Lechervy, Hind Laghmara, and Samia Ainouz. 2023. An Evidential Deep Network Based on Dempster-Shafer Theory for Large Dataset. In *Advances and Applications of DSmT for Information Fusion: Collected Works(Volume 5)*, Florentin Smarandache, Jean Dezert, and Albena Tchamova (Eds.). 907–914. https://normandie-univ.hal.science/hal-04448387

[10] Daniel Deutch, Amir Gilad, Tova Milo, Amit Mualem, and Amit Somech. 2022. FEDEX: An Explainability Framework for Data Exploration Steps. *Proceedings of the VLDB Endowment* 15, 13 (Sept. 2022), 3854–3868. doi:10.14778/3565838.3565841

[11] Yanlei Diao. 2015. Explore-By-Example: A New Database Service for Interactive Data Exploration. In *Proceedings of the Second International Workshop on Exploratory Search in Databases and the Web*. ACM, Melbourne VIC Australia, 1–1. doi:10.1145/2795218.2795226

[12] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. *IEEE Transactions on Knowledge and Data Engineering* 28, 11 (Nov. 2016), 2842–2856. doi:10.1109/TKDE.2016.2599168

[13] Marina Drosou and Evaggelia Pitoura. 2013. YmalDB: exploring relational databases via result-driven recommendations. *The VLDB Journal* 22, 6 (Dec. 2013), 849–874. doi:10.1007/s00778-013-0311-4

[14] Magdalini Eirinaki and Sweta Patel. 2015. QueRIE reloaded: Using matrix factorization to improve database query recommendations. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, Santa Clara, CA, USA, 1500–1508. doi:10.1109/BigData.2015.7363913

[15] Edwin D. El-Mahassni and Karen L. White. 2015. A Discussion of Dempster-Shafer Theory and its Application to Identification Fusion. https://api.semanticscholar.org/CorpusID:119579601

[16] GitHub. [n. d.]. TAU-DB/REACT-IDA-Recommendation-benchmark. https://github.com/TAU-DB/REACT-IDA-Recommendation-benchmark. [Accessed 23-05-2025].

[17] Apostolos Glenis and Georgia Koutrika. 2021. PyExplore: Query Recommendations for Data Exploration without Query Logs. In *Proceedings of the 2021 International Conference on Management of Data*. ACM, Virtual Event China, 2731–2735. doi:10.1145/3448016.3452762

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 6645–6649. doi:10.1109/ICASSP.2013.6638947

[19] Muhammad Hafeez. 2011. *Application of Dempster Shafer Theory to Assess the Status of Sealed Fire in a Cole Mine.* https://urn.kb.se/resolve?urn=urn:nbn:se:bth-5323

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735 _eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

[21] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJlWWJSFDH

[22] Enhui Huang, Yanlei Diao, Anna Liu, Liping Peng, and Luciano Di Palma. 2023. Efficient and robust active learning methods for interactive database exploration. *The VLDB Journal* (Nov. 2023). doi:10.1007/s00778-023-00816-x

[23] Jiulun Fan, Ju Fan, Guoliang Li, Guoliang Li, Lizhu Zhou, and Lizhu Zhou. 2011. Interactive SQL query suggestion: Making databases user-friendly. *IEEE International Conference on Data Engineering* (April 2011), 351–362. doi:10.1109/icde.2011.5767843 MAG ID: 2108932068 S2ID: abc44137f902af786bb413b24ccb7192f7370a25.

[24] Eugenie Y Lai, Zainab Zolaktaf, Mostafa Milani, Omar AlOmeir, Jianhao Cao, and Rachel Pottinger. 2023. Workload-Aware Query Recommendation Using Deep Learning. (2023).

[25] Tavor Lipman, Tova Milo, and Amit Somech. 2023. ATENA-PRO: Generating Personalized Exploration Notebooks with Constrained Reinforcement Learning. In *Companion of the 2023 International Conference on Management of Data*. ACM, Seattle WA USA, 167–170. doi:10.1145/3555041.3589727

[26] Pingchuan Ma, Rui Ding, Shi Han, and Dongmei Zhang. 2021. MetaInsight: Automatic Discovery of Structured Knowledge for Exploratory Data Analysis. In *Proceedings of the 2021 International Conference on Management of Data*. ACM, Virtual Event China, 1262–1274. doi:10.1145/3448016.3457267

[27] Antonis Mandamadiotis, Georgia Koutrika, and Sihem Amer-Yahia. 2024. Guided SQL-Based Data Exploration with User Feedback. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 4884–4896. doi:10.1109/ICDE60146.2024.00372 ISSN: 2375-026X.

[28] Tova Milo and Amit Somech. 2018. Next-Step Suggestions for Modern Interactive Data Analysis Platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, London United Kingdom, 576–585. doi:10.1145/3219819.3219848

[29] Tova Milo and Amit Somech. 2020. Automating Exploratory Data Analysis via Machine Learning: An Overview. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, Portland OR USA, 2617–2622. doi:10.1145/3318464.3383126

[30] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. 2021. DORA THE EXPLORER: Exploring Very Large Data With Interactive Deep Reinforcement Learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, Virtual Event Queensland Australia, 4769–4773. doi:10.1145/3459637.3481967

[31] Gregory M. Provan. 1990. The Application of Dempster Shafer Theory to a Logic-Based Visual Recognition System. In *Uncertainty in Artificial Intelligence*, Max HENRION, Ross D. SHACHTER, Laveen N. KANAL, and John F. LEMMER (Eds.). Machine Intelligence and Pattern Recognition, Vol. 10. North-Holland, 389–405. doi:10.1016/B978-0-444-88738-2.50037-3 ISSN: 0923-0459.

[32] Björn Scheuermann and Bodo Rosenhahn. 2011. Feature Quarrels: The Dempster-Shafer Evidence Theory for Image Segmentation Using a Variational Framework. In *Computer Vision – ACCV 2010*, Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 426–439.

[33] Scott J Seims. 2009. *A Study of Dempster-Shafer's Theory of Evidence in Comparison to Classical Probability Combination.* Ph. D. Dissertation. California Polytechnic State University, San Luis Obispo, California. doi:10.15368/theses.2009.104

[34] Kari Sentz and Scott Ferson. 2002. *Combination of Evidence in Dempster-Shafer Theory.* Technical Report SAND2002-0835. Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US). doi:10.2172/800792

[35] Glenn Shafer. 2016. Dempster's rule of combination. *International Journal of Approximate Reasoning* 79 (2016), 26–40. doi:10.1016/j.ijar.2015.12.009

[36] Amit Somech, Tova Milo, and Chai Ozeri. 2019. Predicting "What is Interesting" by Mining Interactive-Data-Analysis Session Logs. doi:10.5441/002/EDBT.2019.42

[37] L. Spitzner. 2003. The Honeynet Project: trapping the hackers. *IEEE Security Privacy* 1, 2 (2003), 15–23. doi:10.1109/MSECP.2003.1193207

[38] Shirin Tahmasebi, Amir H. Payberah, Ahmet Soylu, Dumitru Roman, and Mihhail Matskin. 2023. TRANSQLATION: TRANSformer-based SQL RecommendATION. In *2023 IEEE International Conference on Big Data (BigData)*. 4703–4711. doi:10.1109/BigData59044.2023.10386277

[39] Xiaole Wang and Jiwei Qin. 2024. Multimodal recommendation algorithm based on Dempster-Shafer evidence theory. *Multimedia Tools and Applications* 83, 10 (March 2024), 28689–28704. doi:10.1007/s11042-023-15262-8

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[41] Roland R. Yager and Liping Liu (Eds.). 2008. *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Studies in Fuzziness and Soft Computing, Vol. 219. Springer, Berlin, Heidelberg. doi:10.1007/978-3-540-44792-4

[42] Vahid Yaghoubi, Liangliang Cheng, Wim Van Paepegem, and Mathias Kersemans. 2022. A novel multi-classifier information fusion based on Dempster–Shafer theory: application to vibration-based fault detection. *Structural Health Monitoring* 21, 2 (March 2022), 596–612. doi:10.1177/14759217211007130 Publisher: SAGE Publications.

[43] Farzaneh Zirak, Farhana Murtaza Choudhury, and Renata Borovica-Gajic. 2024. SeLeP: Learning Based Semantic Prefetching for Exploratory Database Workloads. *Proc. VLDB Endow.* 17, 8 (2024), 2064–2076. doi:10.14778/3659437.3659458