Fishing Fort: A System for Graph Analytics with ML Prediction and Logic Deduction

Wenfei Fan
Shenzhen Institute of Computing Sciences, China
University of Edinburgh, UK
Beihang University, Beijing, China

Shuhao Liu
Shenzhen Institute of Computing Sciences, China

Abstract

This paper reports Fishing Fort, a graph analytic system developed in response to the following questions. What practical value can we get out of graph analytics? How can we effectively deduce the value from a real-life graph? Where can we get clean graphs to make accurate analyses possible? To answer these questions, Fishing Fort advocates to unify logic deduction and ML prediction by proposing Graph Association Rules (GARs), a class of logic rules in which ML models can be embedded as predicates. It employs GARs to deduce graph associations, enrich graphs and clean graphs. It has been deployed in production lines and proven effective in online recommendation, drug discovery, credit risk assessment, battery manufacturing and cybersecurity, among other things.

2012 ACM Subject Classification Information systems → Data mining; Information systems → Data management systems; Information systems → Information integration

Keywords and phrases graph analytics, data cleaning, association analysis

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.6

Acknowledgements The paper is a tribute to Professor Val Tannen. Val was a professor at UPenn when Fan was doing PhD there, and has been providing Fan with unfailing support ever since.

1 Introduction

When working with practitioners in industry, we often hear the following questions.

(1) What practical value can we get out of big data analytics? After a decade of study, people are still scrambling to find killer apps of big data analytics. While the success of machine learning (ML) should be at least partly attributed to big data, people have higher expectations from big data analytics to help them solve problems at hand, e.g., optimize a costly step in a manufacturing process or find the use of an old drug for a new disease. Besides, big data analytics gets credits for making an advance only if it is not entirely ML-based, i.e., it approaches the problem not by simply training and applying an ML model.

(2) How should we analyze big data, ML prediction or logic deduction? Data analytics is often approached by employing either ML models or logic rules. However, neither of the two is superior to the other. On the one hand, it is hard to discover a small number of accurate logic rules to cover different cases of our application and data. On the other hand, ML predictions are probabilistic and hard to explain; practitioners may not want to make critical decisions based on ML predictions alone. Is it possible to unify the two and benefit from both?

(3) Where can we get clean data for analytics? Real-life data is often dirty, as evidenced by duplicates, semantic inconsistencies, stale data and missing values commonly found in practice. Dirty data has been a longstanding challenge. To make practical use of big data analytics, it is a must to clean the data. Indeed, data-driven decisions based on dirty data can be worse than making decisions with no data. For instance, "as a healthcare, retail, or financial services business you cannot afford to make decisions based on yesterday's data" [9].

In an effort to tackle these questions, we have developed Fishing Fort, a system for deducing associations among entities in graphs. Fishing Fort has been deployed in the production lines of several domains, and has proven effective there. The system is named in memory of the siege of Diaoyu (Fishing) Fortress (a.k.a. Diaoyu Castle), a battle during which the fourth khagan of the Mongol Empire was severely wounded to death by the defenders, and a battle that changed the history of China and the world. The system has several unique features.

(1) Unification of ML and logic. Fishing Fort unifies ML prediction and logic deduction by employing a class of Graph Association Rules (GARs) [17]. A GAR is composed of a graph pattern to identify related entities, and a dependency on the entities to disclose their association, correlation and interaction. Taken together, the pattern and dependency reveal regularities among entities in graphs. Moreover, a GAR may embed ML models as predicates. As a consequence, GARs may embed ML predictions in logic deduction, and moreover, explain local predictions of ML models such as graph neural networks (GNNs) [26, 27, 4].

(2) Accurate association deduction. Fishing Fort deduces associations with GARs. It supports algorithms for discovering GARs from real-life graphs [13], conducting (batch and incremental) deduction [17] and making event predictions [18]. It has found successful applications in early drug discovery [11], lithium-ion battery manufacturing, risk control for bank loans, cyber attack detection and online recommendation in e-commerce [10], among other things.

(3) Data enrichment and cleaning. Fishing Fort can also be used to improve the quality of real-life graphs. Given a graph G, it enriches G by extracting data that pertains to the entities of G from external data sources, e.g., knowledge graphs [19]. It detects and fixes errors in the (enriched) graph by employing a special form of GARs [12], in polynomial time (PTIME). The fixes computed are logical consequences of the rules and accumulated ground truth (i.e., validated facts); hence if the rules and ground truth are correct, so are the fixes.

Organization. The remainder of the paper is organized as follows. Section 2 defines GARs. Section 3 presents the architecture of Fishing Fort and showcases how it deduces associations with GARs in real-life applications. Section 4 shows how Fishing Fort enriches and cleans graphs with (a special form) of GARs. Finally, Section 5 discusses future plans.

2 Unifying ML prediction and Logic Deduction

This section reviews basic notations (Section 2.1), and defines Graph Association Rules (GARs) (Section 2.2). It also reports the complexity for reasoning about GARs (Section 2.3).

2.1 Preliminaries

Assume three countably infinite sets of symbols, denoted by Ω, Υ and U, for labels, attributes and constants, respectively. We consider *graphs* $G = (V, E, L, F_A)$, where (a) V is a finite set of vertices, (b) $E \subseteq V \times \Omega \times V$ is a finite set of edges, where each $e = (v, l, v')$ in E denotes an edge from vertex v to v' that is labeled with $l \in \Omega$, (c) L is a function such that for each vertex $v \in V$, $L(v)$ is a label from Ω, and (d) each vertex $v \in V$ carries a tuple $F_A(v) = (A_1 = a_1, \ldots, A_n = a_n)$ of *attributes* of a finite arity, where $A_i \in \Upsilon$ and $a_i \in U$, written as $v.A_i = a_i$, and $A_i \neq A_j$ if $i \neq j$, representing properties. Different vertices may carry different attributes, which are not constrained by a schema.

A *graph pattern* is $Q[\bar{x}] = (V_Q, E_Q, L_Q, \mu)$, where (1) V_Q (resp. E_Q) is a finite set of *pattern nodes* (resp. *edges*), (2) L_Q assigns a label $L_Q(u) \in \Gamma$ to each pattern node $u \in V_Q$, (3) $\bar{x}$ is a list of distinct variables; and μ is a mapping that assigns a distinct variable to each node v in V_Q. For $x \in \bar{x}$, we use $\mu(x)$ and x interchangeably when it is clear in the context.

A *match* of pattern $Q[\bar{x}]$ in graph G is defined as a homomorphism h from Q to G such that (a) for each node $u \in V_Q$, $L_Q(u) = L(h(u))$; and (b) for each pattern edge $e = (u, l, u')$ in Q, $e' = (h(u), l, h(u'))$ is an edge in G. We denote the match as a vector $h(\bar{x})$, consisting of $h(x)$ for all $x \in \bar{x}$ in the same order as $\bar{x}$, denoting entities identified by Q.

2.2 Graph Association Rules

GARs are defined with predicates. A *predicate* of pattern $Q[\bar{x}]$ is one of the following:

$$p ::= l(x, y) \mid x.A \otimes y.B \mid x.A \otimes c \mid \mathcal{M}(x.\bar{A}, y.\bar{B}),$$

where $\otimes$ is one of $=, \neq, <, \leq, >, \geq$; $l \in \Omega$; x and y are variables in $\bar{x}$; c is a constant in U; A and B are attributes; and $x.\bar{A}$ is a list of attributes at "vertex" x; similarly for $y.\bar{B}$.

The predicates are classified as follows. (1) Logic predicates: *link predicate* $l(x, y)$ indicates the existence of an edge labeled l from vertex x to y; *variable predicate* $x.A \otimes y.B$ and *constant predicate* $x.A \otimes c$ check the correlation and interaction of attribute values. (2) ML predicates: $\mathcal{M}(x.\bar{A}, y.\bar{B})$ is a pre-trained ML model that returns true iff $\mathcal{M}$ predicts true at $(x.\bar{A}, y.\bar{B})$. Here $\mathcal{M}$ can be any ML model that returns Boolean (e.g., $\mathcal{M} \geq \sigma$ for a predefined bound σ).

Rules. A GAR (*Graph Association Rule*) is a pair of a graph pattern and a dependency [17]:

$$\varphi = Q[\bar{x}](X \to p_0),$$

where $Q[\bar{x}]$ is a graph pattern, X is a conjunction of predicates of $Q[\bar{x}]$, and p_0 is a single predicate of $Q[\bar{x}]$. We refer to $Q[\bar{x}]$ and $X \to p_0$ as the *pattern* and *dependency* of GAR φ, respectively, and to X and p_0 as the *precondition* and *consequence* of φ, respectively.

Intuitively, Q in a GAR identifies entities in a graph, and $X \to p_0$ is applied to the entities. Predicates $x.A \otimes c$ and $x.A \otimes y.B$ specify *value associations of* attributes. Link predicates enforce edge existence to deduce missing links. Moreover, one can "plug in" pre-trained ML models $\mathcal{M}$ for entity resolution [33], link predictions [44] and similarity checking [8]. For each application domain, Fishing Fort maintains a library of pre-trained ML models.

We can uniformly express various ML models we used as link prediction for $l(x, y)$, where the label l can indicate (1) a predicted link, (2) the match of x and y as the same entity, linked by a dummy edge with "=" as l, or (3) semantic similarity between vertices x and y linked by an edge with "$\approx$" as l, indicating that x and y are "semantically" close.

We will showcase GARs for real-life applications in Section 3.

As shown in [17], graph functional dependencies (GFDs) [23], graph entity dependencies (GEDs) [20] and graph pattern association rules (GPARs) [22] are special cases of GARs. GARs extend these graph dependencies by supporting ML and link predicates. Note that link predicates mutate the topological structure of a graph. Besides the primitive predicates above, Fishing Fort also supports, e.g., local 2-dimensional Weisfeiler-Leman (local 2-WL) test [27], to explain GNN-based recommendations and link predictions. It is known that such GNN models are no more expressive than local 2-WL test [27].

Semantics. Consider a GAR $\varphi = Q[\bar{x}](X \to p_0)$. Denote by $h(\bar{x})$ a match of pattern Q in a graph G, and by p a predicate of $Q[\bar{x}]$. We write $h(\mu(x))$ as $h(x)$, where μ is the mapping in Q from $\bar{x}$ to vertices in Q. A match $h(\bar{x})$ *satisfies* a predicate p, denoted by $h(\bar{x}) \models p$, if one of following conditions is satisfied: (a) when p is $l(x, y)$, there exists an edge with label l from $h(x)$ to $h(y)$; (b) when p is $x.A \otimes y.B$, the vertex $h(x)$ (resp. $h(y)$) carries attribute A (resp. B), and $h(x).A \otimes h(y).B$; similarly for constant predicate $h(x).A \otimes c$; and (c) when p is $\mathcal{M}(x.\bar{A}, y.\bar{B})$, the ML model $\mathcal{M}$ predicts true at $(h(x).\bar{A}, h(y).\bar{B})$.

We write $h(\bar{x}) \models X$ if $h(\bar{x}) \models p$ for *all* p in a set X of predicates. We write $h(\bar{x}) \models X \to p_0$ if whenever $h(\bar{x}) \models X$, then $h(\bar{x}) \models p_0$. We say that a graph G *satisfies* GAR $\varphi = Q[\bar{x}](X \to p_0)$, denoted by $G \models \varphi$, if for all matches $h(\bar{x})$ of $Q[\bar{x}]$ in G, $h(\bar{x}) \models X \to p_0$. We say that G *satisfies* a set Σ of GARs, denoted by $G \models \Sigma$, if for all GARs $\varphi \in \Sigma$, $G \models \varphi$.

For $p = x.A \otimes c$, we use $h(p)$ to denote $h(x).A \otimes c$; similarly for the other predicates. For a set X of predicates, we denote by $h(X)$ the collection of $h(p)$ for all p in X. For a set Γ of validated facts of the forms $u.A \otimes v.B$ and $u.A \otimes c$ for vertices u, v in a graph G, we say that $h(p)$ is *validated* with Γ if it is enclosed in Γ; in particular, for $p = \mathcal{M}(x.\bar{A}, y.\bar{B})$, the values of $h(x).A$ and $h(y).B$ are enclosed in Γ for all corresponding $A \in \bar{A}$ and $B \in \bar{B}$; similarly for $h(X)$. As will be seen in Section 3.1, we check validated facts to deduce reliable associations.

2.3 Complexity

There are three classical problems for dependencies, stated as follows.

The *satisfiability* problem is to decide, given a set Σ of GARs, whether there exists a graph G such that $G \models \Sigma$ and for each GAR $Q[\bar{x}](X \to p_0) \in \Sigma$, Q has a match in G? Intuitively, this is to ensure that all GARs can be applied to G at the same time without conflicts.

A set Σ of GARs *implies* a GAR φ, denoted by $\Sigma \models \varphi$, if for all graphs G, if $G \models \Sigma$ then $G \models \varphi$, i.e., φ is a logical consequence of Σ. The *implication problem* is to decide, given a set Σ of GARs and a GAR φ, whether $\Sigma \models \varphi$? Intuitively, the implication analysis can help us remove redundant GARs in rule discovery and speed up association analyses with GARs.

The *validation problem* is to decide, given a graph G and a set Σ of GARs, whether $G \models \Sigma$? Intuitively, this is to settle the complexity of association analyses with GARs.

To simplify the implication analysis, we consider ML predicates of the form $\mathcal{M}(x.\bar{A}, y.\bar{B}) \geq \sigma$ for a predefined σ. Moreover, we assume that the ML models in a set of GARs are independent, i.e., they are trained for different purposes and do not overlap/entail each other, e.g., models for predicting the anomaly and capacity of battery cells in battery manufacturing.

It has been shown that the satisfiability, implication and validation are coNP-complete, NP-complete, and coNP-complete for GARs, respectively [17, 20]. Here we assume that given two lists of attributes $x.\bar{A}$ and $y.\bar{B}$, checking $\mathcal{M}(x.\bar{A}, y.\bar{B})$ is in PTIME in the sizes $|x.\bar{A}|$ and $|y.\bar{B}|$, as commonly found in practice for pre-trained $\mathcal{M}$. The complexity bounds are the same as for reasoning about GEDs [20]. Moreover, the implication and satisfiability analyses are no harder than their counterparts for relational conditional functional dependencies [14].

3 Deducing Graph Associations

This section first presents the architecture and workflow of Fishing Fort (Section 3.1), and then showcases its applications in different domains (Section 3.2).

3.1 The Architecture of Fishing Fort

As shown in Figure 1, Fishing Fort supports both association deduction, and graph enrichment and cleaning. For each application domain, it maintains a set Γ of ground truth, and references relevant external knowledge graphs KG. Given a real-life graph G, it first enriches G by extracting relevant data from KG and cleans the enriched G; to simplify the discussion, we refer to the enriched and cleaned graph also as G. After the preprocessing step, Fishing Fort then discovers a set Σ of GARs for the application, and deduces associations from G with the GARs of Σ; in the process it accumulates ground truth for subsequent analyses. Below we focus on association deduction, and defer data enrichment and cleaning to Section 4.

Association deduction is carried out by the following main modules and algorithms.

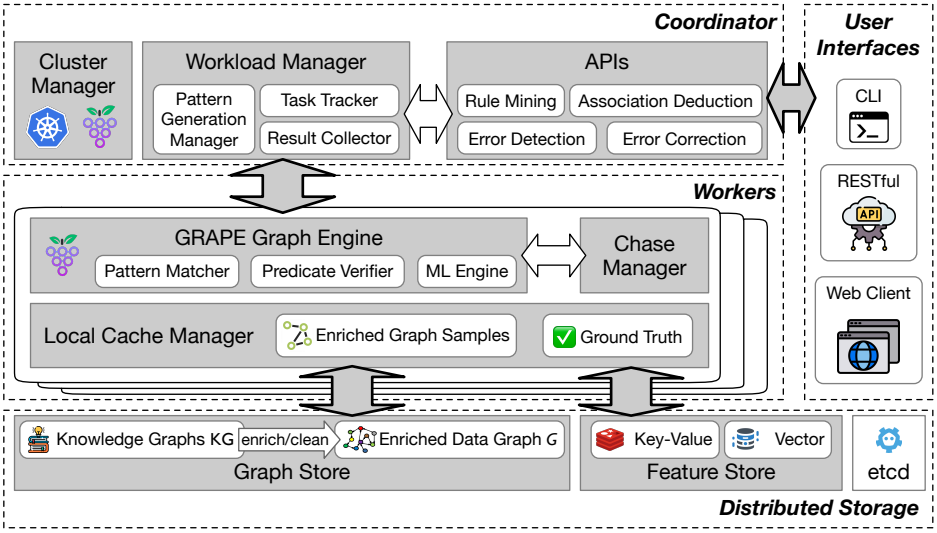

Figure 1 The architecture of Fishing Fort.

Rule discovery. Fishing Fort implements the parallel algorithm of [13] for discovering GARs. To scale with a large graph G, it employs three strategies. (1) Application-driven discovery. Given an application $\mathcal{A}$, Fishing Fort trains an ML model $\mathcal{M}_{\mathcal{A}}$ to identify vertices, edges and properties in G that pertain to $\mathcal{A}$. It reduces G to a smaller graph $G_{\mathcal{A}}$ that consists only of the data pertaining to $\mathcal{A}$. Moreover, it discovers $\mathcal{A}$-*relevant* rules, i.e., GARs with consequence predicates selected by $\mathcal{M}_{\mathcal{A}}$. (2) Sampling. It samples a set H of graphs from $G_{\mathcal{A}}$ such that each graph is at most $\rho\%$ of $G_{\mathcal{A}}$ and consists of representative entities in $G_{\mathcal{A}}$ and their surrounding subgraphs, for a predefined bound $\rho\%$. Denote by Σ_G and Σ_H the set of $\mathcal{A}$-relevant rules discovered from G and H, respectively. Fishing Fort guarantees that given bounds σ and $\gamma\%$, it samples H such that (a) at least $\gamma\%$ of rules in Σ_G are covered by Σ_H, and (b) each of these rules can be applied at least σ times on the entire G, i.e., the rules in Σ_H have recall above $\gamma\%$ and support above σ over the original graph G. (3) Parallel scalability. Fishing Fort employs data-partitioned parallelism. It partitions G and distributes the fragments across different machines. It works on the fragments in parallel and guarantees to reduce runtime when more machines are used, i.e., the parallel scalability [29]. Hence in principle, it can scale with large graphs G by using more machines when needed.

Association deduction. Fishing Fort deduces link and value associations from graph G, by chasing G with the set $\Sigma = \Sigma_H$ of GARs discovered. More specifically, it applies a GAR $Q[\bar{x}](X \to p_0)$ to G only if there exists a match h of Q in G such that $h \models X$ and the $h(X)$ is validated with the ground truth in Γ (see Section 2.2); if so, it deduces association $h(p_0)$ and adds $h(p_0)$ to Γ as validated facts for subsequent deduction. To carry this out, it implements a combination of the parallel algorithms of [17, 21], and supports two modes: (1) batch mode, to deduce all associations from G at once, and (2) incremental mode, to deduce changes online to associations in response to updates to G. In both modes, users may also opt to pick entities of their interest and deduce associations pertaining to these entities only. Moreover, it explains ML predictions if requested by users (see Section 3.2.1).

Fishing Fort provides the following performance guarantees. (a) The chase is Church-Rosser [1], i.e., it guarantees to terminate and converge at the same result no matter what GARs in Σ are used and in what order the rules are applied. (b) The deduced associations are logical consequences of the GARs of Σ and ground truth of Γ; hence if Σ and Γ are correct, so are the deduced associations. (c) Its deduction algorithm is also parallelly scalable.

Implementation. Fishing Fort is implemented on top of GRAPE [24], a graph engine that parallelizes single-machine graph algorithms based on a fixpoint model in terms of partial evaluation and incremental computation. GRAPE was acquired by Alibaba Group and renamed as GraphScope [16]; it supports 90+% of daily graph computations of Alibaba.

3.2 Fishing Fort in Action

We next present example applications of Fishing Fort, along with GARs discovered from real-life data. The graph patterns of the GARs are depicted in Figure 2.

3.2.1 Online Recommendation

ML models have been widely used in e-commerce to recommend items to users. The models are often classified as collaborative filtering (CF) and content-based (CB). CF identifies user preference and makes recommendation by learning from user-item historical interactions, e.g., users' previous ratings and browsing history, and CB primarily compares the contents of users and items such as user profiles and item features. However, a single strategy, either CF or CB, often does not suffice in practice. For example, instead of exploring new interesting items, CF tends to find similar ones w.r.t. the user's past interaction due to its collaborative nature. It does not work well when the interaction data is sparse and when a recommender system starts cold. To rectify these limitations, hybrid models have been explored to unify interaction-level similarity and content-level similarity. However, the hybrid approach often requires training a new ML model starting from scratch. Moreover, the ML models usually compute a prediction score, i.e., recommendation strength, between each user x and item y, and recommend y to x only if the score is above some predetermined thresholds. As a result, these ML models often make false positive (FP) or false negative (FN) predictions, especially in the "fuzzy area" where the predicted scores are near the thresholds, i.e., an area where the prediction strengths are neither sufficiently large nor sufficiently small.

Another issue concerns the explanation of ML predictions $\mathcal{M}(x, y)$, to tell why an item y is recommended to user x. The need for this is twofold, (a) to provide the users with insights and establish their trust in the predictions, and (b) help developers debug ML models by revealing errors or bias in training data that result in adverse and unexpected behaviors.

Fishing Fort has been deployed at e-commerce companies to reduce FPs and FNs of ML recommendation models, and provide local explanations of GNN-based recommendations. It improves the average click-through rate of the companies by up to 50×.

(a) Reducing FPs and FNs of ML predictions. Consider an ML model $\mathcal{M}_{\mathsf{rate}}(x, y)$ for movie recommendations that, given a movie y and a user x, outputs a numeric score, i.e., recommendation strength, between $(0, 1)$. The higher the score, the stronger the confidence. Take 0.5 as the strength threshold, i.e., $\mathcal{M}_{\mathsf{rate}}$ recommends y to x if $\mathcal{M}_{\mathsf{rate}}(x, y) \geq 0.5$.

As an example, GAR $\varphi_1 = Q_1[\bar{x}](X_1 \wedge 0.5 \leq \mathcal{M}_{\mathsf{rate}}(x_0, x_1) \leq 0.6 \rightarrow \mathsf{dislike}(x_0, x_1))$ corrects some FPs of ML model $\mathcal{M}_{\mathsf{rate}}$ in its "fuzzy area" of recommendations. Together with Q_1, precondition X_1 specifies the following: (1) user x_0 disliked a high-rating movie x_2, which stars the same actor x_4 as movie x_1 does; and (2) x_0 also disliked another high-rating movie x_3, which is directed by the director x_5 of movie x_1. Intuitively, model $\mathcal{M}_{\mathsf{rate}}$ (especially if it is CF-based) is inclined to make the recommendation of movie x_1 to user x_0, since many other users could have positive interactions with movies x_1, x_2 and x_3 (hence the high ratings). However, GAR φ_1 overrides the prediction with logic conditions X_1, because unlike an average user, user x_0 seems to approve neither the leading actor nor the director of movie x_1. Among the matching cases of Q_1 in real-world datasets, this rule reduces FPs by 92.3%.

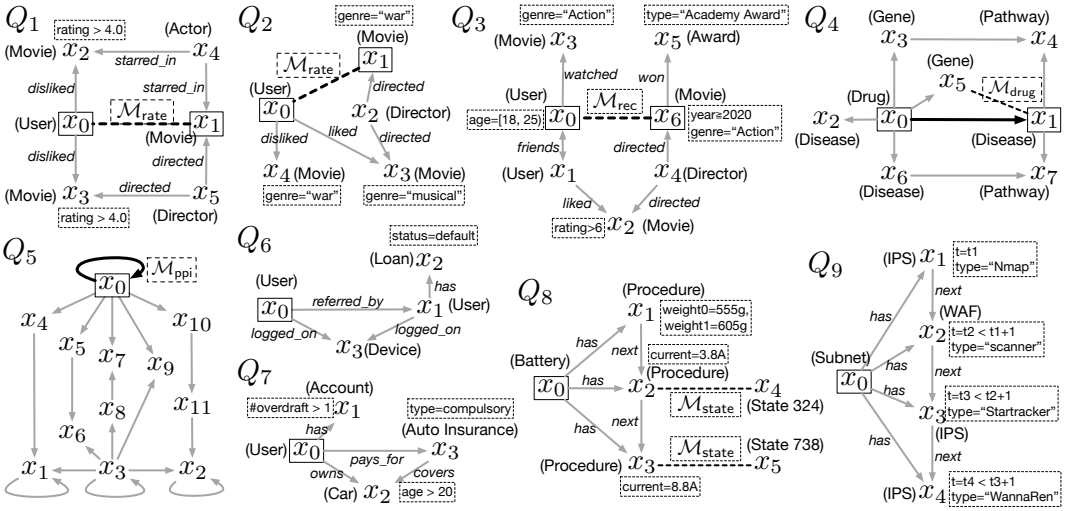

Figure 2 Graph patterns in real-life GARs.

In contrast, GAR $\varphi_2 = Q_2[\bar{x}](X_2 \wedge 0.4 \leq \mathcal{M}_{\text{rate}}(x_0, x_1) \leq 0.5 \rightarrow \text{like}(x_0, x_1))$ suggests recommending a war movie x_1 to user x_0 regardless of the negative ruling of $\mathcal{M}_{\text{rate}}(x_0, x_1)$, if the following conditions (Q_2 and X_2) are met: (1) x_0 disliked a war movie x_4, yet (2) x_0 liked a musical x_3 directed by x_2, the director of movie x_1. Model $\mathcal{M}_{\text{rate}}$ may infer that x_0 is not a fan of war movies based on the interaction with x_4. Nonetheless, rule φ_2 rejects the prediction, based on the evidence that x_0 is likely a fan of director x_2. Over verification datasets, GAR φ_2 reduces FNs of $\mathcal{M}_{\text{rate}}$ by 87% for the matching cases of "fuzzy" predictions.

(b) Explaining ML predictions. Consider $\varphi_3 = Q_3[\bar{x}](X_3 \rightarrow \mathcal{M}_{\text{rec}}(x_0, x_6))$, where $\mathcal{M}_{\text{rec}}$ is a GNN-based model, $\mathcal{M}_{\text{rec}}(x_0, x_6)$ indicates that the model recommends movie x_6 to user x_0, and X_3 is $x_0.\text{age} \geq 18 \wedge x_0.\text{age} < 25 \wedge x_2.\text{rating} \geq 6 \wedge x_3.\text{genre} =$ "Action" $\wedge x_6.\text{genre} =$ "Action" $\wedge x_6.\text{year} \geq 2020 \wedge x_5.\text{type} =$ "Academy Award". As opposed to the previous rules, the consequence predicate of this GAR is an ML predicate. Fishing Fort discovers pattern Q_3 and precondition X_3 to explain $\mathcal{M}_{\text{rec}}$ predictions from the movie-watching history and friendship of x_0, and highlights important features of x_0 and x_6.

As a concrete example, suppose that the GNN model $\mathcal{M}_{\text{rec}}$ recommends a movie y_0 "Everything Everywhere All at Once" to a user x_0 "Mike". Then φ_3 may provide high-level rationale behind the recommendation as follows: the movie is recommended to Mike because Mike is a young adult and has watched action movies before, the movie won the 95th academy award for best picture, and it is directed by "Daniel Scheinert", the director of a favorite movie of a friend Peter of Mike. This provides a *local explanation* of the prediction.

Unlike previous methods for explaining ML predictions, e.g., SubgraphX [47] and GNNExplainer [46], Fishing Fort not only provides a subgraph as an explanation, but also tells us what features are decisive for an ML model to make predictions and under what conditions the predictions can be made. Moreover, Fishing Fort supports a predicate of local 2-WL test, and therefore, is able to explain common GNN-based recommendations, which is no more expressive than local 2-WL test [27]. In this way, Fishing Fort deduces local explanations for individual instances with GARs. Moreover, the rules for a GNN-based recommendation model $\mathcal{M}$ disclose the general behaviors of $\mathcal{M}$ and can serve as global explanations of $\mathcal{M}$.

3.2.2 Early Drug Discovery

Drug discovery is the process of new drug identification, which starts from target selection and validation, through preclinical screening, to clinical trials [25]. It is time-consuming and quite expensive. The development of a new drug takes 10–15 years, costs around 2 billion US dollars, and typically has a high risk of failure (>90%) [50].

To accelerate drug discovery, reduce the cost, and increase the success rate, ML models have been explored to identify drug-disease associations (DDAs), drug-drug interactions (DDIs), and protein-protein interactions (PPIs). However, it is costly to train deep-learning models for accurate predictions, and the predictions often have false positives and false negatives. Moreover, ML predictions "still lack reliable explanations that are crucial to drug repurposing and ADR (adverse drug reaction)" [50]. Worse still, noise is often introduced by "unavoidable inaccuracy" of the ML models, e.g., nonexistent relations and inaccurately named entities. Add to the complication that the models are often trained on possibly dirty data, especially when the data comes from multiple sources.

Fishing Fort has been adopted in the early stage of drug discovery. It is used to assist (1) target identification, to identify the right biological molecules or cellular pathways that can be modulated by drugs to achieve therapeutic benefits, where PPIs are often crucial, (b) drug repurposing, to reuse existing drugs for a new disease, and (c) ADR, to disclose undesirable impacts that do not meet the anticipated therapeutic effects and may cause injuries to patients. The need for these is evident. For example, target identification is perhaps the most crucial initial step in drug discovery, and influences the chances of success at every step of drug development. Traditional target identification usually starts in an academic setting, and takes from years to decades. Fishing Fort helps select candidate targets and speed up the process. It identifies DDAs, DDIs and PPIs simply as missing links. Moreover, it has been used to integrate and clean biomedical libraries and data banks (see Section 4).

As an example, Fishing Fort was used to discover GARs for repositioning of existing drugs that may have therapeutic effect on a particular type of Parkinson disease. A GAR is $\varphi_4 = Q_4[\bar{x}](X_4 \rightarrow l(x_0, x_1))$. Together with Q_4, precondition X_1 specifies the following: (1) drug x_0 has a known effect on an inborn genetic blood disease x_2; (2) disease x_1 is the Parkinson; (3) drug x_0 interacts with a gene x_3, which shares an effect pathway x_4 with x_1; (4) drug x_0 can interact with a gene x_5, which has an $\mathcal{M}_{\mathsf{drug}}$-predicted relationship with x_1 (the dashed arrow in Q_4), where $\mathcal{M}_{\mathsf{drug}}$ is a pre-trained ML model that predicts the associations between genes and diseases [40, 34, 44]; and (5) drug x_0 has a known effect on a type of skin cancer x_6, which shares an effect pathway with x_1. The predicted link $l(x_0, x_1)$ (the bold line in Q_4) indicates that drug x_0 may have impact on Parkinson x_1 in some way.

Such GARs suggest five drugs that may have a hidden association with the particular type of Parkinson's disease, including Colforsin (Forskolin) [53], Sulindac [36, 6], Tamoxifen [32], and Tretinoin [43]. Our partners in pharmacy have verified four predictions with published evidence. The remaining one is undergoing their active lab investigation.

As another example, Fishing Fort discovered GARs for PPIs. Such a GAR is $\varphi_5 = Q_5[\bar{x}](\mathcal{M}_{\mathsf{ppi}}(x_0, x_0) \geq \delta \wedge X_5 \rightarrow l(x_0, x_0))$, in which $\mathcal{M}_{\mathsf{ppi}}$ is an RGCN model [37] that predicts PPIs, δ is a prediction threshold, where each vertex in Q_5 denotes a protein and each edge denotes a known PPI, and X_5 specifies additional logic conditions e.g., the domains and subcellular locations, on proteins in Q_5. This GAR identifies the self-interaction $l(x_0, x_0)$ on x_0. It has found a self-interaction on protein SYT2 (the bold line in Q_5) in May 2022, which coincides the finding published in Nature in the same month [31].

When the consequence p_0 of a GAR is an ML model for DDA, DDI or PPI, Fishing Fort can discover pattern Q and conditions X to explain the ML prediction, along the same lines as what we have shown in Section 3.2.1 for local explanations of online recommendation.

3.2.3 Credit Risk Assessment in Banking

Credit risk assessment is crucial for a bank in making decisions on applications for loans. It is the process of evaluating the likelihood that a borrower will default on a loan obligation. It helps minimize the risk of loss from defaults while also making credit available to qualified borrowers, thereby ensuring the bank's profitability and long-term sustainability.

In practice, a bank can use credit scoring models to rate borrowers based on their personal financial information, e.g., credit history, cash flows, and debt levels. In addition, it is equally important to look for other indicators, e.g., the borrower's social interactions, which may expose risks in financial health and the potential inability to repay a loan. However, this requires extensive labor from human experts. One of our FinTech partners employs a team of 1,000+ data analysts, dedicated to disclosing such hidden indicators.

Fishing Fort provides an automated and comprehensive approach. Better yet, it produces indicators that are highly explainable. On graphs G that model user/account interactions, Fishing Fort mined various GARs to assess user risks. A discovered GAR is $\varphi_6 = Q_6[\bar{x}](X_6 \wedge \mathcal{M}_{\mathsf{risk}}(x_0) = \mathsf{low} \rightarrow x_0.\mathsf{risk} = \mathsf{true})$, where $\mathcal{M}_{\mathsf{risk}}(x)$ is a credit scoring model that evaluates user x's default risk as either low, moderate or high, and X_6 is the conjunction of the following conditions: (1) a new user x_0 is referred by user x_1, who failed to repay a loan x_2; and (2) x_0 and x_1 shared a device x_3 for online banking. The rule suggests that user x_0 is likely a high-risk borrower if both pattern Q_6 and precondition X_6 are satisfied, despite the low-risk prediction made by model $\mathcal{M}_{\mathsf{risk}}$. Intuitively, by taking into account clear evidence of straw borrowing, GAR φ_6 corrects some false negatives of $\mathcal{M}_{\mathsf{risk}}$ predictions.

Another GAR $\varphi_7 = Q_7[\bar{x}](X_7 \rightarrow x_0.\mathsf{risk} = \mathsf{true})$ states that user x_0 is likely to default on a loan if x_0 meets the following conditions specified by pattern Q_7 and precondition X_7: (1) x_0 over-drafted bank account x_1 multiple times within the past year; (2) x_0 owns a car x_2 that is 20+ years old; and (3) x_0 is paying only minimum, compulsory liability insurance x_3 for the car. It would be difficult for a human expert to discover such a strong indicator of default risk, since any single factor alone constitutes a mere weak association.

Our partner in personal banking has verified the effectiveness of the GARs mined by Fishing Fort over real-world borrower data. Both φ_6 and φ_7 have a lift over 3, whereas the average lift of an expert-written rule is 2. These rules have been deployed at several banks because of their competitive performance in identifying high-risk borrowers.

3.2.4 Lithium-ion Battery Manufacturing

An electric vehicle (EV) battery pack consists of thousands of lithium-ion cells. These cells must have a balanced capacity, since imbalanced cells may reduce EV range and lead to safety issues such as thermal runaway. An essential phase in the battery manufacturing process consists of two steps: (a) battery formation, which activates a cell by performing the initial charge/discharge operation (to form special electrochemical solid electrolyte interphase at the electrode), followed by battery standing (to stabilize the voltage and cool off); and (b) battery grading, to fully discharge/recharge the cell and test its capacity. This phase is costly, taking from 14 to 20+ hours at different production lines in the industry, and demands specialized devices. It is energy-consuming; per estimation in [48], reducing the battery charge energy by half can save the production line $65,000 per GWh capacity (assuming a price of $0.13/kWh). It accounts for more than one third of the total cost of the Lithium-ion battery cell manufacturing process, and is considered one of the bottlenecks that hinders battery manufacturers from increasing output and reducing the total cost of battery production. Therefore, the Battery Formation and Grading System Market has been singled out as a specialized industry segment [30]. It is crucial for the entire Lithium-ion battery industry, including batteries for EVs, electronics and energy storage, among other things.

Fishing Fort provides a cost-effective solution. Traditionally the formation stage first activates the cell by charging it to a ratio of capacity and then cooling it off for a period of time. Grading then fully discharges it, recharges it to its maximum capacity, and discharges it again at a specific rate. It categorizes the capacity of the cell and its performance based on its individual characteristics. As opposed to full discharge/recharge in the traditional process, Fishing Fort only partially charges/discharges a cell. For example, it charges the cell to 50% in the formation stage to activate the cell, and collects data in the process; it then either grades the cell based on the data without discharging/recharging it, or fully discharges it and then partially charges it to a certain voltage for grading, to improve the accuracy. By analyzing the data together with data collected from earlier stages prior to formation, Fishing Fort can accurately grade the capacity of the cell and moreover, determine whether the cell is anomalous or not.

To this end, Fishing Fort first discretizes the time-series measurements; it splits the entire process into consecutive procedures based on their charging/discharging current. It models the data as a graph G with three types of vertices: (1) Procedure carries an array of attributes including the procedure ID, its initial/final weights, the initial/final battery state statistics, and the charging/discharging current; (2) State denotes a state in $\mathcal{S}$; and (3) Battery carries metadata of a battery cell, e.g., the cell ID, the testing slot ID, and its capacity interval. An edge denotes a transition between procedures, an association between a battery cell and a procedure (with range constraints), or an edge between a procedure and a state.

On such a graph G, Fishing Fort discovered GARs for capacity grading. A (simplified) GAR is $\varphi_8 = Q_8[\bar{x}](X_8 \to x_0.\mathsf{capacity} = 8)$, where its consequence grades a matching battery cell as Capacity Interval 8. Together with Q_8, X_8 specifies the following conditions: (1) the weight before and after the Electrolyte Filling procedure (x_1) is 555 ± 25g and 605 ± 25g, respectively; (2) its Formation-A procedure (x_2) uses a constant charging current at 3.8A, with initial voltage between 0–100mV and a final state 324 (x_4) categorized by $\mathcal{M}_{\mathsf{state}}$ (dashed arrows in Q_8); and (3) its Formation-B procedure (x_3) uses a constant charging current at 8.8A, with initial voltage between 3.3–3.4V and final state 738 (x_5). Intuitively, this rule grades the capacity of a battery cell based on the data collected from partial charge/discharge and from prior stages. It does not require full charge/discharge for the grading step.

In real production, about 3% of the cells have abnormal capacity and thus, should not be packed and assembled with other cells. Fishing Fort also detects anomalous cells online with GARs. It trains an ML model $\mathcal{M}_a$, a binary classification model based on LightGBM for detecting anomaly [28]. It embeds $\mathcal{M}_a$ in the GARs as a predicate, and filters false positives and false negatives of the predictions of $\mathcal{M}_a$ by adding logic predicates to the precondition X. Such a (simplified) GAR is $\varphi'_8 = Q'_8[\bar{x}](x_1.\mathsf{pt}_{10} \leq 1.9 \wedge x_1.\mathsf{pt}_{14} \geq 86 \wedge \mathcal{M}_{\mathsf{a}}(x_1) = \mathsf{false} \to x_0.\mathsf{status} = \mathsf{anomalous})$, where pattern Q'_8 is similar to Q_8 but does not contain vertices of State, and $x_1.\mathsf{pt}_{10}$ and $x_1.\mathsf{pt}_{14}$ are the values of the voltage and the temperature of cell x_0, respectively. GAR φ'_8 overrides the incorrect negative rulings of model $\mathcal{M}_{\mathsf{a}}$ by directly considering erratic measurements, thus effectively reducing the FNs of $\mathcal{M}_{\mathsf{a}}$.

With such GARs, Fishing Fort reduces charging to 35–50% of the full battery capacity, 75–100% of discharge (in some cases it avoids the grading step completely), and the time for the capacity grading process from 14 hours to 4 hours. With the data of partial charge/discharge, it keeps the error rate under 0.4%, a record in the industry. These translate to an 80% reduction in energy consumption for charging and cooling, cutting equipment costs by half.

3.2.5 Cyber Attack Detection

Cyber attack detection is increasingly vital to enterprise cybersecurity. It aims to identify, analyze, and mitigate threats that could compromise the integrity, confidentiality, and availability of a network. To this end, a common practice is to employ various specialized

hardware devices for threat detection and prevention, while ensuring compliance with security standards and regulations. For example, an enterprise network may deploy multiple instances of (a) Web Application Firewalls (WAF), which monitor, filter and block malicious traffic to and from a Web application, and (b) Intrusion Prevention Systems (IPS), which examines network traffic flows to detect and prevent vulnerability exploits.

However, such traditional cybersecurity devices have inherent limitations. First, they operate by identifying and mitigating known threats through predefined policies for access control, signatures for byte-level packet inspection and pattern matching, and detection mechanisms for anomalies; as a result, they can only respond *reactively* to security threats and lack the ability to take *proactive* measures. Second, they often make false positive (legitimate traffic identified as malicious) and false negative (malicious traffic not detected) predictions. This can lead to legitimate users being blocked or attackers slipping through defenses. Worse yet, each single device can only detect a subset of threats depending on its type, configuration and deployed location. Attackers may devise sophisticated, distributed and coordinated attacks to evade detection, leading to increasing false negative rates.

Overcoming these limitations requires advanced *threat intelligence* techniques, which gather and analyze information about emerging threats from various sources around the network. Fishing Fort approaches this by means of temporal graph analytics [18]. By integrating event-level threat intelligence feeds from all devices, Fishing Fort can synthesize security events in real-time, effectively discovering vulnerabilities and new attack vectors. It complements hardware-based solutions with a global view over the entire enterprise network.

As an example, consider GAR $\varphi_9 = Q_9[\bar{x}](X_9 \rightarrow x_0.\mathsf{attack} = \langle \mathsf{active}, t \in [t_4, t_4 + 1)\rangle)$ mined from device logs from an enterprise network. It predicts an active attack on subnet x_0 within the next one min, if the following event series have been reported where consecutive events are at most one min apart from each other: (1) an IPS device detects an Nmap event (x_1); (2) a WAF detects a scanner operation (x_2); (3) an IPS device reports a Startracker alert (x_3); and (4) an IPS device catches a WannaRen transmission (x_4). This rule has been deployed at a large company, where its accuracy is over 85%. Better still, it allows preemptive defensive measures *before* the attacks on the subnet are carried out.

4 Enriching and Cleaning Graphs

Fishing Fort is also able to enrich (Section 4.1) and clean (Section 4.2) real-life graphs.

4.1 Graph Enrichment

It is common to find the data in a real-life graph G_1 "incomplete", witnessed by missing properties and links. This is because in contrast to relational databases, graph G_1 is often "schemaless"; in such a graph, entities are specified by vertices and their subgraphs with irregular structures in the absence of constraints. With this comes the need for filling in the information missing from G_1 with data from external graphs G_2 that have overlapping information, thereby improving the accuracy of association analyses in graph G_1.

The need for referencing external graphs has been recognized in medical knowledge discovery [38], which aims to find semantic patterns and improve the accuracy and efficiency of diagnoses and treatment. As reported in [38, 45], as high as 77% of the discovered patterns span across heterogeneous biological networks. Moreover, e-commerce requires inspecting multiple graphs since an average social media user engages 6.6 different platforms [7]. Analyzing a single platform cannot fully understand users' interests. In addition, recommendation [41], information diffusing prediction [51] and network dynamics analysis [49] have been deducing associations across multiple graphs. Fishing Fort also enriches data for, e.g., drug discovery.

Fishing Fort implements the data enrichment algorithm of [19]. Given a real-life graph G_1 and an external graph G_2 with overlapping information, it enriches G_1 with only relevant information from G_2, to reduce noise and cost. It develops a graph filtering method to (a) identify vertices u in G_1 and v in G_2 that refer to the same real-world entity via heterogeneous entity resolution [15], and (b) locate relevant data of v in G_2 that pertains to u in G_1 by training an ML model. It supports two modes: (1) a physical mode by extracting properties of v from G_2 and enriches u by adding the data to G_1; and (2) a virtual mode by discovering GARs that pertain to entities in G_1 across G_1 and G_2, without physically merging the two; the discovered GARs are applied to G_1 and the filtered subgraphs of G_2, again without merging the two. Fishing Fort supports batch and incremental discovery of such GARs.

For drug discovery, Fishing Fort builds a uniform drug-disease graph by integrating data from eleven libraries and data banks, including CTD [5], MeSH [35] and BioGrid [2]. Given data banks $G_1 = (V_1, E_1, L_1, F_1)$ and $G_2 = (V_2, E_2, L_2, F_2)$, Fishing Fort "joins" the two as $G_\oplus(G_1, G_2) = (V_\oplus, E_\oplus, L_\oplus, F_\oplus)$, where $V_\oplus$ (resp. $E_\oplus$) is a revision of the union $V_1 \cup V_2$ (resp. $E \cup E_2$) such that u and v are represented as the same (merged) vertex in $V_\oplus$ if (u, v) is a match by parametric simulation [15]; and $L_\oplus$ and $F_\oplus$ inherit the label and attribute assignments from G_1 and G_2. When u and v both carry attribute A, the merged vertex takes the value $v.A$ from G_i ($\in [1, 2]$) if the data in G_i is more reliable. It incrementally enriches the graph in the physical mode by extracting data from external sources [19]. Moreover, it cleans the graph by detecting and fixing duplicates and inconsistencies (see Section 4.2).

4.2 Graph cleaning

Critical to graph analytics is the quality of the data in graphs. Real-life graphs often have duplicates and conflicts. Even knowledge graphs widely in use include such errors. In Yago, for instance, two different vertices named "Motorola" and "Team Motorola" refer to the same cycling team sponsored by Motorola, yielding a duplicate. The Baron Hirsch Synagogue, a building designed by architect George Awsumb, is labeled as a "flagship" of American Orthodox Judaism, while in DBpedia, this synagogue is classified as a ship. It is known that 26% of triples in knowledge graph NELL bear conflicts [52], and duplicate entities of Wikidata are involved in 27.8% factual statements it provides [39]. Moreover, noise in biomedical knowledge graphs is considered a big challenge to drug discovery [50]. Thus, two primitive issues of graph data quality are (a) *entity resolution (ER)*, to identify vertices that refer to the same real-world entity, and (b) *conflict resolution (CR)*, to fix inconsistencies among entities. The situation gets worse when it comes to data merged from multiple sources.

Besides ER and CR, there are two other critical issues of data quality. One is *timeliness* (a.k.a. data currency), for how up-to-date the information is. The other is *information completeness*, for the availability of data on-hand for analytics. In the real world, data easily becomes obsolete, and stale data is inaccurate. For instance, "82% of companies are making decisions based on stale information", and 85% of the companies witness that "this stale data is leading to incorrect decisions and lost revenue" [3]. Equally damaging is missing data. As revealed by a poll on clinical data management challenges in 2018, 77% of the participants rated missing patient data as a critical problem [42], which may result in incorrect diagnoses.

Hence the need is evident for (c) *timeliness deduction* (TD), to deduce temporal orders on attribute values, and (d) *missing data imputation* (MI), to fill in missing values/links.

One could use GARs to clean graphs. Indeed, GARs can express rules for ER, CR and MI. However, as indicated in Section 2.3, it is intractable to detect and fix errors with GARs. Moreover, GARs stop short of supporting temporal ranking and deducing timeliness.

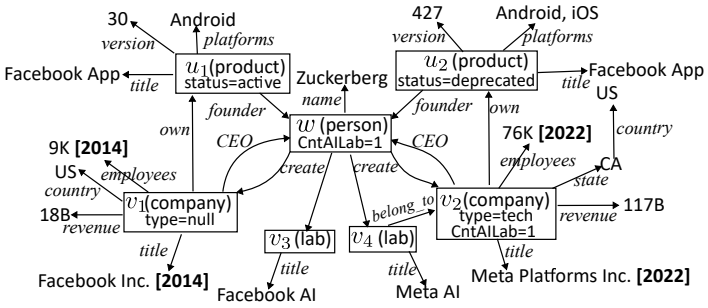

Figure 3 A graph G of companies and products.

Graph Cleaning Rules. In light of these, Fishing Fort develops a class of rules, referred to as GCRs [12]. A GCR has the form of $\mathcal{Q}[x_0, y_0](X \to p_0)$. It differs from a GAR in the following.

(1) On the one hand, it adopts a restricted form of graph patterns: $\mathcal{Q}[x_0, y_0]$ is a pair $\langle Q_x[x_0, \bar{x}], Q_y[y_0, \bar{y}]\rangle$ of patterns, where Q_x has a "star" shape centered at a vertex x_0, which denotes an entity of interest, and links to a set of characteristic features (leaves); similarly for Q_y. Intuitively, $\mathcal{Q}[x_0, y_0]$ represents two entities x_0 and y_0 with (possibly) different types and heterogeneous structures in a schemaless graph. It specifies features for pairwise comparison between x_0 and y_0. GCRs employ dual star-shaped patterns rather than general (possibly cyclic) patterns in GARs. It is in PTIME to check the matches of such patterns in a graph, as opposed to the intractability of general patterns.

(2) On the other hand, in addition to all the predicates of GARs, GCRs support temporal predicates $x.A \prec y.B$ (resp. $x.A \preceq y.B$), indicating that attribute $y.B$ is more current than $x.A$ (resp. at least as current as $x.A$). Moreover, Fishing Fort trains a temporal ranking model $\mathcal{M}_{\mathsf{rank}}$ $(x.A, y.B)$, which returns true iff $\mathcal{M}_{\mathsf{rank}}$ predicts that $x.A \preceq y.B$.

(3) GCRs further restrict the preconditions X such that all binary predicates are defined on two leaves x and y in Q_x and Q_y of $\mathcal{Q}$, respectively, and each leaf may carry at most one such predicate (but multiple unary predicates $z.A \oplus c$).

GCRs are able to express rules for ER, CR, TD and MI. As an example, DBpedia and Yago include two company entities having distinct titles "Facebook, Inc." and "Meta Platforms, Inc.", without any timestamp. Worse yet, other data of these entities is either outdated or missing, e.g., revenue, research labs, and type. There are also two products for the online social networking service named "Facebook App". They carry different values for, e.g., current status, version numbers and platforms, but with few timestamps. In fact, snapshot-based knowledge graphs, e.g., DBpedia, maintain the "most recent" facts with no timestamps. Figure 3 depicts such a graph G that includes the product entities u_1, u_2, and companies v_1, v_2. To catch errors, we may use the GCRs below with dual patterns in Figure 4.

(1) $\phi_1 = \mathcal{Q}_1[x_0, y_0](X_1 \wedge x_3.\mathsf{val} < y_3.\mathsf{val} \to x_3.\mathsf{val} \prec y_3.\mathsf{val})$, where X_1 is $x_1.\mathsf{id} = y_1.\mathsf{id} \wedge x_2.\mathsf{val} = y_2.\mathsf{val} \wedge \mathcal{M}_{\mathsf{rank}}(x_4.\mathsf{val}, y_4.\mathsf{val})$. It deduces temporal orders from the version numbers of an online service App (product), i.e., if two services x_0 and y_0 are created by the same person, have the same title, and if model $\mathcal{M}_{\mathsf{rank}}$ predicts that y_0 has newer value for platforms (specified in X_1), then the larger version number of y_0 is more current (for TD). In practice, the version number of an App grows as the platforms evolve.

(2) $\phi_2 = \mathcal{Q}_1[x_0, y_0](X_1 \wedge x_3.\mathsf{val} \prec y_3.\mathsf{val} \wedge x_0.\mathsf{status} = \text{active} \to y_0.\mathsf{status} = \text{active})$. That is, if service y_0 has newer values for version numbers (by $x_3.\mathsf{val} \prec y_3.\mathsf{val}$) than x_0 and if x_0 is active, then with the same condition X_1 as in (1), y_0 should also be active (for CR).

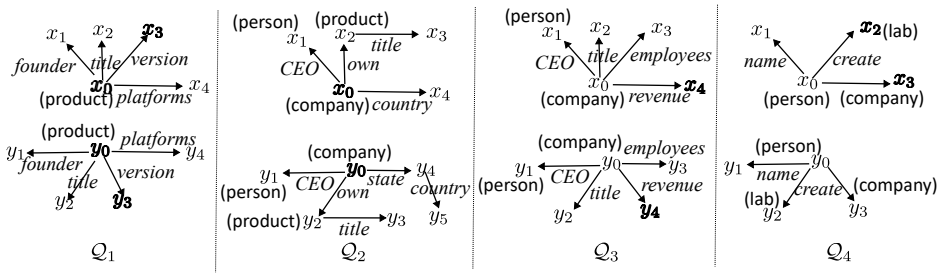

Figure 4 Example dual patterns.

(3) $\phi_3 = \mathcal{Q}_2[x_0, y_0](x_1.\mathsf{id} = y_1.\mathsf{id} \wedge x_2.\mathsf{status} = \text{active} \wedge y_2.\mathsf{status} = \text{active} \wedge x_3.\mathsf{val} = y_3.\mathsf{val} \wedge x_4.\mathsf{val} = y_5.\mathsf{val} \rightarrow x_0.\mathsf{id} = y_0.\mathsf{id})$. It identifies two companies x_0 and y_0 (for ER), if they have the same CEO and country of location, and if they own *active* products of the same title. Here the country property is fetched along different paths in the two stars of $\mathcal{Q}_2$.

(4) $\phi_4 = \mathcal{Q}_3[x_0, y_0](x_0.\mathsf{type} = \text{tech} \wedge y_0.\mathsf{type} = \text{tech} \wedge x_1.\mathsf{id} = y_1.\mathsf{id} \wedge \mathcal{M}_s(x_2.\mathsf{val}, y_2.\mathsf{val}) \wedge x_3.\mathsf{val} \prec y_3.\mathsf{val} \rightarrow x_4.\mathsf{val} \prec y_4.\mathsf{val})$. It deduces that company y_0 has a newer revenue than x_0 (for TD) if the number of employees of y_0 is more current, and both x_0 and y_0 are tech companies with the same CEO and similar titles (checked by ML model $\mathcal{M}_s$).

(5) $\phi_5 = \mathcal{Q}_4[x_0, y_0](x_1.\mathsf{val} = y_1.\mathsf{val} \wedge \mathcal{M}_e(x_2, y_2) \wedge x_3.\mathsf{id} = y_3.\mathsf{id} \wedge y_0.\mathsf{CntAILab} = 1 \wedge y_3.\mathsf{CntAILab} = 1 \rightarrow \mathsf{belong\_to}(x_2, x_3))$. The GCR states that if companies x_3, y_3 and AI labs x_2, y_2 are created by persons x_0 and y_0 of the same name, x_3 and y_3 are the same entity, x_2 and y_2 are identified by an ER model $\mathcal{M}_e$, and if y_0 and y_3 pertain to AI lab y_2, then x_2 is an AI lab belonging to x_3. It adds a missing link from x_2 to x_3 (for MI).

Moreover, Fishing Fort leverages the interaction among ER, CR, TD, and MI to improve the overall quality of the graph. Consider the GCRs above and graph G of Figure 3.

(a) Initially, we deduce that product u_2's version number (427) is more current than the version (30) of u_1 by using GCR ϕ_1 (TD).

(b) Then a conflict between the status values of products u_1 and u_2 is found by GCR ϕ_2, and u_2's status can be rectified to be active (CR).

(c) We next identify companies v_1 and v_2 by using GCR ϕ_3 (ER), where pattern nodes x_4 and y_5 in $\mathcal{Q}_2$ are mapped to vertices with value US.

(d) The type at v_1 is then copied from (more reliable) v_2 (MI), as a byproduct of ER.

(e) Next the revenue of v_2 is verified to be more current than v_1, via GCR ϕ_4 (TD).

(f) Finally, we deduce a missing link by GCR ϕ_5, i.e., the AI lab v_3 belongs to v_1 (MI).

ER, CR, TD and MI help each other. Indeed, ER step (c), CR step (b), TD steps (a) and (e), and MI steps (d) and (f) interact with each other in this process. For instance, step (b) is applicable only after we deduce temporal orders for version numbers in (a); and step (e) needs the results of ER and MI in (c)-(d), which decide the right type value for company v_1.

Complexity. The restrictions on graph patterns and preconditions make it easier to clean graphs with GCRs. The validation problem becomes tractable for GCRs. Moreover, error detection and correction with GCRs are also in PTIME (see below). One step further, the satisfiability and implication problems are in PTIME if we consider GCRs with predicates $x.A \oplus y.B$, $x.A \oplus c$ and $\mathcal{M}(x.\bar{A}, y.\bar{B})$ only, where $\oplus$ is either $=$ or $\neq$ [12]. These problems become intractable under any of the following conditions, unless P = NP: (a) in the presence of link predicates, temporal predicates or comparison $\leq, <, \geq, >$, which are necessary for, e.g., CR, TD, and MI; or (b) if the restrictions on the preconditions X of GCRs are lifted.

Algorithms. Given an (enriched) real-life graph G, Fishing Fort discovers a set Σ of GCRs and detects and fixes errors with the GCRs using the following algorithms.

(a) *Rule discovery* [12]. Fishing Fort discovers GCRs from samples of G as remarked earlier [13]. In contrast to how it mines GARs, it makes practical use of star patterns to speed up the discovery process [12]. It computes "scattered matches" of a star pattern, i.e., the matches $(h(x_0), h(y_0))$ of the center nodes in $\mathcal{Q}[x_0, y_0]$, rather than complete homomorphic matches h of $\mathcal{Q}$. The scattered matches can be computed in PTIME for star patterns via dynamic programming, as opposed to (possibly) exponential time for complete matches of general patterns. As shown in [12], it is much faster to discover GCRs than GARs.

(b) *Error detection* [12]. Employing the discovered GCRs in Σ, Fishing Fort detects errors in G, i.e., violations of a GCR $\phi = \mathcal{Q}[x_0, y_0](X \to p_0)$ in Σ, which are scattered matches $(h(x_0), h(y_0))$ for a match h such that $h \models X$ but $h \not\models p_0$. The violations can be duplicates, conflicts, conflicting timeline or missing links/attributes.

(c) *Error correction*. Fishing Fort enforces GCRs in Σ on graph G via chase [21], to deduce fixes. Besides ER and CR, in the process it determines temporal orders on graph properties, and fills in missing values and links by possibly referencing ground truth Γ and knowledge graph KG. It updates G with the deduced fixes, e.g., new edges for missing links. Moreover, it adds the fixes to ground truth Γ for subsequent error corrections. The chase is also Church-Rosser, and the fixes are logical consequences of Σ and Γ, i.e., the fixes are validated as long as the GCRs and ground truth are correct. In the process, it suffices to compute scattered matches of patterns, rather than complete ones.

All these algorithms are parallelly scalable and can scale with large graphs in principle. Moreover, the error detection and correction algorithms are in PTIME via scattered matches.

5 Conclusion

The novelty of Fishing Fort consists of the following. (1) It makes a first effort to conduct ML prediction and logic deduction in the same process, by embedding ML models as predicates in logic rules. As verified by real-life applications, the unification improves the accuracy of association analyses and the explainability of ML predictions. (2) Fishing Fort automatically discovers rules from real-life graphs, and it supports not only association analyses but also data enrichment and cleaning. (3) It can be readily adapted to different domains and has proven effective in a variety of applications. (4) The algorithms of Fishing Fort are provably parallelly scalable and hence in principle, can scale with large graphs.

We are currently adapting Fishing Fort to quantitative trading, fraud detection, and manufacturing industry beyond EV batteries. The preliminary results are quite encouraging.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 BioGRID, 2024. URL: `https://thebiogrid.org/`.

3 Businesswire. Over 80 percent of companies rely on stale data for decision-making, 2022. *https://www.businesswire.com/news/home/20220511005403/en/Over-80-Percent-of-Companies-Rely-on-Stale-Data-for-Decision-Making*.

4 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Comb.*, 12(4):389–410, 1992.

5 Comparative Toxicogenomics Database (CTD), 2024. URL: `https://ctdbase.org/`.

6 A Dairam, Edith M Antunes, KS Saravanan, and Santylal Daya. Non-steroidal anti-inflammatory agents, tolmetin and sulindac, inhibit liver tryptophan 2, 3-dioxygenase activity and alter brain neurotransmitter levels. *Life sciences*, 79(24):2269–2274, 2006.

7 Brian Dean. Social network usage and growth statistics. *https://backlinko.com/social-media-users*, 2023.

8 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

9 Exasol. Exasol research finds 58% of organizations make decisions based on outdated data, 2020. *https://www.exasol.com/news-exasol-research-finds-organizations-make-decisions-based-on-outdated-data/*.

10 Lihang Fan, Wenfei Fan, Ping Lu, Chao Tian, and Qiang Yin. Enriching recommendation models with logic conditions. *Proc. ACM Manag. Data*, 2024.

11 Wenfei Fan. Big graphs: Challenges and opportunities. *PVLDB*, 15(12):3782–3797, 2022.

12 Wenfei Fan, Wenzhi Fu, Ruochun Jin, Muyang Liu, Ping Lu, and Chao Tian. Making it tractable to catch duplicates and conflicts in graphs. *Proc. ACM Manag. Data*, 1(1):86:1–86:28, 2023.

13 Wenfei Fan, Wenzhi Fu, Ruochun Jin, Ping Lu, and Chao Tian. Discovering association rules from big graphs. *PVLDB*, 15(7):1479–1492, 2022.

14 Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. on Database Systems*, 33(1), 2008.

15 Wenfei Fan, Liang Geng, Ruochun Jin, Ping Lu, Resul Tugey, and Wenyuan Yu. Linking entities across relations and graphs. In *ICDE*, pages 634–647. IEEE, 2022.

16 Wenfei Fan, Tao He, Longbin Lai, Xue Li, Yong Li, Zhao Li, Zhengping Qian, Chao Tian, Lei Wang, Jingbo Xu, Youyang Yao, Qiang Yin, Wenyuan Yu, Kai Zeng, Kun Zhao, Jingren Zhou, Diwen Zhu, and Rong Zhu. GraphScope: A unified engine for big graph processing. *PVLDB*, 14(12):2879–2892, 2021.

17 Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Chao Tian, and Jingren Zhou. Capturing associations in graphs. *PVLDB*, 13(11):1863–1876, 2020.

18 Wenfei Fan, Ruochun Jin, Ping Lu, Chao Tian, and Ruiqi Xu. Towards event prediction in temporal graphs. *PVLDB*, 15(9):1861–1874, 2022.

19 Wenfei Fan, Muyang Liu, Shuhao Liu, and Chao Tian. Capturing more associations by referencing knowledge graphs. *PVLDB*, 2024.

20 Wenfei Fan and Ping Lu. Dependencies for graphs. *ACM Trans. Database Syst.*, 44(2):5:1–5:40, 2019.

21 Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. Deducing certain fixes to graphs. *PVLDB*, 12(7):752–765, 2019.

22 Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. Association rules with graph patterns. *PVLDB*, 8(12):1502–1513, 2015.

23 Wenfei Fan, Yinghui Wu, and Jingbo Xu. Functional dependencies for graphs. In *SIGMOD*, pages 1843–1857. ACM, 2016.

24 Wenfei Fan, Wenyuan Yu, Jingbo Xu, Jingren Zhou, Xiaojian Luo, Qiang Yin, Ping Lu, Yang Cao, and Ruiqi Xu. Parallelizing sequential graph computations. *ACM Trans. Database Syst.*, 43(4):18:1–18:39, 2018.

25 Chris Fotis, Asier Antoranz, Dimitris Hatziavramidis, Theodore Sakellaropoulos, and Leonidas G. Alexopoulos. Pathway-based technologies for early drug discovery. *Drug Discovery Today*, 2017.

26 Martin Grohe. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In *PODS*, pages 1–16. ACM, 2020.

27 Yang Hu, Xiyuan Wang, Zhouchen Lin, Pan Li, and Muhan Zhang. Two-dimensional Weisfeiler-Lehman graph neural networks for link prediction. *CoRR*, abs/2206.09567, 2022.

28 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

29 Clyde P. Kruskal, Larry Rudolph, and Marc Snir. A complexity theory of efficient parallel algorithms. *Theor. Comput. Sci.*, 71(1):95–132, 1990.

30 Market Research Intellignece Lab. Battery formation and grading system market size, outlook: Share, growth, and forecast (2024-2031), 2024. *https://www.linkedin.com/pulse/battery-formation-grading-system-market-yae8f/*.

31 Ying Lai, Giorgio Fois, Jose R Flores, Michael J Tuvim, Qiangjun Zhou, Kailu Yang, Jeremy Leitz, John Peters, Yunxiang Zhang, Richard A Pfuetzner, Luis Esquivies, Philip Jones, Manfred Frick, Burton F. Dickey, and Axel T. Brunger. Inhibition of calcium-triggered secretion by hydrocarbon-stapled peptides. *Nature*, 603(7903):949–956, 2022.

32 Jeanne C Latourelle, Merete Dybdahl, Anita L Destefano, Richard H Myers, and Timothy L Lash. Risk of parkinson's disease after tamoxifen treatment. *BMC neurology*, 10(1):1–7, 2010.

33 Bing Li, Wei Wang, Yifang Sun, Linhan Zhang, Muhammad Asif Ali, and Yi Wang. GraphER: Token-centric entity resolution with graph convolutional neural networks. In *AAAI*, pages 8172–8179, 2020.

34 Yu Li, Hiroyuki Kuwahara, Peng Yang, Le Song, and Xin Gao. PGCN: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks. *biorxiv*, page 532226, 2019.

35 Medical Subject Headings (MeSH), 2024. URL: `https://www.nlm.nih.gov/mesh/`.

36 R Sandyk and MA Gillman. Acute exacerbation of parkinson's disease with sulindac. *Annals of neurology*, 17(1):104–105, 1985.

37 Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web (ESWC)*, pages 593–607. Springer, 2018.

38 Feichen Shen and Yugyung Lee. Knowledge discovery from biomedical ontologies in cross domains. *PloS one*, 11(8):e0160005, 2016.

39 Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro A. Szekely. A study of the quality of Wikidata. *J. Web Semant.*, 72:100679, 2022.

40 Juan Shu, Yu Li, Sheng Wang, Bowei Xi, and Jianzhu Ma. Disease gene prediction with privileged information and heteroscedastic dropout. *Bioinformatics*, 37(Supplement_1):i410–i417, 2021.

41 Kai Shu, Suhang Wang, Jiliang Tang, Reza Zafarani, and Huan Liu. User identity linkage across online social networks: A review. *SIGKDD Explor.*, 18(2):5–17, 2016.

42 Julie Smiley. Missing data and its impact on clinical research, 2016. *https://blogs.oracle.com/health-sciences/post/missing-data-and-its-impact-on-clinical-research*.

43 Bo-Tao Tan, Li Wang, Sen Li, Zai-Yun Long, Ya-Min Wu, and Yuan Liu. Retinoic acid induced the differentiation of neural stem cells from embryonic spinal cord into functional neurons in vitro. *International journal of clinical and experimental pathology*, 8(7), 2015.

44 Xiaochan Wang, Yuchong Gong, Jing Yi, and Wen Zhang. Predicting gene-disease associations from the heterogeneous network using graph embedding. In *IEEE International conference on bioinformatics and biomedicine (BIBM)*, pages 504–511, 2019.

45 Antony J Williams, Lee Harland, Paul Groth, Stephen Pettifer, Christine Chichester, Egon L Willighagen, Chris T Evelo, Niklas Blomberg, Gerhard Ecker, Carole Goble, and Barend Mons. Open PHACTS: semantic interoperability for drug discovery. *Drug discovery today*, 17(21-22):1188–1198, 2012.

46 Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*, pages 9240–9251, 2019.

47 Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *ICML*, pages 12241–12252. PMLR, 2021.

48 Yuebo Yuan, Xiangdong Kong, Jianfeng Hua, Yue Pan, Yukun Sun, Xuebing Han, Hongxin Yang, Yihui Li, Xiaoan Liu, Xiaoyi Zhou, Languang Lu, and Hewu Wang. Fast grading method based on data driven capacity prediction for high-efficient lithium-ion battery manufacturing. *Journal of Energy Storage*, 73:109143, 2023.

49 Reza Zafarani and Huan Liu. Users joining multiple sites: Friendship and popularity variations across sites. *Inf. Fusion*, 28:83–89, 2016.

50 Xiangxiang Zeng, Xinqi Tu, Yuansheng Liu, Xiangzheng Fu, and Yansen Su. Toward better drug discovery with knowledge graph. *Current opinion in structural biology*, 72:114–126, 2022.

51 Qianyi Zhan, Jiawei Zhang, Senzhang Wang, Philip S. Yu, and Junyuan Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD*, pages 58–69, 2015.

52 Qinggang Zhang, Junnan Dong, Keyu Duan, Xiao Huang, Yezi Liu, and Linchuan Xu. Contrastive knowledge graph error detection. In *CIKM*, 2022.

53 Jie Zhao, Manish Kumar, Jeevan Sharma, and Zhihai Yuan. Arbutin effectively ameliorates the symptoms of parkinson's disease: The role of adenosine receptors and cyclic adenosine monophosphate. *Neural regeneration research*, 16(10):2030, 2021.