Discovering Patterns for Fact Checking in Knowledge Graphs

PENG LIN, QI SONG, and YINGHUI WU, Washington State University
JIAXING PI, Siemens Corporation

This article presents a new framework that incorporates graph patterns to support fact checking in knowledge graphs. Our method discovers discriminant graph patterns to construct classifiers for fact prediction. First, we propose a class of *graph fact checking rules* (GFCs). A GFC incorporates graph patterns that best distinguish true and false facts of generalized fact statements. We provide statistical measures to characterize useful patterns that are both discriminant and diversified. Second, we show that it is feasible to discover GFCs in large graphs with optimality guarantees. We develop an algorithm that performs localized search to generate a stream of graph patterns, and dynamically assemble the best GFCs from multiple GFC sets, where each set ensures quality scores within certain ranges. The algorithm guarantees a $(\frac{1}{2} - \epsilon)$ approximation when it (early) terminates. We also develop a space-efficient alternative that dynamically spawns prioritized patterns with best marginal gains to the verified GFCs. It guarantees a $(1 - \frac{1}{e})$ approximation. Both strategies guarantee a bounded time cost independent of the size of the underlying graph. Third, to support fact checking, we develop two classifiers, which make use of top-ranked GFCs as predictive rules or instance-level features of the pattern matches induced by GFCs, respectively. Using real-world data, we experimentally verify the efficiency and the effectiveness of GFC-based techniques for fact checking in knowledge graphs and verify its application in knowledge exploration and news prediction.

CCS Concepts: • **Information systems** → **Data cleaning**;

Additional Key Words and Phrases: Fact checking, supervised graph pattern mining, knowledge graph

ACM Reference format:
Peng Lin, Qi Song, Yinghui Wu, and Jiaxing Pi. 2019. Discovering Patterns for Fact Checking in Knowledge Graphs. *J. Data and Information Quality* 11, 3, Article 13 (May 2019), 27 pages.
https://doi.org/10.1145/3286488

1 INTRODUCTION

Knowledge graphs have been adopted to support emerging applications such as knowledge search [9], recommendation [34], and decision making [17]. A knowledge graph consists of a set of *facts*. Each fact is a triple statement $<v_x, r, v_y>$, where v_x and v_y denote a *subject* entity and an *object* entity, respectively, and r refers to a *predicate* (a relationship) between v_x and v_y. One of the

Y. Wu, P. Lin, and Q. Song were supported in part by NSF IIS-1633629, USDA/NIFA 2018-67007-28797, and research grants from Siemens and Huawei Technologies.

Authors' addresses: P. Lin, Q. Song, and Y. Wu, Washington State University, 355 NE Spokane Street, Pullman, WA 99164; emails: {peng.lin, qi.song, yinghui.wu}@wsu.edu; J. Pi, Siemens Corporation, 755 College Road East, Princeton, NJ 08540; email: jiaxing.pi@siemens.com.

1936-1955/2019/05-ART13 $15.00
https://doi.org/10.1145/3286488

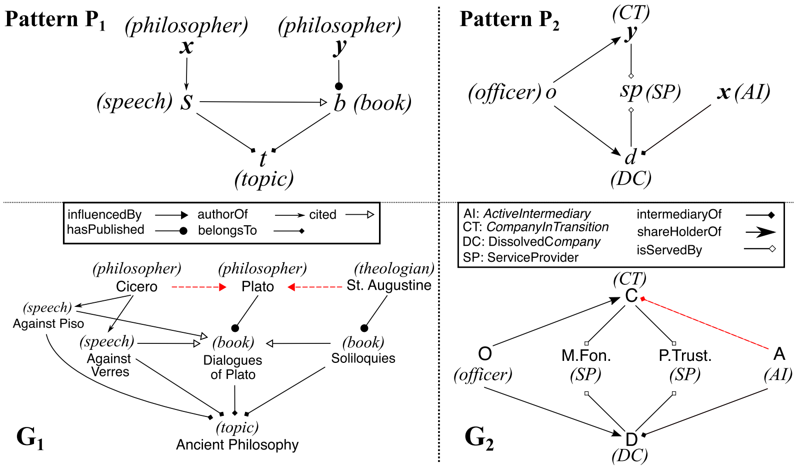


Fig. 1. Facts and associated graph patterns

cornerstone tasks for knowledge base management is *fact checking*. Given a knowledge graph G, and a fact t, it is to decide whether t belongs to the missing part of G. Fact checking can be used to (1) enrich incomplete knowledge bases [5, 9, 25, 33], (2) provide evidence for error detection in "dirty" knowledge bases [6, 16, 28, 48], and (3) improve the quality of knowledge search [32, 35].

Facts in real-world knowledge graphs are often associated with nontrivial regularities that involve both topological and semantic constraints. Such regularities can often be captured by graph patterns and their matches. Consider the following example.

Example 1. The graph G_1 in Figure 1 illustrates a fraction of DBpedia [25] about facts of philosophers. A user is interested in understanding whether and how philosophers influence each other. To check whether a fact <Cicero, `influencedBy`, Plato> is true in G_1, a regularity can be specified, which states that "if a philosopher v_x (e.g., 'Cicero') gave one or more speeches (e.g., 'Against Piso') that cited a book (e.g., 'Dialogues of Plato') of a philosopher v_y (e.g., 'Plato') with the same topic (e.g., 'Ancient Philosophy'), then v_x is likely to be influenced by v_y." Such regularities can be easily represented by a graph pattern P_1 associated to philosophers. As "Cicero" and "Plato" are matches of P_1, the triple <Cicero, `influencedBy`, Plato> should be true in G_1.

Consider graph G_2, a fraction of a real-world offshore activity network [19] in Figure 1. To find whether an active intermediary (`AI`) `A` is likely to serve a company in transition (`CT`) `C`, a pattern P_2 that explains such an action may identify G_2 by stating that "`A` is likely an intermediary of `C` if it served for a dissolved company `D` which has the same shareholder `O` and one or more providers with `C`."

We also have the following observations:

(1) Patterns with "weaker" constraints may not explain facts well. Consider a graph pattern P_1' obtained by removing the edge `cited`(*speech*, *book*) from P_1. Although "Cicero" and "Plato" match P_1', a false fact <Cicero, `influencedBy`, John Stuart Mill> also matches P_1' (not shown), thus P_1' alone does not distinguish true and false facts for `influencedBy`(*philosopher*, *philosopher*) well.
(2) Facts that "approximately" match P_1 can also be true. Consider a fact <St.Augustine, `influencedBy`, Plato> in G_1. It states that "theologian" St. Augustine was influenced by Plato, indicated by his book (e.g., "Soliloquies") that cited "Dialogues of Plato." The fact

should be identified as true, given that "theologian" is usually related to "philosopher," and "book" may record "speech," although these pairs of labels do not match exactly.

The preceding examples show that discriminant patterns, which can distinguish between true and false facts well, are often more useful in "defining" true facts. These graph patterns can be easily interpreted as rules, and the matches of the graph patterns readily provide relevant instance-level evidence to "explain" the facts. These matches also indicate accurate predictive models for various similar facts. We ask the following question: How to characterize useful graph patterns and efficiently discover these graph patterns to support fact checking in large knowledge graphs?

Contributions. We propose new fact checking models that explicitly incorporate discriminant graph patterns to support fact checking in knowledge graphs. It nontrivially extends Lin et al. [27] by including new rule models, complete proofs, new pattern discovery algorithms with optimality guarantees, and enriched experimental studies.

A new rule model. We introduce a class of *graph fact checking rules* (GFCs) (Section 2). GFCs incorporate discriminant graph patterns as the antecedent and generalized triple patterns as the consequent, and characterize fact checking with approximate graph pattern matching. GFCs extend their counterparts in Lin et al. [27] with a new capacity of automatically identifying and exploiting similar facts that are not necessarily exact instances of the targeted triple pattern. Thus, GFCs can predict new triples that may not exactly match any known triple patterns (consequent). We adopt computationally efficient pattern matching to ensure tractable fact checking.

We develop several statistical measures, including support, confidence, significance, and diversity, to characterize useful GFCs (Section 3). These criteria are defined under a weak partial closed world assumption (WPCA) that reduces the impact from uncertain facts, especially for nonfunctional relations. Based on these measures, we formulate the top-k GFC discovery problem to discover useful GFCs for the fact checking task.

Feasible rule discovery algorithms. We develop supervised pattern discovery algorithms to compute useful GFCs (Section 4). These algorithms solve a submodular optimization problem with provable optimality guarantees by a single scan of graph patterns without visiting the entire pattern set. They incur a small update cost in response to the arrival of new patterns. The general ideas of our two discovery algorithms are summarized as follows:

(1) We extend the "sieve-streaming" strategy in Lin et al. [27] for the new GFC model. We show that stream-based pattern mining guarantees a $(\frac{1}{2} - \epsilon)$ approximation with an update cost determined by ϵ and the size of neighborhood of training facts up to a certain hop.
(2) We also introduce a new rule discovery strategy that exploits a prioritized spawning strategy. By exploiting the antimonotonicity of marginal gains of GFCs, we dynamically select the patterns that can best improve the marginal benefit during the pattern generation and selection. We show that this ensures a $(1 - \frac{1}{e})$ approximation.

We compare and suggest the scenarios for which each method works better than the others.

New fact checking models. To ensure the applications of GFCs, we develop two classifiers to predict the existence of the facts to be checked. The first model directly uses GFCs as predictive rules to enhance rule-based classifiers. The second extracts richer instance-level features from the pattern matches induced by GFCs to learn a feature-based classifier (Section 5).

Enriched experimental studies. Using real-world graphs, we experimentally verify the efficiency and effectiveness of GFC-based techniques (Section 6). We found that the discovery algorithms of GFCs are feasible over large graphs. GFC-based fact checking also achieves high accuracy and outperforms its counterparts using Horn clause rules and path-based prediction. We also show that the models are highly interpretable and can predict facts that do not exactly match known triple patterns in the graph, as verified by our case studies.

These results verify the effectiveness of GFC-based techniques compared to additional baseline fact checking models, over diversified real-world datasets, and case analysis that leads to new applications of GFC algorithms. These are not addressed in Lin et al. [27].

Related work. We categorize the related work as follows.

Fact checking. Fact checking has been studied for both unstructured data [13, 38] and structured (relational) data [18, 49]. This work relies on text analysis and crowd sourcing. Automatic fact checking in knowledge graphs is not addressed in this work. Beyond relational data, several methods have been studied to predict triples in graphs:

(1) Rule-based models extract rules that describe associations between a pattern P (antecedent) and a template of facts r (consequent). The rules state that an instance of r should exist if it also satisfies the description of the pattern P. These models infer the existence of a fact by testing if it satisfies at least one of a set of rules. For example, AMIE [14, 15] discovers rules with conjunctive Horn clauses for knowledge base enhancement. Beyond Horn rules, GPARs [11] discover association rules in the form of $Q \Rightarrow p$, with a subgraph pattern Q (defined by subgraph isomorphism) and a single edge p. GPARs recommend users via co-occurred frequent subgraphs. The rules can be provided by users or learned from positive examples that are validated as true.

(2) Supervised link prediction has been applied to train predictive models with (latent) features extracted from entities [9, 23]. Recent works make use of path features [7, 8, 23, 39] and subgraph features [16, 45]. The paths involving targeted entities are sampled from 1-hop neighbors [8] or via random walks [23], or are constrained to be shortest paths [7]. Discriminant paths with the same ontology are grouped to generate positive and negative examples in Shi and Weninger [39]. Beyond random walk-based sampling, Gardner and Mitchell [16] and Thor et al. [45] also exploit subgraph-based features. Subgraphs around entities are first extracted and summarized. Paths are then sampled and repopulated by replacing original labels with similar ones. These paths are then used as features to train classifiers for link prediction.

(3) Embedding-based algorithms differ from rule-based algorithms in that they predict links by computing proper vector transformations. Entities are first mapped to vectorized representations called *embeddings*. Given training facts, the goal is then to learn a transformation model that minimizes a ranking loss such that pairs of entities in true facts are closer to each other than those in nonexisting relationships. They optimize a score function carried out by, for example, stochastic gradient decent, to make the sum of the subject vector and the relation vector be close to the object vector as much as possible (cf. [4, 22]). Deep learning is also applied, mostly for facts in unstructured texts instead of exploiting topological features in graphs [32].

Rule-based models are easy to interpret but usually cover only a subset of useful patterns [32]. Path-based features sampled by random walks or from subgraphs also provide explainable models [7, 23, 39] but lack the necessary expressiveness to explicitly describe subgraphs as discriminant features. However, latent feature models are more difficult to interpret than rule models [32]. Our work aims to balance the interpretability and the construction cost of the models. In contrast to AMIE [15], we use more expressive rules enhanced by graph patterns to express both constant and topological context of facts. Unlike Fan et al. [11], we use approximate pattern matching for GFCs instead of subgraph isomorphism, which may produce redundant examples and is computationally hard in general. GFCs can induce useful and discriminant features from patterns and subgraphs, beyond path features [7, 23, 39]. GFCs can be used as a stand-alone rule-based method. They also provide context-dependent features to support supervised link prediction to learn highly interpretable models. These are not addressed in Fan et al. [11] and Galárraga et al. [15].

Graph pattern mining. Frequent pattern mining defined by subgraph isomorphism has been studied for a single graph. GRAMI [10] discovers frequent subgraph patterns without edge labels. Parallel algorithms are also developed for association rules with subgraph patterns [11]. In contrast, first, we adopt approximate graph pattern matching for feasible fact checking rather than subgraph isomorphism as in Elseidy et al. [10] and Fan et al. [11]. Second, we develop a more feasible stream mining algorithm with optimality guarantees on rule quality, which incurs a small cost to process each pattern. Third, supervised graph pattern mining over observed ground truth is not discussed in Elseidy et al. [10] and Fan et al. [11]. In contrast, we develop algorithms that discover discriminant patterns that best distinguish observed true and false facts.

Graph dependency. Data dependencies have been extended to capture errors in graph data. Functional dependencies for graphs (GFDs) [12] enforce topological and value constraints by incorporating graph patterns with variables and subgraph isomorphism. These hard constraints (e.g., subgraph isomorphism) are useful for detecting data inconsistencies but are often violated by incomplete knowledge graphs for fact checking tasks. We focus on "soft rules" to infer new facts toward data completion rather than identifying errors with hard constraints [35].

Graph completion. Several other problems related to fact checking have been studied, such as knowledge graph completion [3, 40] (see Cai et al. [4] for a survey). In addition, node classification infers missing node labels [2], entity resolution [36] can be viewed as predicting a *sameAs* relation between two entities, and fact checking specifically for social networks recommends potential friendships or collaborations between users [11]. Notable learning-based fact checking models include translation embedding (TransE) [3] and its variants [28, 40]. For example, ProjE [40] solves a ranking problem by projecting the entities onto a ranking score vector with a combining operator. Factorization-based methods treat the graph completion as a dimension reduction problem, which preserves the invariant properties between entities [46]. Beyond rule models, GFCs can be used to provide useful, interpretable, and discriminant features for embedding-based methods. We envision that such a combination provides more interpretable knowledge graph completion methods. We will study such methods in future work.

2 FACT CHECKING WITH GRAPH PATTERNS

We first review several notions of knowledge graphs and fact checking.

Knowledge graphs. A knowledge graph is a directed graph $G = (V, E, L)$, where V is a finite set of nodes (entities) and $E \subseteq V \times V$ is a set of edges (relationships). For each node $v \in V$ (respectively, edge $e \in E$), $L(v)$ (respectively, $L(e)$) is a label that encodes the content of v (respectively, e), such as types, properties, or names (respectively, relations or actions) as found in knowledge bases and social networks. Knowledge graphs are graph-based representations of knowledge bases [9]. A *fact* (triple) $<v_x, r, v_y>$ in a knowledge base [9] can be encoded as an edge $e = (v_x, v_y)$ with an edge label r, where v_x and v_y are two nodes in G, and x and y are their node labels, respectively (i.e., $x = L(v_x)$ and $y = L(v_y)$).

Fact checking in knowledge graphs. Given a knowledge graph $G = (V, E, L)$ that contains two nodes v_x and v_y, and a fact $<v_x, r, v_y>$ not in E, the task of fact checking is to decide whether

$<v_x, r, v_y>$ exists in G [32]. In other words, the goal is to compute a model that can answer the question "does relation r exist between entities v_x and v_y?"

Graph pattern matching revisited. Before we present our fact checking model, we first introduce a class of graph patterns that incorporate approximate label and topological matching.

Graph pattern. A graph pattern $P(u_x, u_y)=(V_P, E_P, L_P)$ is a directed graph that contains a set of pattern nodes V_P and pattern edges E_P. Each pattern node $u_p \in V_P$ (respectively, edge $e_p \in E_P$) has a label $L_P(u_p)$ (respectively, $L_P(e_p)$). Moreover, it contains two designated *anchored nodes* u_x and u_y in V_P, with labels x and y, respectively. For simplicity, we will refer to $P(u_x, u_y)$ as $P(x, y)$ or P.

Specifically, a *triple pattern* $r(u_x, u_y)$ (or simply $r(x, y)$) is a graph pattern that contains a single pattern edge with two pattern nodes with labels x and y, respectively.

Graph pattern matching. Applying graph patterns that are defined by subgraph isomorphism for fact checking can be expensive for large graphs, due to an excessive number of identical matches, leaving alone its high complexity (NP-hard) [27]. In contrast, we incorporate both label and topological similarities for graph pattern-based fact checking, bringing pattern matching to tractable cases.

Given two labels l and l', we make use of a label similarity predicate to verify if l and l' are similar for a similarity threshold α, denoted as $l \sim_\alpha l'$. Given a label similarity predicate, we extend the approximate pattern matching [27, 43] as a matching relation $R \subseteq V_p \times V$ between pattern P and graph G, under similarity threshold α. For each node pair, $(u, v) \in R$, $L(u) \sim_\alpha L(v)$, and moreover,

- for every edge $e = (u, u') \in E_P$, there exists an edge $e' = (v, v') \in E$ such that $L(u') \sim_\alpha L(v')$, and $L(e) \sim_\alpha L(e')$, and
- for every edge $e = (u'', u) \in E_P$, there exists an edge $e'' = (v'', v) \in E$ such that $L(u'') \sim_\alpha L(v'')$, and $L(e) \sim_\alpha L(e'')$.

In other words, the relation R preserves parent-child relations of each pattern node to its matches with similar entity and relation labels. We say a pattern P *covers* a fact $<v_x, r, v_y>$ in G with threshold α if v_x and v_y match its anchored nodes u_x and u_y, respectively, via matching relation R. In practice, the predicate can be defined by string transformations [51], such as synonyms and acronyms, or semantic similarity [52] for labels that are concepts.

Example 2. The fact t = <Cicero, `influencedBy`, Plato> (Figure 1) is represented by an edge between two nodes, "Cicero" and "Plato," with label "influencedBy" in G_1. Both "Cicero" and "Plato" have a label specifying their type "philosopher." The regularity in Example 1 that verifies the fact t can be represented by a graph pattern P_1 with two anchored nodes x and y, both labeled by "philosopher." A targeted triple pattern for the fact checking task is $r(x, y)$=`influencedBy`$(philosopher, philosopher)$. Both P_1 and $r(x, y)$ cover the fact t.

The fact <St. Augustine, `influencedBy`, Plato> is also covered by P_1, given that (1) a predicate, such as concept similarity [52], asserts that "theologian" is related to "philosopher," and "book" is similar to "speech," and (2) "St. Augustine" and "Plato" match the two anchored nodes of P_1, respectively.

We now introduce our rule model that incorporates graph patterns.

Rule model. A *graph fact checking rule* (denoted as GFC) is in the form of $\varphi : P(x, y) \rightarrow r(x, y)$, where (1) $P(x, y)$ and $r(x, y)$ are two graph patterns carrying the same pair of anchored nodes (u_x, u_y), and (2) $r(x, y)$ is a triple pattern and is not in $P(x, y)$.

Semantics. A GFC $\varphi : P(x, y) \rightarrow r(x, y)$ states that *a fact* $<v_x, r, v_y>$ *holds between* v_x *and* v_y *in* G, *if* (v_x, v_y) *is covered by* P. In other words, given a new fact t=$<v_x, r, v_y>$ not in G, φ states that t *should* exist in G if it is an instance (match) of $r(x, y)$, and v_x and v_y are also matches of $P(x, y)$.

To ensure computationally efficient fact checking, we prefer approximate patterns instead of subgraph patterns. Subgraph isomorphism may be an overkill in capturing meaningful patterns [29, 42, 43] and is computationally expensive (NP-hard). Moreover, it generates (exponentially) many isomorphic subgraphs and thus introduces redundant features for model learning. In contrast, it is tractable to decide whether a fact is covered by an approximate pattern [29, 43]. The tractability carries over to the validation of GFCs (Section 4).

Example 3. Consider the patterns and graphs in Figure 1. First, to verify the influence between philosophers, a GFC is given by φ_1:$P_1(\mathit{philosopher}, \mathit{philosopher}) \rightarrow$ `influencedBy` $(\mathit{philosopher}, \mathit{philosopher})$. Second, another GFC is given by φ_2:$P_2(\mathit{AI}, \mathit{CT}) \rightarrow$ `intermediaryOf` $(\mathit{AI}, \mathit{CT})$ to verify whether there is a service relationship between the pair of offshore entities (`A`, `C`) in G_2.

When the graph pattern P_1 is defined by subgraph isomorphism, it induces two subgraphs of G_1 that only differ by entities with label `speech`. Subgraphs with highly overlapped entities induced by a same graph pattern $P(x, y)$ are often redundant for predicting instances of $r(x, y)$, not to mention the high cost of enumeration. However, φ_1 defined by rules in Lin et al. [27] cannot be used to predict the fact <St. Augustine, `influencedBy`, Plato> due to enforced label equality.

Remarks. The rule model in Lin et al. [27] is a special case of our GFC model by enforcing label equality. By incorporating approximate pattern matching with label similarity, GFCs further mitigate incomplete data and capture richer features for supervised fact checking (see Section 4). This differs GFCs (even with label equality) from Horn-clause rules [14] and subgraph isomorphism–based association rules [11], as verified in Lin et al. [27].

3 DISCOVERING GFCS FOR FACT CHECKING

We next introduce several criteria to characterize useful GFCs. These measures extend their conventional counterparts used by established rule models [15] and discriminant patterns [50], and are specialized for training facts and the approximate graph pattern matching.

3.1 Statistical Measures

Given a knowledge graph $G = (V, E, L)$, we characterize useful GFCs in terms of a set of *training facts* Γ. The training facts Γ consists of a set Γ^+ of true facts in G and a set Γ^- of false facts that are known not in G, respectively. Extending the *silver standard* in knowledge base completion [35], (1) Γ^+ can be usually sampled from manually cleaned knowledge bases [30], and (2) Γ^- are populated following a weak form of the conventional partial closed world assumption (see the Confidence section).

We shall use the following notations. Given a GFC $\varphi : P(x, y) \rightarrow r(x, y)$, the graph G, training facts Γ, first, $P(\Gamma^+)$ (respectively, $P(\Gamma^-)$) refers to the set of training facts in Γ^+ (respectively, Γ^-) that are covered by $P(x, y)$ in Γ^+ (respectively, Γ^-). $P(\Gamma)$ is defined as $P(\Gamma^+) \cup P(\Gamma^-)$ (i.e., all the facts in Γ covered by P). Second, $r(\Gamma^+)$, $r(\Gamma^-)$, and $r(\Gamma)$ are defined similarly. For the rest of this section, we assume that a knowledge graph G, training facts Γ, and a label similarity predicate (with a similarity threshold α) are given.

Support. The support of a GFC $\varphi : P(x, y) \rightarrow r(x, y)$, denoted by $\mathrm{supp}(\varphi)$, is defined as

$$\mathrm{supp}(\varphi) = \frac{|P(\Gamma^+) \cap r(\Gamma^+)|}{|r(\Gamma^+)|}.$$

The support is the fraction of the true facts that covered by both $P(x, y)$ and $r(x, y)$ over those only covered by $r(x, y)$. It extends head coverage, a practical version for rule support [15] to address triple patterns $r(x, y)$ without many matches due to the incompleteness of knowledge bases.

We show that the support of GFCs is well defined, with *antimonotonicity* under rule refinement.

Rule refinement. Given two GFCs $\varphi_1 : P_1(x, y) \to r(x, y)$ and $\varphi_2 : P_2(x, y) \to r(x, y)$, where $P_1(x, y)$, $P_2(x, y)$, and $r(x, y)$ pertain to the same anchored nodes u_x and u_y, we say that φ_2 *refines* φ_1, denoted as $\varphi_1 \preceq \varphi_2$, if P_1 is a subgraph of P_2.

Lemma 3.1 [Antimonotonicity of Rule Support]. *Given the graph G, and any two* GFCs *φ_1: $P_1(x, y) \to r(x, y)$ and $\varphi_2 : P_2(x, y) \to r(x, y)$,* $\mathsf{supp}(\varphi_2) \leq \mathsf{supp}(\varphi_1)$ *if $\varphi_1 \preceq \varphi_2$.*

Proof. It suffices to show that any node pair (v_{x_2}, v_{y_2}) in G covered by P_2 in Γ^+ is also covered by P_1 (i.e., $P_2(\Gamma^+) \subseteq P_1(\Gamma^+)$). Assume that there exists a node pair (v_{x_2}, v_{y_2}) covered by P_2 but not covered by P_1, and assume that, without loss of generality, v_{x_2} does not match the anchored node u_x in P_1. Then there exists either (a) an edge (u_x, u) (or (u, u_x)) in P_1 such that no edge (v_{x_2}, v) (or (v, v_{x_2})) is a match or (b) a node u as an ancestor or a descendant of u_x in P_1 such that no ancestor or descendant of v_{x_2} in G is a match. As P_1 is a subgraph of P_2, both (a) and (b) lead to a contradiction that v_{x_2} is covered by P_2. Thus, $P_2(\Gamma^+) \subseteq P_1(\Gamma^+)$. As $r(\Gamma^+)$ is fixed for given $r(x, y)$ and Γ^+, $|P_2(\Gamma^+) \cap r(\Gamma^+)| \leq |P_1(\Gamma^+) \cap r(\Gamma^+)|$. Hence, $\mathsf{supp}(\varphi_2) \leq \mathsf{supp}(\varphi_1)$. □

Confidence. We evaluate the confidence of GFCs under a "weaker" form of the established partial closed world assumption (PCA) [9, 15], denoted as weak PCA or WPCA. The motivation is to reduce the impact of facts that are not necessarily false to the evaluation of the confidence of GFCs, given their similarity to true facts asserted by the label similarity predicate.

Weak PCA. Given a triple pattern $r(x, y)$ and a true fact $<v_x, r, v_y> \in r(\Gamma^+)$, we say that a fact (v_x, r, v'_y) is a false fact *witnessed* by $<v_x, r, v_y>$ if $L(v_y) \not\sim_\alpha L(v'_y)$, under WPCA. We remark that PCA used in other works [9, 15, 27] is a special case of WPCA: when $L(v_y) \neq L(v'_y)$ is enforced, WPCA "degrades" to PCA. Indeed, a true fact $<v_x, r, v_y>$ witnesses a false fact (v_x, r, v'_y) as long as $L(v_y) \neq L(v'_y)$ under PCA. This nevertheless may not hold for many nonfunctional (many-to-many) relations.

Example 4. Considered a triple pattern $r'(x, y)$=`influences`(*philosopher*, *philosopher*) (the inverse relation of "influencedBy"), and a true fact t_1 = <Plato, `influences`, Cicero> in Γ^+, observe the following. First, t_2 = <Plato, `influences`, Kurt Gödel>, where Kurt Gödel is labeled as "logician," is a true fact in the real world. This missing fact is considered to be false under PCA (due to a witness t_1) but is excluded to be a false case under WPCA, given that "logician" is often considered as a type of philosopher, asserted by a predicate that quantifies closeness of concepts (i.e., L(Kurt Gödel) $\sim_\alpha$ L(Cicero)). Note that WPCA does not assert t_2 to be true unless $t_2 \in \Gamma^+$. Second, t_3 = <Plato, `influences`, "Tom & Jerry"> is a false fact under both PCA and weak PCA. For the latter, the TV show *Tom & Jerry* is not semantically close to a "philosopher" (i.e., L(Plato) $\not\sim_\alpha$ L("Tom & Jerry"). Third, t_4 = <Plato, `isFriendOf`, Kurt Gödel> is undecidable to be false under both PCA and WPCA, given that no friend of Plato is known in Γ^+, in which case the witness true fact is missing.

Intuitively, WPCA excludes "false" facts (v_x, r, v'_y) under PCA that are actually "unknown" cases unless these facts have significantly different objects v'_y witnessed by known true facts having same subject v_x and predicate r. This helps GFCs to better predict true facts carrying nonfunctional relations (e.g., `influencedBy`), for which PCA may not hold.

Under WPCA, the confidence of a GFC φ in G, denoted as $\mathsf{conf}(\varphi)$, is defined as

$$\mathsf{conf}(\varphi) = \frac{|P(\Gamma^+) \cap r(\Gamma^+)|}{|N(\Gamma^+)|}.$$

The confidence is normalized by a set $N(\Gamma^+)$ of entity pairs (v_x, v_y) such that (1) v_x and v_y occur in some (but not necessarily the same) true fact in $P(\Gamma^+)$ and (2) there exists a false fact given (v_x, v_y), "assuming" $<v_x, r, v_y>$ is a true fact under WPCA.

The confidence measures the probability that a GFC holds over the entity pairs that satisfy $P(x, y)$, normalized by all the possible true facts that witness at least a false fact under WPCA. We follow this assumption to construct false facts in our experimental study (see Section 6).

Significance. Our third criterion quantifies how well a GFC φ "distinguishes" between the true and false facts. To this end, we extend the G-test [50] for φ. Specifically, its significance score (denoted as $\mathsf{sig}(\varphi, p, n)$, or simply $\mathsf{sig}(\varphi)$) is defined as

$$\mathsf{sig}(\varphi, p, n) = 2|\Gamma^+| \left(p \ln \frac{p}{n} + (1-p) \ln \frac{1-p}{1-n}\right),$$

where p (respectively, n) is the fraction of the facts covered by graph pattern P of φ in Γ^+ (respectively, Γ^-) (i.e., $p = \frac{|P(\Gamma^+)|}{|\Gamma^+|}$ (respectively, $n = \frac{|P(\Gamma^-)|}{|\Gamma^-|}$)). Intuitively, the G-test verifies the null hypothesis that the number of true facts covered by $P(x, y)$ fits its distribution in the false facts. If not, $P(x, y)$ is considered to be statistically significant in distinguishing true and false facts. For example, when P does not distinguish true and false facts ($p = n$), $\mathsf{sig}(\varphi)$ is 0. The higher $\mathsf{sig}(\varphi)$ is, the more "discriminant" P is.

Remarks. It is observed that significance determined by p values and a user-specific threshold can be sensitive to applications [37]. We do not compute p values nor assign a threshold of $\mathsf{sig}(\cdot)$ to decide whether a pattern is discriminant. Instead, we incorporate $\mathsf{sig}(\cdot)$ directly as a part of a normalized ranking function to discover top GFCs (see the Redundancy-aware selection section).

Redundancy-aware selection. In practice, one wants to choose GFCs with both high significance and low redundancy. Indeed, a set of GFCs can be less useful if they "cover" the same set of true facts in Γ^+. We introduce a bi-criteria function that favors significant GFCs that cover more diversified true facts. Given a set of GFCs $\mathcal{S}$, when the true facts Γ^+ and the false facts Γ^- are known, the *coverage score* of $\mathcal{S}$, denoted as $\mathsf{cov}(\mathcal{S})$, is defined as

$$\mathsf{cov}(\mathcal{S}) = \mathsf{sig}(\mathcal{S}) + \mathsf{div}(\mathcal{S}).$$

More specifically, (1) the term $\mathsf{sig}(\mathcal{S})$, defined as $(\sum_{\varphi \in \mathcal{S}} \mathsf{sig}(\varphi))^{\frac{1}{2}}$, aggregates the total significance of GFCs in $\mathcal{S}$, and (2) the term $\mathsf{div}(\mathcal{S})$, defined as $(\sum_{t \in \Gamma^+} (\sum_{\varphi \in \Phi_t(\mathcal{S})} \mathsf{supp}(\varphi))^{\frac{1}{2}})/|\Gamma^+|$, quantifies the diversity of $\mathcal{S}$ following a diversity reward function [26]. Here, $\Phi_t(\mathcal{S})$ refers to the set of GFCs in $\mathcal{S}$ that cover a true fact $t \in \Gamma^+$. Intuitively, it rewards the diversity benefit in selecting a GFC that covers new facts not covered by other GFCs in $\mathcal{S}$ yet. Both terms are normalized to $(0, |\mathcal{S}|^{\frac{1}{2}}]$.

The coverage score favors GFCs that cover more distinct true facts with more discriminant patterns. We now formulate the supervised top-k GFC discovery problem.

Problem statement. Given the graph G, a triple pattern $r(x, y)$, thresholds σ and θ for support and confidence, and training facts Γ of $r(x, y)$, the top-k GFC discovery problem is to compute a set $\mathcal{S}^*$ of k GFCs that pertain to $r(x, y)$ such that for each GFC $\varphi \in \mathcal{S}^*$, $\mathsf{supp}(\varphi) \geq \sigma$, $\mathsf{conf}(\varphi) \geq \theta$, and $\mathcal{S}^* = \text{argmax}_{|\mathcal{S}|=k} \mathsf{cov}(\mathcal{S})$.

4 COMPUTING TOP-*K* GFCS

Unsurprisingly, the discovery problem for GFCs is intractable [27]. We show that GFC discovery is feasible for large graphs by developing efficient near-optimal algorithms. To this end, we apply rounding techniques to the coverage measure of GFCs to ensure that our problem is approximable. We then develop efficient approximation algorithms that aim to optimize the rounded scores.

4.1 Rounding Selection Criteria

The first technique is to "round up" the significance and coverage scores. The idea is to round up $\text{sig}(\cdot)$ (respectively, $\text{cov}(\cdot)$) to a counterpart $\hat{\text{sig}}(\cdot)$ (respectively, $\hat{\text{cov}}(\cdot)$), which has good properties to ensure feasible pattern discovery, and moreover, the discovery problem becomes approximable.

We next show how to construct such functions.

"Rounded up" significance. The significance score $\text{sig}(\cdot)$ does not preserve the antimonotonicity property under rule refinement. Given a GFC φ, we round up $\text{sig}(\varphi)$ to a function $\hat{\text{sig}}(\varphi)$, which is defined as $\tanh(\max\{\text{sig}(\varphi, p, \delta), \text{sig}(\varphi, \delta, n)\})$, where $\delta > 0$ is a small constant (to prevent the case that $\hat{\text{sig}}(\varphi) = \infty$). $\hat{\text{sig}}(\varphi)$ is normalized in range $[0, 1]$ by the hyperbolic tangent function $\tanh(\cdot)$. We show the following result.

Lemma 4.1 [Antimonotonicity of Rounded Significance]. *Given the graph G, for any* GFCs $\varphi_1 : P_1(x, y) \to r(x, y)$ *and* $\varphi_2 : P_2(x, y) \to r(x, y)$, $\hat{\text{sig}}(\varphi_2) \le \hat{\text{sig}}(\varphi_1)$, *if* $\varphi_1 \preceq \varphi_2$.

Proof. As $\hat{\text{sig}}(\varphi) = \tanh(\max\{\text{sig}(\varphi, p, \delta), \text{sig}(\varphi, \delta, n)\})$, it suffices to show that both $\text{sig}(\varphi, p, \delta)$ and $\text{sig}(\varphi, \delta, n)$ are antimonotonic in terms of rule refinement:

(1) As $\text{sig}(\varphi, p, \delta) = 2|\Gamma^+|(p \ln \frac{p}{\delta} + (1-p) \ln \frac{1-p}{1-\delta})$, the derivative w.r.t. p is

$$\text{sig}'_p(\varphi, p, \delta) = 2|\Gamma^+| \left(\ln \frac{p}{1-p} - \ln \frac{\delta}{1-\delta} \right).$$

In addition, as $\text{sig}(\varphi, \delta, n) = 2|\Gamma^+|(\delta \ln \frac{\delta}{n} + (1-\delta) \ln \frac{1-\delta}{1-n})$, the derivative w.r.t. n is

$$\text{sig}'_n(\varphi, \delta, n) = 2|\Gamma^+| \left(\frac{1-\delta}{1-n} - \frac{\delta}{n} \right) = 2|\Gamma^+| \left(\frac{n-\delta}{n(1-n)} \right).$$

When $\delta \ll \min\{p, n\}$, both $\text{sig}'_p(\varphi, p, \delta) \ge 0$ and $\text{sig}'_n(\varphi, \delta, n) \ge 0$. Hence, both $\text{sig}(\varphi, p, \delta)$ and $\text{sig}(\varphi, \delta, n)$ are monotonic w.r.t. p and n, respectively.

(2) Given Lemma 3.1, we have $p_2 \le p_1$ and $n_2 \le n_1$ if $\varphi_1 \preceq \varphi_2$. Then $\text{sig}(\varphi_2, p_2, \delta) \le \text{sig}(\varphi_1, p_1, \delta)$ and $\text{sig}(\varphi_2, \delta, n_2) \le \text{sig}(\varphi_1, \delta, n_1)$, and thus
$\hat{\text{sig}}(\varphi_2) = \tanh(\max\{\text{sig}(\varphi_2, p_2, \delta), \text{sig}(\varphi_2, \delta, n_2)\}) \le \tanh(\max\{\text{sig}(\varphi_1, p_1, \delta), \text{sig}(\varphi_1, \delta, n_1)\}) = \hat{\text{sig}}(\varphi_1)$.

This completes the proof of Lemma 4.1. □

Rounded coverage score. We next round up $\text{cov}(\cdot)$ with rounded significance. Given a set of GFCs $\mathcal{S}$, the rounded coverage $\hat{\text{cov}}(\mathcal{S})$ is defined as $\hat{\text{sig}}(\mathcal{S}) + \text{div}(\mathcal{S})$, where $\hat{\text{sig}}(\mathcal{S}) = (\sum_{\varphi \in \mathcal{S}} \hat{\text{sig}}(\varphi))^{\frac{1}{2}}$.

The new coverage $\hat{\text{cov}}(\cdot)$ is well defined in terms of submodularity [31], a property widely used to justify goodness measures for set mining. Indeed, adding a new GFC φ to a set $\mathcal{S}$ improves its significance and coverage at least as much as adding it to any superset of $\mathcal{S}$ (diminishing gain to $\mathcal{S}$). Define the marginal gain $\text{mg}(\varphi, \mathcal{S})$ of a GFC φ to $\mathcal{S}$ ($\varphi \notin \mathcal{S}$) as $\hat{\text{cov}}(\mathcal{S} \cup \{\varphi\})$-$\hat{\text{cov}}(\mathcal{S})$. We show the following result.

Lemma 4.2 [Submodularity of Coverage]. *The function* $\hat{\text{cov}}(\cdot)$ *is a monotone submodular function for* GFCs*—that is, (1) for any two sets* $\mathcal{S}_1$ *and* $\mathcal{S}_2$, *if* $\mathcal{S}_1 \subseteq \mathcal{S}_2$, *then* $\hat{\text{cov}}(\mathcal{S}_1) \le \hat{\text{cov}}(\mathcal{S}_2)$, *and (2) for any two sets* $\mathcal{S}_1$ *and* $\mathcal{S}_2$, *if* $\mathcal{S}_1 \subseteq \mathcal{S}_2$ *and for any* GFC $\varphi \notin \mathcal{S}_2$, $\text{mg}(\varphi, \mathcal{S}_2) \le \text{mg}(\varphi, \mathcal{S}_1)$.

Proof. We show that both parts pertaining to $\hat{\text{cov}}(\mathcal{S})$ (i.e., $\hat{\text{sig}}(\mathcal{S})$ and $\text{div}(\mathcal{S})$) are monotone submodular functions w.r.t. $\mathcal{S}$, and therefore $\hat{\text{cov}}(\mathcal{S})$ is a monotone submodular function w.r.t. $\mathcal{S}$:

(1) We show that both $\hat{\text{sig}}(\mathcal{S})$ and $\text{div}(\mathcal{S})$ are monotone functions w.r.t. $\mathcal{S}$. Each term $\hat{\text{sig}}(\varphi)$ is positive, and the sum $\sum_{\varphi \in \mathcal{S}_1} \hat{\text{sig}}(\varphi) \leq \sum_{\varphi \in \mathcal{S}_2} \hat{\text{sig}}(\varphi)$, as every φ in $\mathcal{S}_1$ is also in $\mathcal{S}_2$ for any two sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$ of GFCs. Hence, $\hat{\text{sig}}(\mathcal{S})$ is a monotone function w.r.t. $\mathcal{S}$.

We denote each term $(\sum_{\varphi \in \Phi_t(\mathcal{S})} \text{supp}(\varphi))^{\frac{1}{2}}$ in $\text{div}(\mathcal{S})$ as $T_t(\mathcal{S})$. For each term $T_t(\mathcal{S})$ in $\text{div}(\mathcal{S})$, similarly, $\text{supp}(\varphi)$ is positive and the sum $\sum_{\varphi \in \Phi_t(\mathcal{S}_1)} \text{supp}(\varphi) \leq \sum_{\varphi \in \Phi_t(\mathcal{S}_2)} \text{supp}(\varphi)$, as every φ in $\Phi_t(\mathcal{S}_1)$ covers t is also in $\Phi_t(\mathcal{S}_2)$ for any two sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$ of GFCs. Hence, each term $T_t(\mathcal{S})$ in $\text{div}(\mathcal{S})$ is a monotone function w.r.t. $\mathcal{S}$, and thus $\text{div}(\mathcal{S})$ is a monotone function w.r.t. $\mathcal{S}$.

(2) Next, we show that both $\hat{\text{sig}}(\mathcal{S})$ and $\text{div}(\mathcal{S})$ are submodular functions w.r.t. $\mathcal{S}$. For any GFC $\varphi' \notin \mathcal{S}$, the marginal gain for $\hat{\text{sig}}(\mathcal{S})$ is $\hat{\text{sig}}(\mathcal{S} \cup \{\varphi'\}) - \hat{\text{sig}}(\mathcal{S}) = (\sum_{\varphi \in \mathcal{S} \cup \{\varphi'\}} \hat{\text{sig}}(\varphi))^{\frac{1}{2}} - \hat{\text{sig}}(\mathcal{S}) = (\hat{\text{sig}}^2(\mathcal{S}) + \hat{\text{sig}}(\varphi'))^{\frac{1}{2}} - \hat{\text{sig}}(\mathcal{S}) = \hat{\text{sig}}(\varphi') \big/ ((\hat{\text{sig}}^2(\mathcal{S}) + \hat{\text{sig}}(\varphi'))^{\frac{1}{2}} + \hat{\text{sig}}(\mathcal{S}))$, which is an antimonotonic function w.r.t. $\hat{\text{sig}}(\mathcal{S})$. As $\hat{\text{sig}}(\mathcal{S})$ is monotonic w.r.t. $\mathcal{S}$, for any two sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and $\varphi' \notin \mathcal{S}_2$, $\hat{\text{sig}}(\mathcal{S}_1) \leq \hat{\text{sig}}(\mathcal{S}_2)$. Hence, $\hat{\text{sig}}(\mathcal{S}_2 \cup \{\varphi'\}) - \hat{\text{sig}}(\mathcal{S}_2) \leq \hat{\text{sig}}(\mathcal{S}_1 \cup \{\varphi'\}) - \hat{\text{sig}}(\mathcal{S}_1)$. Therefore, $\hat{\text{sig}}(\mathcal{S})$ is a submodular function w.r.t. $\mathcal{S}$.

Similarly, for any GFC $\varphi' \notin \mathcal{S}$, the marginal gain of $\text{div}(\cdot)$ for each term $T_t(\mathcal{S})$ is $T_t(\mathcal{S} \cup \{\varphi'\}) - T_t(\mathcal{S}) = (\sum_{\varphi \in \Phi_t(\mathcal{S} \cup \{\varphi'\})} \text{supp}(\varphi))^{\frac{1}{2}} - T_t(\mathcal{S})$. If φ' does not cover t, then $T_t(\mathcal{S} \cup \{\varphi'\}) - T_t(\mathcal{S}) = 0$. Otherwise, if φ' covers t, following the similar process for $\text{sig}(\mathcal{S})$, we have $T_t(\mathcal{S} \cup \{\varphi'\}) - T_t(\mathcal{S}) = \text{supp}(\varphi') \big/ ((T_t^2(\mathcal{S}) + \text{supp}(\varphi'))^{\frac{1}{2}} + T_t(\mathcal{S}))$, which is an antimonotonic function w.r.t. $T_t(\mathcal{S})$. As $T_t(\mathcal{S})$ is monotonic w.r.t. $\mathcal{S}$, for any two sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and $\varphi' \notin \mathcal{S}_2$, $T_t(\mathcal{S}_1) \leq T_t(\mathcal{S}_2)$. Hence, $T_t(\mathcal{S}_2 \cup \{\varphi'\}) - T_t(\mathcal{S}_2) \leq T_t(\mathcal{S}_1 \cup \{\varphi'\}) - T_t(\mathcal{S}_1)$, no matter whether φ' covers t. Thus, each term $T_t(\mathcal{S})$ in $\text{div}(\mathcal{S})$ is a submodular function w.r.t. $\mathcal{S}$, and $\text{div}(\mathcal{S})$ is a submodular function w.r.t. $\mathcal{S}$.

In summary, both $\hat{\text{sig}}(\mathcal{S})$ and $\text{div}(\mathcal{S})$ are monotone submodular functions w.r.t. $\mathcal{S}$, and therefore $\hat{\text{cov}}(\mathcal{S})$ is a monotone submodular function w.r.t. $\mathcal{S}$. Lemma 4.2 thus follows. □

Keeping all other constraints (e.g., support and confidence thresholds) intact, the problem of GFC discovery with rounded coverage is to compute top-kGFCs $\mathcal{S}^*$ that maximizes $\hat{\text{cov}}(\mathcal{S})$. We next show that there are efficient approximations for GFC discovery with the rounded coverage.

4.2 Stream-Based Rule Discovery

An "enumeration-and-verify" algorithm iterates and verifies all k-subsets of GFCs that cover some facts in Γ. This is clearly impractical for large G and Γ. We consider more efficient algorithms.

"Batch + Greedy." We start with an algorithm, denoted as GFC_batch, which performs a batch pattern discovery, following a greedy selection process as follows. First, apply a standard graph pattern mining process (e.g., Apriori [21]) to generate and verify all the graph patterns $\mathcal{P}$. The verification is conducted by an operator Verify to compute the support and confidence for each pattern. Second, invoke a greedy algorithm to do k passes of $\mathcal{P}$. In each iteration i, it selects the pattern P_i such that the corresponding GFC $\varphi_i : P_i(x,y) \rightarrow r(x,y)$ maximizes the marginal gain $\hat{\text{cov}}(\mathcal{S}_{i-1} \cup \{\varphi_i\})$ - $\hat{\text{cov}}(\mathcal{S}_{i-1})$, then it updates $\mathcal{S}_i$ as $\mathcal{S}_{i-1} \cup \{\varphi_i\}$.

Verification. Given a graph pattern $P(x,y)$ and a graph $G = (V, E, L)$, for each pattern node u in $P(x,y)$, the operator Verify first initializes a set of match sets $V(u) = \{v | L(v) \sim_\alpha L(u), v \in V\}$, determined by a given label similarity predicate. It then invokes approximate pattern matching algorithms [29, 43] to iteratively refine the match set $V(u)$ by removing nodes that cannot satisfy the parent-child relation of pattern node u, with ranging u over all pattern nodes in P, until no nodes can be removed from any match set. The support and confidence are then computed by

```
Algorithm GFC_stream
Input:  Graph $G$, training facts $\Gamma$, support threshold $\sigma$,
        confidence threshold $\theta$, integer $k$, triple pattern $r(x, y)$.
Output: Top-$k$ GFCs $\mathcal{S}$ pertaining to $r(x, y)$.
1.  set $\mathcal{S} := \emptyset$; set $\mathcal{P} := \emptyset$; maxpcov := 0;
2.  graph pattern $P :=$ PGen($G, \Gamma, \mathcal{P}, \sigma, \theta$).next();
3.  while $P \neq$ null do
4.      if $|E_P| = 1$ and maxpcov < $\hat{\text{cov}}(P)$ then
5.          maxpcov := $\hat{\text{cov}}(P)$;
6.      if $|E_P| \geq 2$ then
7.          $\mathcal{S} :=$ PSel($P, \mathcal{S}, \mathcal{P}, r(x, y)$, maxpcov);
8.      pattern $P :=$ PGen($G, \Gamma, \mathcal{P}, \sigma, \theta$).next();
9.  return $\mathcal{S}$;
```

Fig. 2. Algorithm GFC_stream

aggregating the match sets. It is known that it takes $O(|V_P| + |V_m|)(|E_P| + |E'|)$ time to verify a pattern, where V_m is the largest initial match set and $E' \subseteq E$ refers to the edges induced by all the initial node matches of $P(x, y)$.

The algorithm GFC_batch is a $(1 - \frac{1}{e})$ approximation, following Lemma 4.2 and the seminal result in Nemhauser et al. [31] on the optimization problem of submodular functions. Nevertheless, it requires verification of all patterns before the construction of GFCs. The selection further requires k passes of all the verified patterns. This is expensive for large G and Γ.

“Stream + Sieve.” We can do better by capitalizing on stream-based optimization [1, 43]. In contrast to “batch”-style mining, we organize newly generated patterns in a stream and assemble new patterns to top-k GFCs with small update costs. This requires a single scan of all patterns without waiting for all patterns to be verified. We develop such an algorithm to discover GFCs with optimality guarantees, as verified by the following result.

THEOREM 4.3. *Given a constant $\epsilon > 0$, there exists an algorithm that computes top-k* GFCs *with rounded coverage, and*

(1) *it achieves an approximation ratio $(\frac{1}{2} - \epsilon)$;*
(2) *it performs a single pass of patterns, with update cost in $O((b + |\Gamma_b|)^2 + \frac{\log k}{\epsilon})$, where b is the largest edge number of a pattern, and Γ_b is the b hop neighbors of the entities occurred in Γ; and*
(3) *it early terminates without visiting the entire pattern set.*

As a proof of Theorem 4.3, we next introduce such a stream-based discovery algorithm. Our discovery algorithm, denoted as GFC_stream (illustrated in Figure 2), interleaves supervised pattern generation and GFC selection as follows:

(1) *Pattern stream generation.* The algorithm GFC_stream invokes a procedure PGen to produce and update a pattern stream $\mathcal{P}$ (lines 2 and 8). In contrast to GFC_batch that verifies all patterns against the entire graph G, it partitions facts Γ to blocks, and iteratively spawns and verifies patterns by visiting local neighbors of the facts in each block. This progressively finds patterns that better “purify” the labels of only those facts they cover and thus reduces unnecessary verification.

(2) *Selection on-the-fly.* GFC_stream invokes a procedure PSel (line 7) to select patterns and construct GFCs on-the-fly. To achieve the optimality guarantee, it applies the stream-sieving strategy in stream data summarization [1]. In a nutshell, it estimates the optimal value of a monotonic submodular function $F(\cdot)$ with multiple "sieve values," which is initialized by the maximum coverage score of single patterns (Section 3), denoted as maxpcov= $\max_{P \in \mathcal{P}}(\hat{\text{cov}}(P))$ (lines 4 and 5), and it eagerly constructs GFCs with high marginal benefits that refine sieve values progressively.

The preceding two procedures interact with each other: each pattern verified by PGen is sent to PSel for selection. The algorithm terminates when no new pattern can be verified by PGen or the set $\mathcal{S}$ can be no longer improved by PSel (as will be discussed). We next introduce the details of the procedures PGen and PSel.

Procedure PGen. Procedure PGen improves its "batch" counterpart in GFC_batch by locally generating patterns that cover particular facts, following the construction of decision trees. We refer to the pattern size as the number of pattern edges. PGen maintains the following structures in each iteration i: (1) a pattern set $\mathcal{P}_i$, which contains graph patterns of size i ($|E_P| = i$) and is initialized to contain a single size-0 ($|E_P| = 0$) pattern with two independent anchored nodes u_x and u_y only, and (2) a partition set $\Gamma_i(P)$, which records the sets of facts $P(\Gamma^+)$ and $P(\Gamma^-)$ and is initialized as $\{\Gamma^+, \Gamma^-\}$, for each pattern $P \in \mathcal{P}_i$. Each iteration i performs the following:

(1) For each block $B \in \Gamma_{i-1}$, PGen generates a set of patterns $\mathcal{P}_i$ with size i. Each pattern P in $\mathcal{P}_i$ is constructed by adding a triple pattern $r'(u, u')$ to its size-(i-1) counterpart P' ($|E_{P'}| = i$-1) in $\mathcal{P}_{i-1}$. Moreover, it only inserts $r'(u, u')$ with instances from the neighborhood of matched nodes of P', bounded by $P'(\Gamma)$.
(2) For each pattern $P \in \mathcal{P}_i$, PGen computes its support, confidence, and significance (G-test) as in the procedure Verify of the GFC_batch algorithm and prunes $\mathcal{P}_i$ by removing unsatisfied patterns. It refines $P'(\Gamma^+)$ and $P'(\Gamma^-)$ to $P(\Gamma^+)$ and $P(\Gamma^-)$ accordingly. Note that $P(\Gamma^+) \subseteq P'(\Gamma^+)$, and $P(\Gamma^-) \subseteq P'(\Gamma^-)$. Once a promising pattern P is verified, PGen returns P to the procedure PSel for the construction of top-k GFCs $\mathcal{S}^*$.

Procedure PSel. To compute the size-k set of GFCs $\mathcal{S}^*$ that maximizes $\hat{\text{cov}}(\mathcal{S})$ for a given $r(x, y)$, it suffices for the procedure PSel to compute top-k graph patterns that maximize $\hat{\text{cov}}(\mathcal{S})$ accordingly. It solves a submodular optimization problem over the pattern stream that specializes the sieve-streaming technique [1] to GFCs.

Sieve streaming [1]. Given a monotone submodular function $F(\mathcal{S})$, a constant $\epsilon > 0$, and an element set $\mathcal{D}$, sieve streaming finds top-k elements $\mathcal{S}^*$ that maximize $F(\mathcal{S})$ as follows. It first finds the largest value of singleton sets $m = \max_{e \in \mathcal{D}} F(\{e\})$ and then uses a set of sieve values $(1+\epsilon)^j$ (j is an integer) to discretize the range $[m, k * m]$. As the optimal value, denoted as $F(\mathcal{S}^*)$, is in $[m, k * m]$, there exists a value $(1+\epsilon)^j$ that "best" approximates $F(\mathcal{S}^*)$. For each sieve value v, a set of top patterns $\mathcal{S}_v$ is maintained by adding patterns with a marginal gain at least $(\frac{v}{2} - F(\mathcal{S}_v))/(k - |\mathcal{S}_v|)$. It is shown that selecting the sieve of the best k elements produces a set $\mathcal{S}$ with $F(\mathcal{S}) \geq (\frac{1}{2} - \epsilon)F(\mathcal{S}^*)$ [1].

A direct application of the preceding sieve streaming for GFCs seems still infeasible: one needs to find the maximum $\hat{\text{cov}}(\varphi)$, which requires verifying the entire pattern set. For a fixed $r(x, y)$, this is equivalent to finding the pattern $P(x, y)$ that maximizes $\hat{\text{cov}}(P)$, where $\hat{\text{cov}}(P)$ is computed as the coverage of the singleton set $\hat{\text{cov}}(\{\varphi\})$, and $\varphi = P(x, y) \to r(x, y)$. We use $\hat{\text{cov}}(\varphi)$ and $\hat{\text{cov}}(P)$ interchangeably when the context is clear.

```
Procedure PSel($P, \mathcal{S}, \mathcal{P}, r(x,y)$, maxpcov)
1.  if $\mathcal{P} = \emptyset$ then
2.      set $\mathcal{S}_V := \{(1+\epsilon)^j | (1+\epsilon)^j \in [\text{maxpcov}, k * \text{maxpcov}]\}$;
3.      for each $v_j \in \mathcal{S}_V$ do $\mathcal{S}_{v_j} := \emptyset$;
4.  for each $v_j \in \mathcal{S}_V$ do
5.      if $\text{mg}(P, \mathcal{S}_{v_j}) \geq (\frac{v_j}{2} - \hat{\text{cov}}(\mathcal{S}_{v_j}))/(k - |\mathcal{S}_{v_j}|)$ and $|\mathcal{S}_{v_j}| < k$ then
6.          $\mathcal{S}_{v_j} := \mathcal{S}_{v_j} \cup \{P\}$;
7.  if all sets $\mathcal{S}_{v_j}$ have size $k$ then
8.      $\mathcal{S}$ := constructGFCs($\mathcal{S}_{v_j}$) ($j \in [1, \log_{(1+\epsilon)}(k * \text{maxpcov})]$);
9.  return $\mathcal{S}$;
```

Fig. 3. Procedure PSel

Capitalizing on data locality of graph pattern matching, as well as Lemma 4.2, and Lemma 3.1 for rounded measures of GFCs, the maximum $\hat{\text{cov}}(P)$ can be computed efficiently without enumerating the entire pattern set in advance, as verified by the following result.

LEMMA 4.4. *It is in* $O(|\Gamma_1|)$ *time to compute the maximum* $\hat{\text{cov}}(P)$.

This can be verified by observing that $\hat{\text{cov}}(P)$ also preserves antimonotonicity in terms of pattern refinement—that is, $\hat{\text{cov}}(P') \leq \hat{\text{cov}}(P)$ if P is a subgraph of P'. Thus, $\max_{P \in \mathcal{P}} \hat{\text{cov}}(P)$ is contributed by single-edge patterns. In other words, procedure PSel only needs to cache at most $|\Gamma_1|$ size-1 patterns ($|E_P| = 1$) from PGen to find the global maximum $\hat{\text{cov}}(P)$ (lines 4 and 5 of GFC_stream in Figure 2). The rest of PSel follows the sieve-streaming strategy, as illustrated in Figure 3. The GFCs are constructed with the top-k graph patterns (line 8).

Optimization. To further prune unpromising patterns, procedure PGen estimates an upper bound $\hat{\text{mg}}(P, \mathcal{S}_{v_j})$ (line 5 of PSel) without verifying a new size-i pattern P ($|E_P| = i$). If $\hat{\text{mg}}(P, \mathcal{S}_{v_j}) < (\frac{v_j}{2} - \hat{\text{cov}}(\mathcal{S}_{v_j}))\Big/(k - |\mathcal{S}_{v_j}|)$, P is skipped without further verification.

To this end, PGen first traces to a GFC $\varphi' : P'(x,y) \rightarrow r(x,y)$, where P' is a verified subpattern of P (i.e., $P' \preceq P$), and P is obtained by adding a triple pattern r' to P'. It estimates an upper bound of the support of the GFC $\varphi : P(x,y) \rightarrow r(x,y)$ as $\hat{\text{supp}}(\varphi) = \text{supp}(\varphi')$-$\frac{l}{|r(\Gamma^+)|}$, where l is the number of the facts in $r(\Gamma^+)$ that have no match of r' in their i hop neighbors (and thus cannot be covered by P). Similarly, one can estimate an upper bound for p and n in $\hat{\text{sig}}(\varphi)$ and thus get an upper bound $\hat{\text{sig}}_i(\varphi)$ for $\hat{\text{sig}}(\varphi)$. For each t in Γ^+, denote term $(\sum_{\varphi \in \Phi_t(\mathcal{S})} \text{supp}(\varphi))^{\frac{1}{2}}$ in $\text{div}(\mathcal{S})$ as $T_t(\mathcal{S})$; it then computes $\hat{\text{mg}}(P, \mathcal{S})$ as

$$\hat{\text{mg}}(P, \mathcal{S}) = \frac{\hat{\text{sig}}_i(\varphi)}{2\text{sig}(\mathcal{S})} + \left(\sum_{t \in P(\Gamma^+)} \frac{\hat{\text{supp}}(\varphi)}{2T_t(\mathcal{S})} \right) \Big/ |\Gamma^+|.$$

To see that $\hat{\text{mg}}(P, \mathcal{S})$ is an upper bound for $\text{mg}(P, \mathcal{S})$, one may note that the marginal gains for the significance part $\hat{\text{sig}}(\mathcal{S})$ and the diversity part $\text{div}(\mathcal{S})$ are both defined in terms of square roots. Given any two positive numbers a_1 and a_2, an upper bound of $\sqrt{a_1 + a_2} - \sqrt{a_1}$ is $\frac{a_2}{2\sqrt{a_1}}$. We apply this inequality to each square root term in $\hat{\text{cov}}(\mathcal{S})$. Taking significance, for example, $\text{sig}(\mathcal{S} \cup \{\varphi\}) - \text{sig}(\mathcal{S}) \leq \sqrt{\text{sig}^2(\mathcal{S}) + \hat{\text{sig}}_i(\varphi)} - \sqrt{\text{sig}^2(\mathcal{S})}$. When substitute a_1 and a_2 in the inequality by $\text{sig}^2(\mathcal{S})$ and $\hat{\text{sig}}_i(\varphi)$, respectively, we can have the upper bound $\frac{\hat{\text{sig}}_i(\varphi)}{2\text{sig}(\mathcal{S})}$. Similarly, for other square root terms in $\text{div}(\mathcal{S})$, one can apply the inequality to obtain their upper bounds. We found that using upper bound estimation effectively reduces redundant verifications.

Performance analysis. Denoting the total patterns verified by the algorithm GFC_stream as $\mathcal{P}$, it takes $O(|\mathcal{P}|(b + |\Gamma_b|)^2)$ time to compute the pattern matches and verify the patterns. Each time a pattern is verified, it takes $O(\frac{\log k}{\epsilon})$ time to update the sets $\mathcal{S}_v$ for sieve values in $\mathcal{S}_V$. Thus, the update time for each pattern is in $O((b + |\Gamma|_b)^2 + \frac{\log k}{\epsilon})$.

Optimality. The approximation ratio follows the analysis of sieve-stream summarization in Badanidiyuru et al. [1]. Specifically, (1) there exists a sieve value $v_j = (1 + \epsilon)^j \in [\text{maxpcov}, k * \text{maxpcov}]$ that is closest to $F(\mathcal{S}^*)$, say, $(1 - 2\epsilon)F(\mathcal{S}^*) \leq v_j \leq F(\mathcal{S}^*)$, and (2) the set $\mathcal{S}_{v_j}$ with v_j is a $(\frac{1}{2} - \epsilon)$ answer for an estimation of $F(\mathcal{S}^*)$. Indeed, if $\text{mg}(P, \mathcal{S}_{v_j})$ satisfies the test in PSel (line 5), $\hat{\text{cov}}(\mathcal{S}_{v_j})$ is at least $\frac{v_j|\mathcal{S}_{v_j}|}{2k} \geq \frac{v_j}{2}$ (when $|\mathcal{S}_{v_j}| = k$) $\geq (\frac{1}{2} - \epsilon)F(\mathcal{S}^*)$. Following Badanidiyuru et al. [1], the value $v_j \in \mathcal{S}_V$ best estimates the optimal $\hat{\text{cov}}(\mathcal{S}^*)$ and achieves $(\frac{1}{2} - \epsilon)\hat{\text{cov}}(\mathcal{S}^*)$. Thus, selecting the GFCs constructed from patterns in the sieve $\mathcal{S}$ with the largest coverage $\hat{\text{cov}}(\mathcal{S})$ guarantees the approximation ratio $(\frac{1}{2} - \epsilon)$.

4.3 Discovering GFCs With Dynamic Spawning

The stream-based algorithm GFC_stream has tunable time cost and quality guarantees, which makes it feasible for large graphs. However, the space cost can be large when a set of size-k sieve sets are created for a small ϵ. We next introduce a space-efficient alternative of GFC_stream. The algorithm preserves the optimality of its batch counterpart GFC_batch, as verified in the following.

Theorem 4.5. *There is a* $(1 - \frac{1}{e})$ *approximation algorithm to compute top-k* GFCs *with rounded coverage, in (1)* $O(\min\{k, b\}k\mathcal{T}(b + |\Gamma_b|)^2)$ *time, where* $\mathcal{T}$ *is the total distinct triple patterns with nonempty instances in G and (2) has space cost in* $O(\min\{k, b\}k\mathcal{T} + |\Gamma_b|)$.

We first show that the marginal gain of GFCs also ensures the antimonotonicity property under rule refinement, as verified by the following result.

Lemma 4.6 [Antimonotonicity of Marginal Gains]. *Given the graph G, for any set of* GFCs $\mathcal{S}$*, and two* GFCs φ_1*:* $P_1(x, y) \rightarrow r(x, y)$ *and* φ_2*:* $P_2(x, y) \rightarrow r(x, y)$*, if* $\varphi_1 \preceq \varphi_2$ *and both* φ_1 *and* φ_2 *are not in* $\mathcal{S}$*, then* $\text{mg}(\varphi_2, \mathcal{S}) \leq \text{mg}(\varphi_1, \mathcal{S})$.

Proof. Define two marginal gains $\text{mg}_s(\varphi, \mathcal{S}) = \hat{\text{sig}}(\mathcal{S} \cup \{\varphi\})$ - $\hat{\text{sig}}(\mathcal{S})$ and $\text{mg}_d(\varphi, \mathcal{S}) = \text{div}(\mathcal{S} \cup \{\varphi\})$ - $\text{div}(\mathcal{S})$. As $\text{mg}(\varphi, \mathcal{S}) = \text{mg}_s(\varphi, \mathcal{S}) + \text{mg}_d(\varphi, \mathcal{S})$, it suffices to show that when $\varphi_1 \preceq \varphi_2$, (1) $\text{mg}_s(\varphi_2, \mathcal{S}) \leq \text{mg}_s(\varphi_1, \mathcal{S})$ and (2) $\text{mg}_d(\varphi_2, \mathcal{S}) \leq \text{mg}_d(\varphi_1, \mathcal{S})$. We next prove both marginal gains preserve antimonotonicity:

(1) As $\varphi_1 \preceq \varphi_2$, P_1 is a subgraph of P_2. Given Lemma 4.1, $\hat{\text{sig}}(\varphi_2) \leq \hat{\text{sig}}(\varphi_1)$. Thus, $\text{mg}_s(\varphi_2, \mathcal{S}) = (\hat{\text{sig}}^2(\mathcal{S}) + \hat{\text{sig}}(\varphi_2))^{\frac{1}{2}} - \hat{\text{sig}}(\mathcal{S}) \leq (\hat{\text{sig}}^2(\mathcal{S}) + \hat{\text{sig}}(\varphi_1))^{\frac{1}{2}} - \hat{\text{sig}}(\mathcal{S}) = \text{mg}_s(\varphi_1, \mathcal{S})$.

(2) Given Lemma 3.1, $\text{supp}(\varphi_2) \leq \text{supp}(\varphi_1)$, and $P_2(\Gamma^+) \subseteq P_1(\Gamma^+)$. Define the term $\text{mg}(\varphi, \mathcal{S}, t)$ as $(\sum_{\varphi' \in \Phi_t(\mathcal{S} \cup \{\varphi\})} \text{supp}(\varphi))^{\frac{1}{2}} - (\sum_{\varphi' \in \Phi_t(\mathcal{S})} \text{supp}(\varphi))^{\frac{1}{2}}$. Clearly, $\text{mg}(\varphi, \mathcal{S}, t) = 0$ if φ does not cover t. Then, $\text{mg}(\varphi_1, \mathcal{S}, t) - \text{mg}(\varphi_2, \mathcal{S}, t)$ is 0 if $t \notin P_1(\Gamma^+)$ (i.e., none of φ_1 and φ_2 covers t); is greater than 0 if $t \in P_1(\Gamma^+) \setminus P_2(\Gamma^+)$ (i.e., t is covered by P_1 but not P_2), as $\text{mg}(P_2, \mathcal{S}, t)$ is 0; and is $(\sum_{\varphi \in \Phi_t(\mathcal{S} \cup \{\varphi_1\})} \text{supp}(\varphi))^{\frac{1}{2}} - (\sum_{\varphi \in \Phi_t(\mathcal{S} \cup \{\varphi_2\})} \text{supp}(\varphi))^{\frac{1}{2}} \geq 0$, as $\text{supp}(\varphi_2) \leq \text{supp}(\varphi_1)$. Thus, $\text{mg}(\varphi_2, \mathcal{S}, t) \leq \text{mg}(\varphi_1, \mathcal{S}, t)$ for any fact $t \in \Gamma^+$. Summing over each fact $t \in \Gamma^+$, $\text{mg}_d(\varphi_2, \mathcal{S}) = \frac{1}{|\Gamma^+|} \sum_{t \in \Gamma^+} \text{mg}(\varphi_2, \mathcal{S}, t) \leq \frac{1}{|\Gamma^+|} \sum_{t \in \Gamma^+} \text{mg}(\varphi_1, \mathcal{S}, t) = \text{mg}_d(\varphi_1, \mathcal{S})$.

Putting (1) and (2) together, Lemma 4.6 follows. □

```
Algorithm GFC_spawn
Input:  Graph $G$, training facts $\Gamma$, support threshold $\sigma$,
        confidence threshold $\theta$, integer $k$, triple pattern $r(x, y)$.
Output: Top-$k$ GFCs $\mathcal{S}$ pertaining to $r(x, y)$.
1.  set $\mathcal{S} := \emptyset$; set $\mathcal{P}' := \{\text{pattern P} : |E_P| = 1\}$;
2.  set $\mathcal{P} := \{\text{pattern } P^* : P^* := \text{argmax}_{|E_P|=1}\, \hat{\text{cov}}(P)\}$;
3.  $\mathcal{S} := \mathcal{S} \cup \{\text{constructGFCs}(P^*)\}$;
4.  $\mathcal{P}' := \mathcal{P}' \setminus \{P^*\}$;
5.  while $|\mathcal{S}| \neq k$ do
6.      $\mathcal{P}' := \mathcal{P}' \cup \text{Spawn}(P^*, r(x, y))$;
7.      $P^* := \text{argmax}_{P \in \mathcal{P}'}\, \text{mg}(P, \mathcal{S})$;
8.      $\mathcal{P} := \mathcal{P} \cup \{P^*\}$; $\mathcal{P}' := \mathcal{P}' \setminus \{P^*\}$;
9.      $\mathcal{S} := \mathcal{S} \cup \{\text{constructGFCs}(P^*)\}$;
10. return $\mathcal{S}$;
```

Fig. 4. Algorithm GFC_spawn

We next introduce an approximation algorithm with the performance guarantee in Theorem 4.5. Instead of verifying all patterns, it utilizes the antimonotonicity of $\text{mg}(\cdot)$ to dynamically spawn and select the most promising patterns.

Algorithm. The algorithm, denoted as GFC_spawn and shown in Figure 4, maintains two sets of patterns: set $\mathcal{P}$ to be used to construct top-k GFCs and set $\mathcal{P}'$ for verified patterns. It uses P^* to mark the best pattern chosen to be spawned from, initialized as the single-edge pattern with maximized rounded coverage score (line 2). The set $\mathcal{S}$ is also initialized as a single GFC constructed with P^* (line 3). The algorithm next spawns and verifies patterns from P^* and greedily selects a new P^* as the pattern from the verified patterns so far (maintained in $\mathcal{P}'$), which maximizes marginal gain to set $\mathcal{S}$ (lines 6 and 7). Each time a new pattern P^* is identified, a GFC is constructed and added to $\mathcal{S}$ (line 9). The process returns $\mathcal{S}$ once k GFCs are constructed.

Given a pattern P^*, the procedure Spawn dynamically spawns new patterns to be verified. It first generates a set of patterns by adding single triple patterns to P^*. It then adopts a "lazy verification" to reduce unnecessary verifications. For each newly generated pattern P', if there is a subpattern P'' in $\mathcal{P}'$ (i.e., $P'' \preceq P'$), Spawn skips the verification of P'. Indeed, such patterns P' contribute no more marginal gain than their subpatterns, given Lemma 4.6.

Analysis. Denote $\mathcal{T}$ as the number of distinct triple patterns with nonempty instances in G. For time complexity, observe that GFC_spawn spawns at most k times and verifies at most $|V_{P^*}|\mathcal{T}$ patterns in each iteration. First, if $b > k$, GFC_spawn verifies $O(1\mathcal{T} + 2\mathcal{T} + \cdots + k\mathcal{T}) = O(k^2\mathcal{T})$ patterns. Second, if $b \leq k$, it spawns patterns with size no larger than b only in k iterations. In this case, GFC_spawn verifies $O(b^2\mathcal{T} + (k-b)(b-1)\mathcal{T}) = O(kb\mathcal{T})$ patterns. It thus takes in total $O(\min\{k, b\}k\mathcal{T}(b + |\Gamma_b|)^2)$ time to verify the patterns. The space cost is in $O(\min\{k, b\}k\mathcal{T} + |\Gamma_b|)$, as GFC_spawn only needs to store $\min\{k, b\}k\mathcal{T}$ spawned patterns from the best pattern P^* in each iteration and incurs a once-for-all cost of $|\Gamma_b|$ to store all the candidates of these patterns.

For optimality, observe that GFC_spawn preserves the invariant that at any step, the selected pattern P^* has the maximum marginal gain $\text{mg}(P, \mathcal{S})$ among *all* unvisited patterns, given the antimonotonicity of marginal gains (Lemma 4.6) and the spawning operator Spawn. As $\hat{\text{cov}}(\cdot)$ is a monotone submodular function w.r.t. $\mathcal{S}$, a k round of greedy selection of patterns that maximize dynamically updated marginal gain in each round guarantees a $(1 - \frac{1}{e})$ approximation ratio.

The preceding analysis completes the proof of Theorem 4.5.

Table 1. Overview of real-world graph datasets

Dataset	Category	Entities (#)	Triples (#)	Entity Labels (#)	Relations (#)	Triple Patterns (#)
YAGO	Knowledge base	$2.1M$	$4.0M$	2,273	33	$15.5K$
DBpedia	Knowledge base	$2.2M$	$7.4M$	73	584	8,240
Wikidata	Knowledge base	$10.8M$	$41.4M$	18,383	693	$209K$
MAG	Academic network	$0.6M$	$1.71M$	8,565	6	11,742
Offshore	Social network	$1.0M$	$3.3M$	356	274	633
GDELT	Event network	$13K$	$66M$	34	255	$78K$

5 GFC-BASED FACT CHECKING

The GFCs can be applied to enhance fact checking as rule models or via supervised link prediction. We introduce two GFC-based models.

Generating training facts. Given a knowledge graph $G = (V, E, L)$ and a triple pattern $r(x, y)$, we generate training facts Γ as follows. First, for each fixed $r(x, y)$, a set of true facts Γ^+ are sampled from the matches of $r(x, y)$ in the knowledge graph G. For each true fact $<v_x, r, v_y> \in \Gamma^+$, we further introduce "noise" by replacing their labels to similar counterparts asserted, for example, by synonyms or acronyms. This generates a set of true facts that approximately match $r(x, y)$. Second, given Γ^+, a set of false facts Γ^- are sampled under WPCA (Section 3). For a missing fact $t = <v_x, r, v'_y> \in (V \times V) \setminus E$, if there exists a true fact $<v_x, r, v_y>$ in Γ^+ and $L(v_y) \not\approx_\alpha L(v'_y)$, then t is added to Γ^- as a false example. Note that $\Gamma^+ \cap \Gamma^- = \emptyset$.

Using GFCs **as rules.** Given facts Γ, a rule-based model, denoted as GFact$_R$, invokes algorithm GFC_stream to discover top-k GFCs $\mathcal{S}$ as fact checking rules. Given a new fact $t = <v_x, r, v_y>$, it follows the "hit and miss" convention [15] to check if there exists a GFC φ in $\mathcal{S}$ that covers t (i.e., both its consequent and antecedent cover t). If so, GFact$_R$ accepts t; otherwise, it rejects t.

Using GFCs **in supervised link prediction.** Useful instance-level features can be extracted from the patterns and their matches induced by GFCs to train classifiers. We develop a second model (denoted as GFact) that adopts the following specifications.

Features. For each example $t = <v_x, r, v_y> \in \Gamma$, GFact constructs a feature vector of size k, where each entry encodes the presence of the ith GFC φ_i in the top-k GFCs $\mathcal{S}$. The class label of the example t is *true* (respectively, *false*) if $t \in \Gamma^+$ (respectively, Γ^-).

By default, GFact adopts logistic regression, which is experimentally verified to achieve slightly better performance than others (e.g., naive Bayes and support vector machines). As verified by our experimental study (Section 6), GFact outperforms GFact$_R$ in accuracy for most cases over real-world graphs, with some additional learning cost that exploits features induced by patterns and its matches.

6 EXPERIMENTAL STUDY

Using real-world knowledge bases, we conduct three sets of experiments to evaluate (1) the efficiency of GFCs discovery for fact checking in large knowledge graphs, (2) the effectiveness of GFC-based fact checking, and (3) applications of GFCs for fact checking in knowledge bases and Web news.

6.1 Experimental Settings

Datasets. We used six real-world graph datasets (Table 1), including (1) YAGO [44] (version 2.5), a knowledge base that contains $2.1M$ entities with $2,273$ distinct labels, $4.0M$ edges with 33 distinct labels, and 15.5K triple patterns; (2) DBpedia (or simply DBP) [25] (version 3.8), a knowledge base that contains 2.2M entities with 73 distinct labels, 7.4M edges with 584 distinct labels, and 8.2K triple patterns; (3) Wikidata (or simply WIKI) [47] (RDF dumps 20,160,801), a knowledge base that contains 10.8M entities with 18, 383 labels, 41.4M edges of 693 labels, and 209K triple patterns; (4) MAG [41], a fraction of an academic graph with 0.6M entities (e.g., papers, authors, venues, and affiliations) of 8,565 labels and 1.71M edges of 6 labels (cite, coauthorship, etc.); (5) Offshore [19], a social network of offshore entities and financial activities, which contains 1M entities (company, country, person, etc) with 356 labels, 3.3M relationships (establish, close, etc) with 274 labels, and 633 triple patterns; and (6) GDELT [24], a dense news network crawled online. We dump GDELT Event 1.0 data in the year of 2017, which has 13K entities of 34 different labels (e.g., country, business, and school), 66M edges of 255 relationships (e.g., consult), and 78K triple patterns.

Algorithms. We implemented the following methods, all in Java.

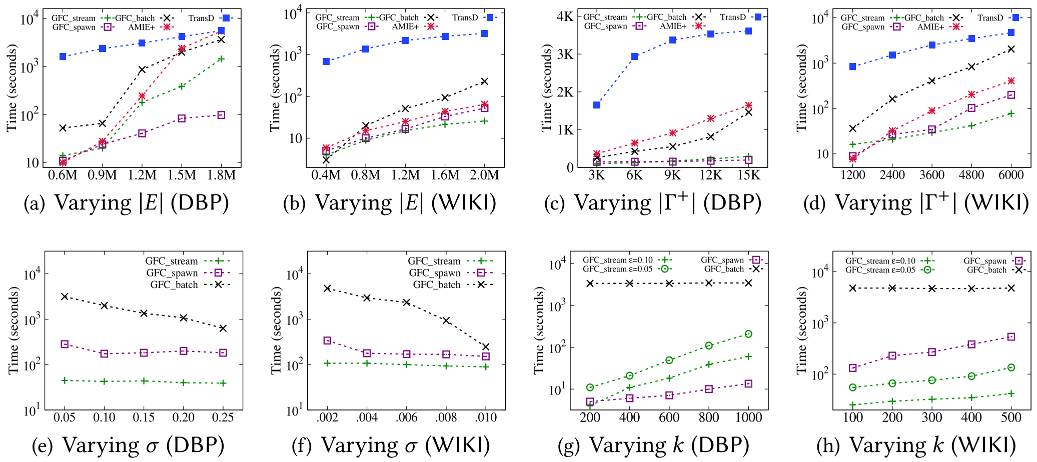
GFC-*based techniques*. We implemented the following: (1) GFC_stream and GFC_spawn that discover GFCs by sieve streaming and dynamic spawning, respectively, compared to their "Batch + Greedy" counterpart GFC_batch (Section 4), and (2) the two GFC-based learning algorithms GFact, and the rule-based counterpart GFact$_R$ (Section 5).

Fact checking models. In addition, we compare GFC-based fact checking to the following models. AMIE+ [14, 15] discovers Horn-clause rules for fact prediction; PRA [23], the well-established path ranking algorithm, trains classifiers with path features from random walks; SFE [16], a graph-based method, learns a classifier from subgraphs extracted around the training facts by sampling paths with replacing similar node and edge labels as features; (d) KGMiner [39], a shortest path–based method, seeks the discriminative predicate paths as features that "describe" the targeted facts; and (e) TransD [20], an embedding-based method, builds a mapping matrix for both entities and edges as a unified model. Among these, PRA, SFE, KGMiner, and GFact are supervised link prediction models, whereas AMIE+ and GFact$_R$ are rule-based models.

Model configuration. For a fair comparison, we have made an effort to calibrate the models and training/testing sets with consistent settings. First, we sample 80% of the facts in a knowledge graph as the training facts Γ and 20% of the facts as the testing set T. In Γ (respectively, T), 20% are true examples Γ^+ (respectively, T^+), and 80% are false examples Γ^- (respectively, T^-). We maintain this ratio empirically because real-world knowledge graphs are often sparse and true facts are far fewer than false facts. We generate Γ^- and T^- under WPCA (Section 3) for all the models. We remove the testing facts in T^+ from the data graph G. Taking DBpedia, for example, we sample 107 triple patterns, and each triple pattern has around 5 to 50 K matched instances. We use logistic regression to train the classifiers, which is the default setting of PRA, SFE, and KGMiner. Second, for rule-based methods GFact$_R$ and AMIE+, we discover rules that cover the same set of Γ^+. We set the size of AMIE+ rule body to be at most 3, the same as the pattern size bound (number of edges) for GFC rules. Third, we adopt the weighted path length as the label similarity predicate [52], which is the reciprocal of the distance between two node labels on the ontology, weighted by the counting frequencies of node labels occurred in the data graph. An ontology is usually a graph consisting of labels and their relationships, such as similarTo(*book*, *speech*) [52]. By default, we set threshold α as 0.75 unless otherwise specified.

Measures. We use the following metrics.

Accuracy. Given the test set T and a fact checking model M, we evaluate M by computing the true positive TP as $|T^+ \cap T_M^+|$, true negative TN as $|T^- \cap T_M^-|$, false positive FP as $|T^- \cap T_M^+|$, and false negative FN as $|T^+ \cap T_M^-|$, respectively, where T_M^+ (respectively, T_M^-) refers to the true (respectively,



(a) Varying $|E|$ (DBP) (b) Varying $|E|$ (WIKI) (c) Varying $|\Gamma^+|$ (DBP) (d) Varying $|\Gamma^+|$ (WIKI)

(e) Varying σ (DBP) (f) Varying σ (WIKI) (g) Varying k (DBP) (h) Varying k (WIKI)

Fig. 5. Impact factors of efficiency

false) facts reported by M. We evaluate four metrics: prediction rate $\text{Pred}(M) = \frac{\text{TP+TN}}{|T|}$, precision $\text{Prec}(M) = \frac{\text{TP}}{|T_M^+|}$, recall $\text{Rec}(M) = \frac{TP}{|T^+|}$, and F_1 score $F_1(M) = \frac{2\text{Prec}(M)\cdot\text{Rec}(M)}{\text{Prec}(M)+\text{Rec}(M)}$.

Efficiency. For GFact and GFact$_R$, we report the performance of GFC_stream, GFC_spawn, and GFC_batch, compared to (1) the efficiency of rule discovery algorithm AMIE+, and (2) the learning cost of supervised link prediction methods of KGMiner, PRA, SFE, and TransD. For a fair comparison, we evaluate all of these methods over the same set of training facts Γ.

6.2 Experimental Results

Overview of results. We find the following. First, it is feasible to discover GFCs in large graphs (Exp-1). For example, it takes 211 seconds for GFC_stream to discover GFCs over YAGO (not shown) with 4 million edges and 3,000 training facts. On average, it outperforms AMIE+ by 3.4 times. For another example, GFC_spawn takes 100 seconds to discover patterns on DBpedia, which is nearly 90 times faster than AMIE+ and TransD. Second, GFCs can improve the accuracy of fact checking models (Exp-2). For example, GFact achieves additional 30%, 20%, and 5% gain of precision over DBpedia, and 20%, 15%, and 16% gain of F_1 score over Wikidata when compared to AMIE+, PRA, and KGMiner, respectively. Third, our case study shows that GFC-based models suggest interpretable results (Exp-3) for comprehensive fact checking in knowledge bases and news data.

Next we report the details of our findings.

Exp-1: Efficiency. We report the efficiency of GFC_stream and GFC_spawn compared to GFC_batch, rule discovery cost of AMIE+, and learning cost of TransD over DBpedia and Wikidata. We found that PRA is insensitive to the graph size because it samples a fixed number of paths from the graphs, SFE is also fast as its inherent simple computation model, and KGMiner has very unstable learning time. Thus, we omit the results for these algorithms and only compare the prediction accuracy and discuss them in case studies.

Varying $|E|$. For DBpedia, fixing $|\Gamma^+| = 15K$, support threshold $\sigma = 0.1$, confidence threshold $\theta = 0.005$, $k = 200$, we sampled five graphs with size (number of edges) varied from 0.6 to $1.8M$, and for Wikidata, fixing $|\Gamma^+| = 6K$, support threshold $\sigma = 0.001$, confidence threshold $\theta = 5 \times 10^{-5}$, $k = 50$, we sampled five graphs, with size varied from 0.4 to $2.0M$ edges. Figure 5(a) and 5(b) both show that all methods take longer time over larger $|E|$, as expected, and both GFC_stream

Table 2. Effectiveness: average accuracy

	YAGO				DBpedia				Wikidata				MAG			
Model	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1
GFact$_R$	0.73	0.40	0.75	0.50	0.70	0.43	0.72	0.52	0.85	0.55	0.64	0.55	0.86	0.78	0.55	0.64
AMIE+ (all)	0.71	0.44	0.76	0.51	0.69	0.50	0.85	0.58	0.64	0.42	0.78	0.48	0.70	0.53	0.62	0.52
AMIE+ (top-k)	0.43	0.34	0.49	0.39	0.60	0.45	0.64	0.47	0.52	0.34	0.65	0.42	0.69	0.40	0.62	0.46
GFact	**0.89**	**0.81**	0.60	**0.66**	**0.91**	**0.80**	0.55	**0.63**	**0.92**	**0.82**	0.63	**0.68**	**0.90**	0.86	0.62	**0.71**
PRA	0.87	0.69	0.34	0.37	0.88	0.60	0.41	0.45	0.90	0.65	0.51	0.53	0.77	0.88	0.21	0.32
KGMiner	0.87	0.62	0.36	0.40	0.88	0.75	0.60	0.63	0.90	0.63	0.49	0.52	0.76	0.74	0.17	0.27
SFE	0.80	0.56	0.89	0.66	0.69	0.50	0.93	0.60	0.58	0.38	0.96	0.52	0.77	0.56	0.83	0.63
TransD	0.76	0.41	0.65	0.48	0.78	0.37	0.56	0.44	0.70	0.40	0.76	0.51	0.63	0.38	0.76	0.48

and GFC_spawn have comparable performance for Wikidata. We observe the following. First, Figure 5(a) shows that GFC_spawn is about 90 times faster than AMIE+ and TransD, on average, due to its greedy selection and the number of verification is bounded. Second, Figure 5(b) shows that GFC_stream is 3.2 and 4.1 times faster than AMIE+ and GFC_batch, on average, respectively, due to its approximate matching scheme and top-k selection strategy. Third, although AMIE+ is faster than GFC_stream over smaller graphs in Figure 5(a), it returns few rules because it ignores low frequent triple patterns in the data graph. AMIE+ does not run to completion when the rule size is set to 5. Fourth, the cost of TransD is high for both datasets, as it requires building the mapping matrix for all the entities and takes many iterations to converge.

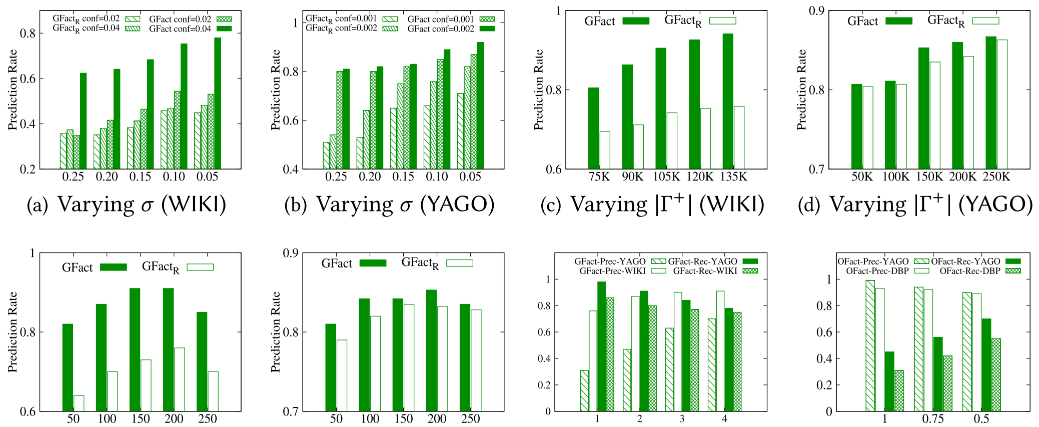
Varying $|\Gamma^+|$. For DBpedia, fixing $|E| = 1.8M$, $\sigma = 0.1$, $\theta = 0.005$, $k = 200$, we varied $|\Gamma^+|$ from 3 to $15K$, and for Wikidata, fixing $|E| = 2M$, $\sigma = 0.001$, $\theta = 5 \times 10^{-5}$, $k = 50$, we varied the positive training facts $|\Gamma^+|$ from 1,200 to 6,000. As shown in Figure 5(c) and 5(d), although all the methods take longer time for larger $|\Gamma^+|$, GFC_stream and GFC_spawn scale better with $|\Gamma^+|$ due to their selection strategies. GFC_stream outperforms GFC_batch and AMIE+ by 3.54 and 5.1 times, on average, respectively. In addition, both GFC_stream and GFC_spawn have better performance than AMIE+ and TransD.

Varying σ. For DBpedia, fixing $|E| = 1.8M$, $|\Gamma^+| = 15K$, $\theta = 0.005$, $k = 200$, we varied σ from 0.05 to 0.25. For Wikidata, fixing $|E| = 2.0M$, $|\Gamma^+| = 6K$, $\theta = 5 \times 10^{-5}$, $k = 50$, we varied σ from 0.002 to 0.010. As shown in Figure 5(e) and 5(f), GFC_batch takes longer time over smaller σ, due to more patterns and because GFC candidates need to be verified. However, GFC_stream and GFC_spawn are much less sensitive. This is because GFC_stream selects a bounded number of patterns with early termination and GFC_spawn has spawned a bounded number of patterns to verify. Neither requires to enumerate and verify all the patterns.

Varying k. For DBpedia, fixing E=$1.8M$, $|\Gamma^+| = 15K$, $\sigma = 0.1$, $\theta = 0.005$, we varied k from 200 to 1000. For Wikidata, fixing $E = 2.0M$, $|\Gamma^+| = 6K$, $\sigma = 0.001$, $\theta = 5 \times 10^{-5}$, we varied k from 100 to 500. Figure 5(g) and 5(h) show that GFC_stream is more sensitive to k because it takes longer to find k best patterns for each sieve value. Although GFC_batch is less sensitive, the major bottleneck is its verification cost. In addition, we found that (1) with larger ϵ, less number of patterns are needed, and thus GFC_stream takes less time, and (2) GFC_spawn and GFC_stream may outperform each other for different input and datasets. Even though GFC_stream has a tunable parameter ϵ, the actual performance is based on the actual number of verified patterns.

Exp-2: Accuracy. We report the accuracy of all the models in Table 2.

Rule-based models. We apply the same support threshold $\sigma = 0.1$ for AMIE+ and GFact$_R$. We sample 20 triple patterns $r(x, y)$ and report the average accuracy. We set $\theta = 0.005$ for GFact$_R$ and



(a) Varying σ (WIKI) (b) Varying σ (YAGO) (c) Varying $|\Gamma^+|$ (WIKI) (d) Varying $|\Gamma^+|$ (YAGO)

(e) Varying k (WIKI) (f) Varying k (YAGO) (g) Varying b (h) Varying α

Fig. 6. Impact factors of accuracy

set $k = 200$. We compare GFact$_R$ to two cases: using top 200 (AMIE+ (top-k)) or all (AMIE+ (all)) AMIE rules discovered by AMIE+. For the latter case, AMIE+ discovers, on average, 854, 1,078, 780, and 330 AMIE rules for each triple pattern $r(x, y)$ over YAGO, DBpedia, Wikidata, and MAG, respectively.

The accuracy (prediction rate, precision, recall, and F_1) are reported in Table 2 (first three rows). First, even with 200 rules, GFact$_R$ constantly improves AMIE+ using all the AMIE rules in prediction rate with up to 21% gain and comparable performance on precision, recall, and F_1 measure. This verifies the effectiveness of the discovered graph patterns and their matches, as well as the feature sets derived from GFCs. Second, moreover, when top 200 AMIE rules are used, GFact$_R$ has on average 23% gain in prediction rate and 15% gain in F_1 score compared to AMIE+.

Supervised models. We next compare GFact to the supervised link prediction models PRA, KGMiner, SFE, and TransD (Table 2). In general, GFact achieves the highest prediction rates and F_1 scores. It outperforms PRA with 12% gain on precision and 23% gain on recall on average, and it outperforms KGMiner with 14% gain on precision and 19% recall. On average, it achieves 16% higher F_1 scores over KGMiner, SFE, and TransD. Indeed, GFact extracts useful features with both high significance and diversity, beyond path features.

We observe the following. First, GFact achieves the highest prediction rate for all datasets, because it exploits richer sets of features from graph patterns and matches, contributed by discovered GFCs. Second, even with a small amount of positive examples (20% true facts and 80% false facts), GFact achieves the highest precision over YAGO, DBpedia, and Wikidata, and outperforms PRA and KGMiner in most cases. Third, the recall of GFact is comparable with, if not better than, PRA and KGMiner over all the datasets. SFE and TransD achieve higher recall, at a cost of sacrificing the prediction rate and precision. We found that more false facts are introduced by both methods, which can be mitigated by incorporating features from both topology and semantics as in GFact.

We next evaluate the impact of various factors to the accuracy of GFC-based fact checking.

Varying σ and θ. For Wikidata, fixing $|E| = 2.0M$, $|\Gamma^+| = 135K$, and $k = 200$, we varied σ from 0.05 to 0.25 and compare patterns to confidence 0.02 and 0.04, respectively, as shown in Figure 6(a). For YAGO, fixing $|E| = 1.5M$, $|\Gamma^+| = 250K$, and $k = 200$, we varied σ from 0.05 to 0.25 and compare patterns to confidence 0.001 and 0.002, respectively, as shown in Figure 6(b). Both figures show that GFact and GFact$_R$ have higher prediction rates when the support threshold (respectively, confidence) is lower (respectively, higher). This is because fewer patterns can be discovered with higher support, leading to more "misses" in facts; whereas higher confidence leads to stronger association of patterns and more accurate predictions. In general, GFact achieves higher prediction rates.

Varying $|\Gamma^+|$. For Wikidata, fixing $|E| = 2.0M$, $\sigma = 0.001$, $\theta = 5 \times 10^{-5}$, $k = 200$, we vary $|\Gamma^+|$ from 75 to $135K$ as shown in Figure 6(c), For YAGO, fixing $|E| = 1.5M$, $\sigma = 0.01$, $\theta = 0.005$, $k = 200$, we vary $|\Gamma^+|$ from 50 to $250K$ as shown in Figure 6(d). Both figures tell us that GFact and GFact$_R$ have a higher prediction rate with more positive examples provided. The results for their precisions (not shown) are consistent. We also observe that the accuracy of our methods is not sensitive to the actual graph size if $|\Gamma|$ remains unchanged (thus not shown).

Varying k. For Wikidata, fixing $|E| = 2.0M$, $|\Gamma^+| = 135K$, $\sigma = 0.001$, and $\theta = 5 \times 10^{-5}$, we varied k from 50 to 250 as shown in Figure 6(e). For YAGO, fixing $|E| = 1.5M$, $|\Gamma^+| = 250K$, $\sigma = 0.01$, and $\theta = 0.05$, we varied k from 50 to 250 as shown in Figure 6(f). Both figures show that the prediction rate first increases and then decreases. For rule-based models, more rules increase accuracies by covering more true facts while increasing the risk of hitting false facts. For supervised link prediction, models will be underfitting with few features for small k and will be overfitting with too many features due to large k. We observe that $k = 200$ is the best setting for a high prediction rate for both datasets. This also explains the need for top-k discovery instead of a full enumeration of rules in applications like fact checking.

Varying b. For Wikidata, fixing $|E| = 2M$, $\sigma = 0.001$, $\theta = 5 \times 10^{-5}$, and $k = 200$, and for YAGO, fixing $|E| = 1.5M$, $\sigma = 0.01$, $\theta = 0.005$, and $k = 200$, we varied b from 1 to 4 for both datasets to train the models. Figure 6(g) verifies an interesting observation: smaller patterns contribute more to recall. This is because smaller patterns are more likely to be matched to the graph G and thus are easier to "hit" facts, whereas larger ones tend to have more informative constraints to recognize true facts once learned, yet harder to be satisfied by a new fact. For precision, as the matches are harder for both positive and negative examples, for each single model M, the true positives $\mathsf{TP} = |T^+ \cap T_M^+|$ and false positives $\mathsf{FP} = |T^- \cap T_M^+|$ both drop with b increased. Thus, the changes of precisions are determined by which of TP and FP drops faster.

Varying α. We next evaluate the impact of label similarity threshold α. As the result varies for different triple patterns from each dataset, we selected 10 triple patterns, all carrying pseudofunctional predicates (to favor PCA), and report the average accuracy. Figure 6(h) shows the impact of α for the performance of GFact over YAGO and DBpedia. The precision is generally not sensitive to the change of α. In contrast, recall is more sensitive: smaller α improves recall. This is because smaller α allows GFact to learn from more facts (some are marked as false under PCA) and thus capture more true facts that cannot be predicted with larger α. Note that when $\alpha = 1$, it predicates facts under strict PCA, which achieves much smaller recall.

Exp-3: Case study. We perform case studies to evaluate the application of GFCs.

Test cases. A test case consists of a triple pattern $r(x, y)$ and a set of test facts that are instances of $r(x, y)$. We categorize each triple pattern into the following four classes of cases:

(1) *Functional cases* refer to test facts that pertain to a functional predicate (a "one-to-one" mapping). For example, the relation `capitalOf` between two locations can only map to each other through the relation. For example, "London" is the capital of "United Kingdom."
(2) Pseudofunctional cases carry predicates that can be "one-to-many" but have high functionality ("usually" functional). For example, the relation `graduatedFrom` between a person and a school is not necessarily functional, but it is functional for most "persons."

Table 3. Case Study: F_1 scores over 30 test cases

Type	ID	Test case $r(x, y)$	AMIE+	PRA	KGMiner	SFE	TransD	GFact$_R$	GFact
Functional	1	<prefecture, `hasCapital`, district> (YAGO)	0.75	1.00	1.00	0.67	0.29	0.60	0.88
	2	<site, `hasCapital`, district> (YAGO)	1.00	1.00	1.00	0.57	0.29	1.00	1.00
	3	<organisation, `owningCompany`, place> (DBP)	0.67	1.00	1.00	1.00	0.00	0.67	1.00
	4	<sportPlayer, `successor`, sportPlayer> (DBP)	1.00	1.00	1.00	1.00	0.00	0.90	1.00
	5	<position, `appliesTo`, jurisdiction> (WIKI)	0.74	1.00	1.00	0.41	0.50	0.77	1.00
	6	<wikipedia, `mainTopic`, family> (WIKI)	0.00	1.00	1.00	0.91	0.59	0.17	1.00
	7	<railStation, `isLocatedAt`, village> (WIKI)	0.67	0.77	0.93	0.52	0.63	0.12	0.88
Pseudofunctional	8	<team, `created`, song> (YAGO)	0.00	0.00	0.00	0.96	0.73	0.79	**0.98**
	9	<district, `participatedIn`, conflict> (YAGO)	0.52	0.60	0.92	0.58	0.67	0.96	**0.96**
	10	<event, `maidenFlight`, aircraftType> (DBP)	1.00	0.00	0.86	0.55	0.89	0.87	0.94
	11	<place, `representative`, politician> (DBP)	0.35	0.90	0.95	0.81	0.61	0.83	**0.98**
	12	<route, `junctionOf`, city> (DBP)	0.57	0.67	1.00	0.44	0.75	0.64	0.93
	13	<music, `subsequentWork`, book> (DBP)	0.45	0.93	0.93	0.52	0.42	0.84	0.88
	14	<film, `followedBy`, book> (WIKI)	0.35	1.00	1.00	0.40	0.44	0.31	0.83
Pseudofunctional (inverse)	15	<person, `graduatedFrom`, institute> (YAGO)	0.54	0.09	0.47	0.65	0.73	0.76	**0.97**
	16	<person, `worksAt`, school> (YAGO)	0.52	0.19	0.59	0.56	0.70	0.51	**0.90**
	17	<scientist, `residenceOf`, city> (YAGO)	0.37	0.14	0.77	0.45	0.56	0.30	**0.88**
	18	<event, `previousMission`, event> (DBP)	1.00	1.00	0.33	0.54	0.56	0.57	0.93
	19	<music, `subsequentWork`, music> (DBP)	0.49	1.00	0.96	0.54	0.53	0.64	1.00
	20	<article, `isOf`, genre> (WIKI)	0.83	0.00	0.75	0.83	0.67	0.24	**0.84**
	21	<MLPaper, `publishedIn`, journal> (MAG)	0.24	0.11	0.00	0.59	0.49	0.46	**0.67**
Nonfunctional	22	<airport, `isConnectedTo`, airport> (YAGO)	0.54	0.35	0.00	0.84	0.72	0.76	**0.88**
	23	<country, `dealsWith`, country> (YAGO)	0.35	0.00	0.82	0.62	0.52	0.74	**0.91**
	24	<politician, `associate`, politician> (DBP)	0.34	0.55	0.67	0.44	0.23	0.57	0.75
	25	<athlete, `almaMater`, school> (DBP)	0.51	0.00	0.84	0.51	0.69	0.80	**0.88**
	26	<film, `basedOn`, book> (WIKI)	0.32	0.29	0.36	0.41	0.43	0.45	**0.64**
	27	<drawing, `createdBy`, human> (WIKI)	0.80	0.06	0.11	0.66	0.60	0.81	**0.93**
	28	<AIPaper, `reference`, DBPaper> (MAG)	0.62	0.51	0.48	0.70	0.43	0.49	**0.82**
	29	<DMPaper, `reference`, AIPaper> (MAG)	0.51	0.38	0.43	0.64	0.42	0.54	**0.64**
	30	<author, `isAffiliatedTo`, institute> (MAG)	0.70	0.37	0.51	0.84	0.88	0.50	**0.99**

(3) Inverse pseudofunctional cases include the facts with predicates ("many-to-one") as *pseudofunctional* predicates inversed. For example, `almaMaterOf` is the inverse predicate of `graduatedFrom`.

(4) Nonfunctional cases are those predicates that allow "many-to-many" mapping. For example, `ActIn` between "actors" and "movies."

Accuracy. We show 30 relations from each of the categories and report their overall F_1 measure in Table 3. Nonfunctional cases refer to facts with predicates that are many-to-many relations, for which PCA may not hold [14]. We found that GFact has good performance for all test cases, especially for those with nonfunctional cases. Indeed, we find that the relaxation of label equality in both pattern matching and in extracting training facts by the WPCA help in improving both precision and recall of the fact checking models.

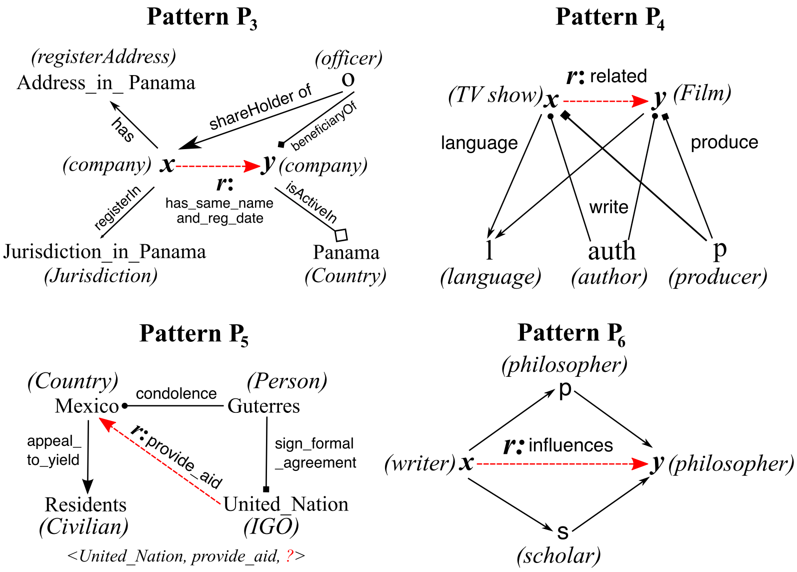


Fig. 7. Real-world GFCs discovered by GFact

Interpretability. We further illustrate four top GFCs in Figure 7, which contribute to highly important features in GFact with high confidence and significance over real-world financial network Offshore, news GDELT, and knowledge bases DBpedia and Wikidata, respectively:

(1) GFC $\varphi_3 : P_3(x, y) \rightarrow$ `has_same_name_and_reg_date`(x, y) (Offshore) states that two (anonymous) companies are likely to have the same name and registration date if they share the same shareholder and beneficiary, and one is registered within a jurisdiction in Panama, and the other one is active in Panama. This GFC has support 0.12 and is quite significant. For this $r(x, y)$, AMIE+ discovers a top rule as `registerIn`$(x$, Jurisdiction_in_Panama) $\wedge$`registerIn`$(y$, Jurisdiction_in_Panama) implies x and y has the same name and registration date. This rule has a relatively low prediction rate.

(2) GFC $\varphi_4 : P_4(x, y) \rightarrow$ `relevant`(x, y) (DBpedia) states that a TV show and a film have relevant content if they have the common language, authors, and producers. This GFC has a support 0.15 and high confidence and significant scores. Within bound 3, AMIE+ reports a top rule as `Starring`$(x, z) \wedge$`Starring`$(y, z) \rightarrow$ `relevant`(x, y), which has low accuracy.

(3) GFC $\varphi_5 : P_5(x, y) \rightarrow$ `provide_aid`(x, y) (GDELT) is a Web news pattern that predicts to which country the intergovernmental organization (IGO) will provide aid. It states that if a country starts to evacuate its residents (denoted by appeal_to_yield) and a Person (e.g., "Guterres") gives condolence to this country, then the IGO (e.g., United Nation) that interacted with the person will likely provide aid to this country. We found that φ_5 successfully predicted the missing values of many triples that satisfy $P_5(x, y)$ by tracing back to the original news report. In this example, the missing object in the triple is predicted as "Mexico," which indicates the 2017 Mexico earthquake.

(4) GFC $\varphi_6 : P_6(x, y) \rightarrow$ `influences`(x, y) (Wikidata) states that a *writer* v_x influences a *philosopher* v_y if v_x influences a *philosopher* p and a *scholar* s, who both influence the philosopher v_y. This rule identifies true facts such as <Bertrand Russell, `influences`,

Ludwig Wittgenstein>, the influence between a logician and a philosopher, enabled by label similarity following an associated ontology.[1]

7 CONCLUSION

We have introduced GFCs, a class of rules that incorporate graph patterns to predict facts in knowledge graphs. We developed a stream-based rule discovery algorithm to find useful GFCs, given observed true and false facts, with optimality guarantees on rounded quality scores. We also developed a space-efficient alternative. We have shown that GFCs can be readily applied as rule models or provide useful instance-level features in supervised link prediction. Our experimental study has verified the effectiveness and efficiency of GFC-based techniques, as well as their applications in knowledge base completion.

This is a first step toward applying graph patterns to improve fact checking in graph data. We are evaluating GFCs with more real-world graphs and pattern models. One future topic is to extend GFC techniques for learning-based knowledge base completion, link prediction, entity resolution, social recommendation, and graph data imputation. A second topic is to extend GFC model to cope with multilabel knowledge graphs. A third topic is to develop parallel GFC discovery and fact checking algorithms over distributed knowledge bases.

REFERENCES

[1] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of KDD*.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan. 2011. Node classification in social networks. In *Social Network Data Analytics*. Springer, 115–148.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*.

[4] Hongyun Cai, Vincent W. Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques and applications. arXiv:1709.07604.

[5] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of AAAI*.

[6] Yang Chen and Daisy Zhe Wang. 2014. Knowledge expansion over probabilistic knowledge bases. In *Proceedings of SIGMOD*.

[7] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS One* 10, 6, 30128193.

[8] William Cukierski, Benjamin Hamner, and Bo Yang. 2011. Graph-based features for supervised link prediction. In *Proceedings of IJCNN*.

[9] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, et al. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of KDD*.

[10] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. GRAMI: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment* 7, 7, 517–528.

[11] Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. 2015. Association rules with graph patterns. *Proceedings of the VLDB Endowment* 8, 12, 1502–1513.

[12] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *Proceedings of SIGMOD*.

[13] Samantha Finn, Panagiotis Takis Metaxas, Eni Mustafaraj, Megan O'Keefe, Lindsay Tang, Susan Tang, and Laura Zeng. 2014. TRAILS: A system for monitoring the propagation of rumors on Twitter. In *Proceedings of the Computation and Journalism Symposium*.

[14] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal* 24, 6, 707–730.

[15] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of WWW*.

[16] Matt Gardner and Tom M. Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of EMNLP*.

[1] http://tools.wmflabs.org/wikidata-exports/rdf/exports/20160801/dump_download.html.

[17] Travis R. Goodwin and Sanda M. Harabagiu. 2016. Medical question answering for clinical decision support. In *Proceedings of CIKM.*

[18] Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. 2014. Data in, fact out: Automated monitoring of facts by factwatcher. *Proceedings of the VLDB Endowment* 7, 13, 1557–1560.

[19] ICIJ. 2016. Offshore Dataset. Retrieved April 8, 2019 from https://offshoreleaks.icij.org/pages/database.

[20] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL.*

[21] Chuntao Jiang, Frans Coenen, and Michele Zito. 2013. A survey of frequent subgraph mining algorithms. *Knowledge Engineering Review* 28, 1, 75–105.

[22] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of RepL4NLP.*

[23] Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP.*

[24] Kalev Leetaru and Philip A. Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *Proceedings of the ASA Annual Convention*, Vol. 2. 1–49.

[25] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, et al. 2015. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 1, 1–5.

[26] Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL/HLT.*

[27] Peng Lin, Qi Song, Jialiang Shen, and Yinghui Wu. 2018. Discovering graph patterns for fact checking in knowledge graphs. In *Proceedings of DASFAA.*

[28] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI.*

[29] Shuai Ma, Yang Cao, Wenfei Fan, Jinpeng Huai, and Tianyu Wo. 2011. Capturing topology in graph pattern matching. *Proceedings of the VLDB Endowment* 5, 4, 310–321.

[30] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. YAGO3: A knowledge base from multilingual wikipedias. In *Proceedings of CIDR.*

[31] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14, 1, 265–294.

[32] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104, 1, 11–33.

[33] Feng Niu, Ce Zhang, Christopher Ré, and Jude W. Shavlik. 2012. DeepDive: Web-scale knowledge-base construction using statistical learning and inference. In *Proceedings of VLDS.*

[34] Alexandre Passant. 2010. dbrec-music recommendations using DBpedia. In *Proceedings of ISWC.*

[35] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 0, 1-23.

[36] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *Proceedings of ISWC.*

[37] Sayan Ranu and Ambuj K. Singh. 2009. Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *Proceedings of ICDE.*

[38] Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. 2016. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the WWW Companion.*

[39] Baoxu Shi and Tim Weninger. 2016. Discriminative predicate path mining for fact checking in knowledge graphs. arXiv:1510.05911.

[40] Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding projection for knowledge graph completion. In *Proceedings of AAAI.*

[41] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. 2015. An overview of Microsoft Academic Service (MAS) and applications. In *Proceedings of WWW.*

[42] Chunyao Song, Tingjian Ge, Cindy Chen, and Jie Wang. 2014. Event pattern matching over graph streams. *Proceedings of the VLDB Endowment* 8, 4, 413–424.

[43] Qi Song, Yinghui Wu, Peng Lin, Xin Luna Dong, and Hui Sun. 2018. Mining summaries for knowledge graph search. *IEEE Transactions on Knowledge and Data Engineering* 30, 10, 1887–1900.

[44] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of WWW.*

[45] Andreas Thor, Philip Anderson, Louiqa Raschid, Saket Navlakha, Barna Saha, Samir Khuller, and Xiao-Ning Zhang. 2011. Link prediction for annotation graphs using graph summarization. In *Proceedings of the ISWC.*

[46] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research* 18, 1, 4735–4772.

[47] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM* 57, 10, 78–85.

[48] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. 2016. Knowledge base completion via coupled path ranking. In *Proceedings of ACL*.

[49] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2014. Toward computational fact-checking. *Proceedings of the VLDB Endowment* 7, 7, 589–600.

[50] Xifeng Yan, Hong Cheng, Jiawei Han, and Philip S. Yu. 2008. Mining significant graph patterns by leap search. In *Proceedings of SIGMOD*.

[51] Shengqi Yang, Yinghui Wu, Huan Sun, and Xifeng Yan. 2014. Schemaless and structureless graph querying. *Proceedings of the VLDB Endowment* 7, 7, 565–576.

[52] Ganggao Zhu and Carlos A. Iglesias. 2017. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering* 29, 1, 72–85.

Received May 2018; revised August 2018; accepted October 2018