Discovering Graph Patterns for Fact Checking in Knowledge Graphs

Peng Lin[1], Qi Song[1], Jialiang Shen[3], and Yinghui Wu[1,2](✉)

[1] Washington State University, Pullman, USA
{plin1,qsong,yinghui}@eecs.wsu.edu
[2] Pacific Northwest National Laboratory, Richland, USA
[3] Beijing University of Posts and Telecommunications, Beijing, China
shenjialiang@bupt.edu.cn

Abstract. Given a knowledge graph and a fact (a triple statement), fact checking is to decide whether the fact belongs to the missing part of the graph. This paper proposes a new fact checking method based on supervised graph pattern mining. Our method discovers discriminant graph patterns associated with the training facts. These patterns can then be used to construct classifiers based on either rules or latent features. (1) 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 quality measures to characterize useful patterns that are both discriminant and diversified. (2) We show that it is feasible to discover GFCs in large graphs, by developing a supervised pattern discovery algorithm. To find useful GFCs as early as possible, it generates graph patterns relevant to training facts, and dynamically selects patterns from a pattern stream with small update cost per pattern. We further construct two GFC-based models, which make use of ordered GFCs as predictive rules and latent features from the pattern matches of GFCs, respectively. Using real-world knowledge bases, we experimentally verify the efficiency and the effectiveness of GFC-based techniques for fact checking.

1 Introduction

Knowledge graphs have been adopted to support emerging applications, *e.g.,* web search [6], recommendation [26], and decision making [14]. 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 relation) between v_x and v_y. One of the 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) directly refine incomplete knowledge bases [2,6,19,25], (2) provide cleaned evidence for error detection in dirty knowledge bases [3,13,21,36], and (3) improve the quality of knowledge search.

J. Pei et al. (Eds.): DASFAA 2018, LNCS 10827, pp. 783–801, 2018.
https://doi.org/10.1007/978-3-319-91452-7_50

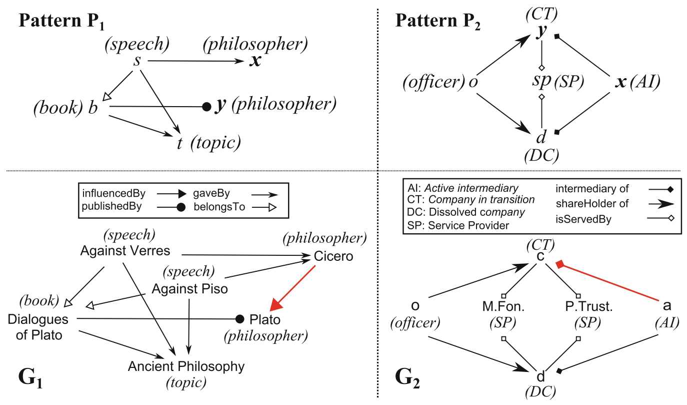


Fig. 1. Facts and associated graph patterns

Real-world facts in knowledge graphs are often associated with nontrivial regularities that involve both topological and semantic constraints beyond paths. Given observed facts, such regularities can be usually captured by graph patterns and their matches associated with the facts. Consider the following example.

Example 1. The graph G_1 in Fig. 1 illustrates a fraction of DBpedia [19] that depicts the facts about philosophers. A user is interested in finding "whether a philosopher v_x is *influenced by* another philosopher v_y". Given a relation `influencedBy` between philosophers (`influencedBy`($philosopher, philosopher$)) and an instance <Cicero, `influencedBy`, Plato> verified to be true in G_1, a graph pattern P_1 can be extracted to define `influencedBy` by stating that "*if a philosopher* v_x *(e.g., "Cicero") gave one or more speeches (e.g., "Against Piso") that cited a book of* v_y *(e.g., "Dialogues of Plato") with the same topic, then "*v_x *is likely to be influenced by* v_y".

Graph patterns with "weaker" constraints may not explain facts well. Consider a graph pattern P'_1 obtained by removing the edge `belongsTo`($speech, book$) from P_1. Although "Cicero" and "Plato" also matches P'_1, a false fact <Cicero, `influencedBy`, John Stuart Mill> also matches P'_1 (not shown). Therefore, discriminant patterns that well distinguish observed true and false facts should be discovered.

As another example, consider graph G_2, a fraction of a real-world offshore activity network [16] in Fig. 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 "*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*."

These graph patterns can be easily interpreted as rules, and the matches of the graph patterns readily provide instance-level evidence to "explain" the facts. These matches also indicate more accurate predictive models for various facts. We ask the following question: *How to characterize useful graph patterns and efficiently discover useful graph patterns to support fact checking in large knowledge graphs?*

Contribution. We propose models and pattern discovery algorithms that explicitly incorporate discriminant graph patterns to support fact checking in knowledge graphs.

(1) We introduce *graph fact checking rules* (GFCs) (Sect. 2). GFCs incorporate discriminant graph patterns as the antecedent and generalized triple patterns as the consequent, and characterize fact checking with graph pattern matching. We adopt computationally efficient pattern models to ensure tractable fact checking via pattern matching.

    We also develop statistical measures (*e.g.,* support, confidence, significance, and diversity) to characterize useful GFCs (Sect. 3). Based on these measures, we formulate the top-k GFC discovery problem to mine useful GFCs for fact checking.
(2) We develop a feasible supervised pattern discovery algorithm to compute GFCs over a set of training facts (Sect. 4). In contrast to conventional pattern mining, the algorithm solves a submodular optimization problem with provable optimality guarantees, by a single scan of a stream of graph patterns, and incurs a small cost for each pattern.
(3) To evaluate the applications of GFCs, we apply GFCs to enhance rule-based and learning-based models, by developing two such classifiers. The first model directly uses GFCs as rules. The second model extracts instance-level features from the pattern matches induced by GFCs to learn a classifier (Sect. 4.2).
(4) Using real-world knowledge bases, we experimentally verify the efficiency of GFC-based techniques (Sect. 5). We found that the discovery of GFCs is feasible over large graphs. GFC-based fact checking also achieves high accuracy and outperforms its counterparts using Horn clause rules and path-based learning. We also show that the models are highly interpretable by providing case studies.

Related Work. We categorize the related work as follows.

Fact Checking. Fact checking has been studied for both unstructured data [10,28] and structured (relational) data [15,37]. These work rely on text analysis and crowd sourcing. Automatic fact checking in knowledge graphs is not addressed in these work. Beyond relational data, several methods have been studied to predict triples in graphs.

(1) Rule-based models extract association rules to predict facts. AMIE (or its improved version AMIE+) discovers rules with conjunctive Horn clauses [11, 12] for knowledge base enhancement. Beyond Horn-rules, GPARs [8] discover association rules in the form of $Q \Rightarrow p$, with a subgraph pattern Q and a single edge p. It recommends users via co-occurred frequent subgraphs.

(2) Supervised link prediction has been applied to train predictive models with latent features extracted from entities [6,18]. Recent work make use of path features [4,5,13,29,34]. The paths involving targeted entities are sampled from 1-hop neighbors [5] or via random walks [13], or constrained to be shortest paths [4]. Discriminant paths with the same ontology are grouped to generate positive and negative examples in [29].

Rule-based models are easy to interpret but usually cover only a subset of useful patterns [24]. It is also expensive to discover useful rules (*e.g.,* via subgraph isomorphism) [8]. On the other hand, latent feature models are more difficult to be interpreted [12,24] compared with rule models. Our work aims to balance the interpretability and model construction cost. (a) In contrast to AMIE [12], we use more expressive rules enhanced with graph patterns to express both constant and topological context of facts. Unlike [8], we use approximate pattern matching for GFCs instead of subgraph isomorphism, since the latter may produce redundant examples and is computationally hard in general. (b) GFCs can induce useful and discriminant features from patterns and subgraphs, beyond path features [5,13,34]. (c) GFCs can be used as a standalone rule-based method. They also provide context-dependent features to support supervised link prediction to learn highly interpretable models. These are not addressed in [8,12].

Graph Pattern Mining. Frequent pattern mining defined by subgraph isomorphism has been studied for a single graph. GRAMI [7] discovers frequent subgraph patterns without edge labels. Parallel algorithms are also developed for association rules with subgraph patterns [8]. In contrast, (1) we adopt approximate graph pattern matching for feasible fact checking, rather than subgraph isomorphism as in [7,8]. (2) we develop a more feasible stream mining algorithm with optimality guarantees on rule quality, which incurs a small cost to process each pattern. (3) Supervised graph pattern mining over observed ground truth is not discussed in [7,8]. 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) [9] 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 towards data completion rather than identifying errors with hard constraints [27].

2 Fact Checking with Graph Patterns

We review the notions of knowledge graphs and fact checking. We then introduce a class of rules that incorporate graph patterns for fact checking.

Knowledge Graphs. A knowledge graph [6] is a directed graph $G = (V, E, L)$, which consists of a finite set of nodes V, a set of edges $E \subseteq V \times V$, and for each node $v \in V$ (resp. edge $e \in E$), $L(v)$ (resp. $L(e)$) is a label from a finite alphabet, which encodes the content of v (resp. e) such as properties, relations or names.

For the example in Fig. 1, a subject $v_x =$ "Cicero" carries a *type* $x =$ "philosopher" and an object $v_y =$ "Plato" carries a *type* $y =$ "philosopher". A fact $<v_x, r, v_y> = <\text{Cicero}, \texttt{influencedBy}, \text{Plato}>$ is an edge e in G encoded with label "influencedBy" between the subject v_x and the object v_y.

Fact Checking in Knowledge Graphs. Given a knowledge graph G and a new fact $<v_x, r, v_y>$, where v_x and v_y are in G, the task of fact checking is to learn and use a model M to decide whether the relation r exists between v_x and v_y [24]. This task can be represented by a binary query in the form of $<v_x, r?, v_y>$, where the model M outputs "true" or "false" for the query. A variant of fact checking is a class of queries in the form of $<v_x, r, v_y?>$, where the model M outputs a set of possible values of type y for specified v_x and r that are likely to be true in G.

Graph Pattern. A graph pattern $P(x, y) = (V_P, E_P, L_P)$ (or simply P) is a directed graph that contains a set of pattern nodes V_P and pattern edges E_P, respectively. Each pattern node $u_p \in V_P$ (resp. edge $e_p \in E_P$) has a label $L_P(u_p)$ (resp. $L_P(e_p)$). Moreover, it contains two designated *anchored nodes* u_x and u_y in V_P of type x and y, respectively. Specifically, when it contains a single pattern edge with label r between u_x and u_y, P is called a *triple pattern*, denoted as $r(x, y)$.

Pattern Matches. Given a graph pattern $P(x, y)$ and a knowledge graph G, a node match $v \in V$ of a pattern node u_p has the same label of node u_p, and an edge match $e = (v, v')$ of a pattern edge $e_p = (u, u')$ is induced by the matches v and v' of nodes u and u', respectively. We say a pattern P *covers* a fact $<v_x, r, v_y>$ in G if v_x and v_y match its anchored nodes u_x and u_y, respectively. Specifically, a pattern P can cover a fact by enforcing two established semantics.

(1) Subgraph patterns [7] define pattern matching in terms of subgraph isomorphism, induced by bijective functions.
(2) *Approximate patterns* [22,32] specify a matching relation R with constraints below to preserve both parent and child relations of P. For each pair $(u, v) \in R$,
   - for every edge $e = (u, u') \in E_P$, there exists an edge match $e' = (v, v') \in E$; and
   - for every edge $e = (u'', u) \in E_P$, there exists an edge match $e' = (v'', v) \in E$.

To ensure feasible fact checking in large knowledge graphs, we adopt approximate patterns for our model (see "Semantics").

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 $\varphi : P(x, y) \to r(x, y)$ 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". To ensure computationally efficient fact checking, we prefer approximate patterns instead of subgraph patterns. Subgraph isomorphism may be an overkill in capturing meaningful patterns [22,31,32] and is expensive (NP-hard). Moreover, it generates (exponentially) many isomorphic subgraphs, and thus introduces redundant features for model learning. In contrast, it is in $O(|V_P|(|V_P| + |V|)(|E_P| + |E|))$ time to find whether a fact is covered by an approximate pattern [22]. The tractability carries over to the validation of GFCs (Sect. 4).

Example 2. Consider the patterns and graphs in Fig. 1. To verify the influence between philosophers, a GFC $\varphi_1 : P_1(x, y) \to \texttt{influencedBy}(x, y)$. Pattern P_1 has two anchored nodes x and y, both with type `philosopher`, and covers the pair $(\texttt{Cicero}, \texttt{Plato})$ in G_1. Another GFC $\varphi_2 : P_2(x, y) \to \texttt{intermediaryOf}(x, y)$ verifies the service between a pair of matched entities (a, c). Note that with subgraph isomorphism, P_1 induces two subgraphs of G_1 that only differ by entities with label `speech`. It is not practical for users to inspect such highly overlapped subgraphs.

Remarks. We compare GFCs with two models below. (1) Horn rules are adopted by AMIE+ [11], in the form of $\bigwedge B_i \to r(x, y)$, where each B_i is an atom (fact) carrying variables. It mines only closed (each variable appears at least twice) and connected (atoms transitively share variables/entities to all others) rules. We allow general approximate graph patterns in GFCs to mitigate missing data and capture richer context features for supervised models (Sect. 4). (2) The association rules with graph patterns [8] have similar syntax with GFCs but adopt strict subgraph isomorphism for social recommendation. In contrast, we define GFCs with semantics and quality measures (Sect. 3) specified for observed true and false facts to support fact checking.

3 Supervised GFC Discovery

To characterize useful GFCs, we introduce a set of measures that extend their counterparts from established rule-based models [12] and discriminant patterns [38], specialized for a set of training facts. We then formalize supervised GFC discovery problem.

Statistical Measures. Our statistical measures are defined for a given graph $G = (V, E, L)$ and a set of *training facts* Γ. The training facts Γ consists of two classes: true facts Γ^+, which contains validated facts that hold in G; and false facts Γ^-, which contains triples not in G, respectively. Following common practice in knowledge base completion [24], we adopt the silver standard to construct Γ from G, where (1) $\Gamma^+ \subseteq E$, and (2) Γ^- are created following partial closed world assumption (see "Confidence").

We use the following notations. Given a GFC $\varphi : P(x, y) \to r(x, y)$, graph G, facts Γ^+ and Γ^-, (1) $P(\Gamma^+)$ (resp. $P(\Gamma^-)$) refers to the set of training facts in

Γ^+ (resp. Γ^-) that are covered by $P(x,y)$ in Γ^+ (resp. Γ^-). $P(\Gamma)$ is defined as $P(\Gamma^+) \cup P(\Gamma^-)$, *i.e.,* all the facts in Γ covered by P. (2) $r(\Gamma^+), r(\Gamma^-)$, and $r(\Gamma)$ are defined similarly.

Support and Confidence. The support of a GFC $\varphi : P(x,y) \to r(x,y)$, denoted by $\mathsf{supp}(\varphi, G, \Gamma)$ (or simply $\mathsf{supp}(\varphi)$), is defined as

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

Intuitively, the support is the fraction of the true facts as instances of $r(x,y)$ that satisfy the constraints of a graph pattern $P(x,y)$. It extends head coverage, a practical version for rule support [12] to address triple patterns $r(x,y)$ with not many matches due to the incompleteness of knowledge bases.

Given two patterns $P_1(x,y)$ and $P_2(x,y)$, we say $P_1(x,y)$ *refines* $P_2(x,y)$ (denoted by $P_1(x,y) \preceq P_2(x,y)$, if P_1 is a subgraph of P_2 and they pertain to the same pair of anchored nodes (u_x, u_y). We show that GFC support preserves anti-monotonicity in terms of pattern refinement.

Lemma 1. *For graph G, given any two* GFCs $\varphi_1 : P_1(x,y) \to r(x,y)$ *and* $\varphi_2 : P_2(x,y) \to r(x,y)$, *if* $P_1(x,y) \preceq P_2(x,y)$, $\mathsf{supp}(\varphi_2) \le \mathsf{supp}(\varphi_1)$.

Proof Sketch: It suffices to show that any pair (v_{x_2}, v_{y_2}) covered by P_2 in G is also covered by $P_1(x,y)$. Assume there exists a pair (v_{x_2}, v_{y_2}) covered by P_2 but not by P_1, and assume *w.l.o.g.* 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 descendant of u_x in P_1, such that no ancestor or descendant of v_{x_2} in G is a match. As P_2 refines P_1, both (a) and (b) lead to that v_{x_2} is not covered by P_2, which contradicts the definition of approximate patterns. □

Following rule discovery in incomplete knowledge base [12], we adopt *partial closed world assumption* (PCA) to characterize the confidence of GFCs. Given triple pattern $r(x,y)$ and a true instance $<v_x, r, v_y> \in r(\Gamma^+)$, PCA assumes that a missing instance $<v_x, r, v'_y>$ of $r(x,y)$ is a false fact if v_y and v'_y have different values. We define a normalizer set $P(\Gamma^+)_N$, which contains the entity pairs (v_x, v_y) that are in $P(\Gamma^+)$, and each pair has at least a false counterpart under PCA. The confidence of φ in G, denoted as $\mathsf{conf}(\varphi, G, \Gamma)$ (or simply $\mathsf{conf}(\varphi)$), is defined as

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

The confidence measures the probability that a GFC holds over the entity pairs that satisfy $P(x,y)$, normalized by the facts that are assumed to be false under PCA. We follow PCA to construct false facts in our experimental study.

Significance. A third measure quantifies how significant a GFC is in "distinguishing" the true and false facts. To this end, we extend the G-test score [38]. The

G-test tests the null hypothesis of whether 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. Specifically, the score (denoted as $\mathsf{sig}(\varphi,p,n)$, or simply $\mathsf{sig}(\varphi)$) is defined as

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

where p (resp. n) is the frequency of the facts covered by pattern P of φ in Γ^+ (resp. Γ^-), *i.e.,* $p = \frac{|P(\Gamma^+)|}{|\Gamma^+|}$ (resp. $n = \frac{|P(\Gamma^-)|}{|\Gamma^-|}$). As $\mathsf{sig}(\cdot)$ is not anti-monotonic, a common practice is to use a "rounded up" score to find significant patterns [38]. We adopt an upper bound of $\mathsf{sig}(\cdot)$, denoted as $\hat{\mathsf{sig}}(\varphi,p,n)$ (or $\hat{\mathsf{sig}}(\varphi)$ for simplicity), which is defined as $\max\{\mathsf{sig}(\varphi,p,\delta),\mathsf{sig}(\varphi,\delta,n)\}$, where $\delta > 0$ is a small constant (to prevent the case that $\hat{\mathsf{sig}}(\cdot) = \infty$). It is not hard to show that $\hat{\mathsf{sig}}(\cdot)$ is anti-monotonic in terms of pattern refinement, following a proof similar to Lemma 1. Besides, we normalize $\hat{\mathsf{sig}}(\varphi)$ to $\mathsf{n}\hat{\mathsf{sig}}(\varphi)$ in range $[0,1]$ by a sigmoid function, *i.e.,* $\mathsf{n}\hat{\mathsf{sig}}(\varphi) = \tanh(\hat{\mathsf{sig}}(\varphi))$.

Redundancy-Aware Selection. In practice, one wants to find 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 set of true facts Γ^+ is known, the *coverage score* of $\mathcal{S}$, denoted as $\mathsf{cov}(\mathcal{S})$, is defined as

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

The first term, defined as $\mathsf{sig}(\mathcal{S}) = \sqrt{\sum_{\varphi\in\mathcal{S}} \mathsf{n}\hat{\mathsf{sig}}(\varphi)}$, aggregates the total significance of GFCs in $\mathcal{S}$. The second term, defined as $\mathsf{div}(\mathcal{S}) = \left(\sum_{t\in\Gamma^+}\sqrt{\sum_{\varphi\in\Phi_t(\mathcal{S})}\mathsf{supp}(\varphi)}\right)/|\Gamma^+|$, where $\Phi_t(\mathcal{S})$ refers to the GFCs in $\mathcal{S}$ that cover a true fact $t \in \Gamma^+$, quantifies the diversity of $\mathcal{S}$, following a diversity reward function [20]. Intuitively, it rewards the diversity in that there is more benefit in selecting a GFC that covers new facts, which are not covered by other GFCs in $\mathcal{S}$ yet. Both terms are normalized to $(0,\sqrt{|\mathcal{S}|}]$.

The coverage score favors GFCs that cover more distinct true facts with more discriminant patterns. Moreover, 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 $\mathcal{S}$). That is, $\mathsf{cov}(\cdot)$ is well defined in terms of submodularity [23], a property widely used to justify goodness measures for set mining. Define the marginal gain $\mathsf{mg}(\varphi,\mathcal{S})$ of a GFC φ to a set $\mathcal{S}$ ($\varphi \notin \mathcal{S}$) as $\mathsf{cov}(\mathcal{S}\cup\{\varphi\}) - \mathsf{cov}(\mathcal{S})$. We show the following result.

Lemma 2. *The function* $\mathsf{cov}(\cdot)$ *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* $\mathsf{cov}(\mathcal{S}_1) \le \mathsf{cov}(\mathcal{S}_2)$*, 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 $\varphi \notin \mathcal{S}_2$, $\mathsf{mg}(\varphi,\mathcal{S}_2) \le \mathsf{mg}(\varphi,\mathcal{S}_1)$*.*

It is easy to verify that $\mathsf{cov}(\cdot)$ is a monotone submodular function by the definition of monotone submodularity and that both $\mathsf{sig}(\mathcal{S})$ and $\mathsf{div}(\mathcal{S})$ are defined in terms of square root functions. Due to space limit, we omit the detailed proof.

We now formulate the supervised top-k GFC discovery problem over observed facts.

Top-k Supervised GFC Discovery. Given graph G, support threshold σ and confidence threshold θ, and training facts Γ^+ and Γ^- as instances of a triple pattern $r(x, y)$, and integer k, the problem is to identify a set $\mathcal{S}$ of top-k GFCs that pertain to $r(x, y)$, such that (a) for each GFC $\varphi \in \mathcal{S}$, $\mathsf{supp}(\varphi) \geq \sigma$, $\mathsf{conf}(\varphi) \geq \theta$, and (b) $\mathsf{cov}(\mathcal{S})$ is maximized.

4 Discovery Algorithm

4.1 Top-k GFC Mining

The supervised discovery problem for GFCs is not surprisingly intractable. A naive "enumeration-and-verify" algorithm that generates and verifies all k-subsets of GFC candidates that cover some examples in Γ is clearly not practical for large G and Γ. We consider more efficient algorithms.

"Batch + Greedy". We start with an algorithm (denoted as GFC_batch) that takes a batch pattern discovery and a greedy selection as follows. (1) Apply graph pattern mining (*e.g.,* Apriori [17]) to generate and verify all the graph patterns $\mathcal{P}$. The verification is specialized by an operator Verify, which invokes the pattern matching algorithm in, *e.g.,* [22] to compute the support and confidence for each pattern. (2) 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 $\mathsf{cov}(\mathcal{S}_{i-1} \cup \{\varphi_i\}) - \mathsf{cov}(\mathcal{S}_{i-1})$, and then it updates $\mathcal{S}_i$ as $\mathcal{S}_{i-1} \cup \{\varphi_i\}$.

GFC_batch guarantees a $(1 - \frac{1}{e})$ approximation, following Lemma 2 and the seminal result in [23]. Nevertheless, it requires the verification of all patterns before the construction of GFCs. The selection further requires k passes of all the verified patterns. This can be expensive for large G and Γ.

We can do better by capitalizing on stream-based optimization [1,32]. 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 result below.

Theorem 1. *Given a constant $\epsilon > 0$, there exists a stream algorithm that computes top-k GFCs with the following guarantees:*

(1) It achieves an approximation ratio $(\frac{1}{2} - \epsilon)$;
(2) It performs a single pass of all processed patterns $\mathcal{P}$, with update cost in $O((b + |\Gamma_b|)^2 + \frac{\log k}{\epsilon})$, where b is the largest edge number of the patterns, and Γ_b is the b hop neighbors of the entities in Γ.

```
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$(G, \Gamma, \sigma, \theta)$.next();
3.  while $P \neq$ null do
4.  if $P$ is a size-1 pattern and maxpcov<cov$(P)$ then
5.      maxpcov := cov$(P)$;
6.  if $P$ contains more than one edge then
7.      $\mathcal{S}$ := PSel$(P, \mathcal{S}, \mathcal{P}, r(x, y)$, maxpcov$)$;
8.      pattern $P$ := PGen$(G, \Gamma, \sigma, \theta)$.next();
9.  return $\mathcal{S}$;
```

Fig. 2. Algorithm GFC_stream

As a proof of Theorem 1, we next introduce such a stream discovery algorithm.

"Stream + Sieve". Our supervised discovery algorithm, denoted as GFC_stream (illustrated in Fig. 2), interleaves pattern generation and GFC selection as follows.

(1) *Pattern stream generation.* The algorithm GFC_stream invokes a procedure PGen to produce a pattern stream (lines 2, 8). In contrast to GFC_batch that verifies patterns against 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 enumeration and 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", initialized by the maximum coverage score of single patterns (Sect. 3) maxpcov $= \max_{P \in \mathcal{P}}(\text{cov}(P))$ (lines 4–5), and eagerly constructs GFCs with high marginal benefit that refines sieve values progressively.

The above 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 no longer be improved by PSel (as will be discussed). We next introduce the details of procedures PGen and PSel.

Procedure PGen. Procedure PGen improves its "batch" counterpart in GFC_batch by locally generating patterns that cover particular sets of facts, following a manner of decision tree construction. It maintains the following structures in each iteration i: (1) a pattern set $\mathcal{P}_i$, which contains graph patterns of size (number of pattern edges) i, and is initialized as a size-0 pattern that contains anchored nodes u_x and u_y only; (2) a partition set $\Gamma_i(P)$, which records the sets of facts $P(\Gamma^+)$ and $P(\Gamma^+)$, is initialized as $\{\Gamma^+, \Gamma^-\}$, for each pattern $P \in \mathcal{P}_i$. At iteration i, it performs the following.

(1) For each block $B \in \Gamma_{i-1}$, PGen generates a set of graph patterns $\mathcal{P}_i$ with size i. A size-i pattern P is constructed by adding a triple pattern $e(u, u')$ to its size-$(i-1)$ counterpart P' in $\mathcal{P}_{i-1}$. Moreover, it only inserts $e(u, u')$ with instances from the neighbors of the matches 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 procedure Verify of algorithm GFC_batch, 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 procedure PSel for the construction of top-k GFCs $\mathcal{S}$.

Procedure PSel**.** To compute the set of GFCs $\mathcal{S}$ that maximizes $\mathsf{cov}(\mathcal{S})$ for a given $r(x, y)$, it suffices for procedure PSel to compute top k graph patterns that maximize $\mathsf{cov}(\mathcal{S})$ 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 $F(\cdot)$, a constant $\epsilon > 0$ and element set $\mathcal{D}$, sieve-streaming finds top-k elements $\mathcal{S}$ that maximizes $F(\mathcal{S})$ 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 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 above sieve-streaming for GFCs seems infeasible: one needs to find the maximum $\mathsf{cov}(\varphi)$ (or $\mathsf{cov}(P)$ for fixed $r(x, y)$), which requires to verify the entire pattern set. Capitalizing on data locality of graph pattern matching, Lemma 2, and Lemma 1, we have good news.

Lemma 3. *It is in $O(|\Gamma_1|)$ time to compute the maximum* $\mathsf{cov}(P)$.

This can be verified by observing that $\mathsf{cov}(\cdot)$ also preserves anti-monotonicity in terms of pattern refinement. That is, $\mathsf{cov}(P') \leq \mathsf{cov}(P)$ if $P \preceq P'$. Thus, $\max_{P \in \mathcal{P}} \mathsf{cov}(P)$ is contributed by a single-edge pattern. That is, procedure PSel only needs to cache at most $|\Gamma_1|$ size-1 patterns from PGen to find the global maximum $\mathsf{cov}(P)$ (lines 4–5 of GFC_stream). The rest of PSel follows the sieve-streaming strategy, as illustrated in Fig. 3. The GFCs are constructed with the top-k graph patterns (line 8).

```
Procedure PSel($P, \mathcal{S}, \mathcal{P}, r(x,y)$, maxpcov)
1.  if $\mathcal{P} := \emptyset$ then
2.      set $\mathcal{S}_V := \{(1+\epsilon)^j | (1+\epsilon)^j \in [\mathsf{maxpcov}, k * \mathsf{maxpcov}]\}$;
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 $\mathsf{mg}(P, \mathcal{S}_{v_j}) \geq (\frac{v_j}{2} - \mathsf{cov}(\mathcal{S}_{v_j}))/(k - |\mathcal{S}_{v_j}|)$ 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.      $\mathcal{S} := \mathsf{constructGFCs}(\mathcal{S}_{v_j})$ $(j \in [1, \log_{(1+\epsilon)}(k * \mathsf{maxpcov})])$
9.  return $\mathcal{S}$;
```

Fig. 3. Procedure PSel

Optimization. To further prune unpromising patterns, procedure PGen estimates an upperbound $\hat{\mathsf{mg}}(P, \mathcal{S}_{v_j})$ (line 5 of PSel) without verifying a new size-i pattern P. If $\hat{\mathsf{mg}}(P, \mathcal{S}_{v_j}) < (\frac{v_j}{2} - \mathsf{cov}(\mathcal{S}_{v_j}))/(k - |\mathcal{S}_{v_j}|), P$ 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 sub-pattern of 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 $\hat{\mathsf{supp}}(\varphi) = \mathsf{supp}(\varphi') - \frac{l}{|r(\Gamma^+)|}$, where l is the number of the facts in $r(\Gamma^+)$ that have no instance of r' in their i hop neighbors (thus cannot be covered by P). Similarly, one can estimate an upper bound for p and n in the formula of $\mathsf{sig}(\cdot)$, and thus get an upper bound for $\hat{\mathsf{nsig}}(\varphi)$. For each t in Γ^+, denote term $\sqrt{\sum_{\varphi \in \Phi_t(\mathcal{S})} \mathsf{supp}(\varphi)}$ in $\mathsf{div}(\mathcal{S})$ as T_t, it then computes $\hat{\mathsf{mg}}(P, \mathcal{S})$ as $\frac{\hat{\mathsf{nsig}}(\varphi)}{2\sqrt{\mathsf{sig}(\mathcal{S})}} + \left(\sum_{t \in P(\Gamma^+)} \frac{\hat{\mathsf{supp}}(\varphi)}{2\sqrt{T_t}}\right) / |\Gamma^+|$. One may prove that this is an upper bound for $\hat{\mathsf{mg}}(P, \mathcal{S})$, by applying the inequality $\sqrt{\alpha+\beta} - \sqrt{\alpha} \leq \beta/(2\sqrt{\alpha})$ to each square root term in $\mathsf{sig}(\cdot)$ and $\mathsf{div}(\cdot)$. We found that this effectively reduces redundant verifications (see Sect. 5).

Performance Analysis. Denote 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 set $\mathcal{S}_v$. Thus the update time for each pattern is in $O((b + |\Gamma|_b)^2 + \frac{\log k}{\epsilon})$.

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

The above analysis completes the proof of Theorem 1.

4.2 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.

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 $e = <v_x, r, v_y>$, it follows "hit and miss" convention [12] and checks if there exists a GFC φ in $\mathcal{S}$ that covers e (*i.e.,* both its consequent and antecedent cover e). If so, GFact_R accepts e, otherwise, it rejects e.

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 $e = <v_x, r, v_y> \in \Gamma$, 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 e is *true* (resp. *false*) if $e \in \Gamma^+$ (resp. Γ^-).

By default, GFact adopts Logistic Regression, which is experimentally verified to achieve slightly better performance than others (*e.g.,* Naive Bayes and SVM). We find that GFact outperforms GFact_R over real-world graphs (See Sect. 5).

5 Experimental Study

Using real-world knowledge bases, we empirically evaluate the efficiency of GFCs discovery and the effectiveness of GFC-based fact checking.

Datasets. We used five real-world knowledge graphs, including (1) YAGO [33] (version 2.5), a knowledge base that contains $2.1M$ entities with 2273 distinct labels, $4.0M$ edges with 33 distinct labels, and $15.5K$ triple patterns; (2) DBpedia [19] (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 [35] (RDF dumps 20160801), a knowledge base that contains $10.8M$ entities with 18383 labels, $41.4M$ edges of 693 relations, and $209K$ triple patterns; (4) MAG [30], a fraction of an academic graph with $0.6M$ entities (*e.g.,* papers, authors, venues, affiliations) of 8565 labels and $1.71M$ edges of 6 relationships (cite, coauthorship), and (5) Offshore [16], a social network of offshore entities and financial activities, which contains $1M$ entities (*e.g.,* companies, countries, person) with 357 labels, $3.3M$ relationships (*e.g.,* establish, close) with 274 labels, and 633 triple patterns. We use Offshore mostly for case studies.

Methods. We implemented the following methods in Java: (1) algorithm GFC_stream, compared with (a) its "Batch + Greedy" counterpart GFC_batch (Sect. 4), (b) algorithm AMIE+ [11] that discovers AMIE rules, (c) PRA [6,18], the Path Ranking Algorithm that trains classifiers with path features from random walks, and (d) KGMiner [29], a variant of PRA that makes use of features from discriminant paths; (2) fact checking models GFact_R and GFact, compared with learned models (and also denoted) by AMIE+, PRA, and KGMiner, respectively. For practical comparison, we set a pattern size (the number of pattern edges) bound $b = 4$ for GFC discovery.

Model Configuration. For fair comparison, we made effort to calibrate the models and training/testing sets under consistent settings. (1) For the supervised link prediction methods (GFact, PRA, and KGMiner), we sample 80% of the facts in a knowledge graph as the training facts Γ, with instances of in total 107 triple patterns, and 20% edges as testing set $\mathcal{T}$. Each triple pattern has $5K$-$50K$ instances. In Γ (resp. $\mathcal{T}$), 20% are true examples Γ^+ (resp. $\mathcal{T}^+$), and 80% are false examples Γ^- (resp. $\mathcal{T}^-$). We generate Γ^- and $\mathcal{T}^-$ under PCA (Sect. 3) for all the models. For all methods, we use Logistic Regression to train the classifiers, same as the default settings of PRA and KGMiner. (2) 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 3, comparable to the number of pattern edges in our work.

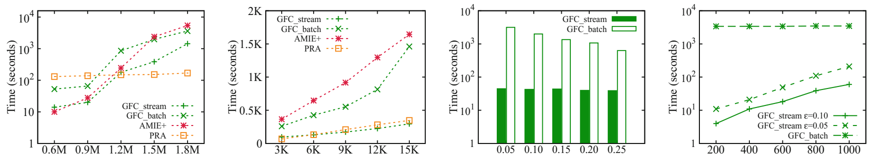
Overview of Results. We find the following. (1) It is feasible to discover GFCs in large graphs (**Exp-1**). For example, it takes 211 s for GFC_stream to discover GFCs over YAGO with 4 million edges and 3000 training facts. On average, it outperforms AMIE+ by 3.4 times. (2) GFCs can improve the accuracy of fact checking models (**Exp-2**). For example, it 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 with AMIE+, PRA, and KGMiner, respectively. (3) Our case study shows that GFact yields interpretable models (**Exp-3**).

We next report the details of our findings.

Exp-1: Efficiency. We report the efficiency of GFC_stream, compared with AMIE+, GFC_batch and PRA over DBpedia. As KGMiner has unstable learning time and is not comparable, the result is omitted.

Varying $|E|$. Fixing $|\Gamma^+| = 15K$, support threshold $\sigma = 0.1$, confidence threshold $\theta = 0.005, k = 200$, we sampled 5 graphs from DBpedia, with size (number of edges) varied from $0.6M$ to $1.8M$. Figure 4(a) shows that all methods take longer time over larger $|E|$, as expected. (1) GFC_stream is on average 3.2 (resp. 4.1) times faster than AMIE+ (resp. GFC_batch) due to its approximate matching scheme and top-k selection strategy. (2) Although AMIE+ is faster than GFC_stream over smaller graphs, we find that it returns few rules due to low support. Enlarging rule size (*e.g.,* to 5), AMIE+ does not run to completion. (3) The cost of PRA is less sensitive due to that it samples a (predefined) fixed number of paths.

Varying $|\Gamma^+|$. Fixing $|E| = 1.5M, \sigma = 0.1, \theta = 0.005, k = 200$, we varied $|\Gamma^+|$ from $3K$ to $15K$. As shown in Fig. 4(b), while all the methods take longer time for larger $|\Gamma^+|$, GFC_stream scales best with $|\Gamma^+|$ due to its stream selection strategy. GFC_stream achieves comparable efficiency with PRA, and outperforms GFC_batch and AMIE+ by 3.54 and 5.1 times on average, respectively.



(a) Varying $|E|$ (b) Varying $|\Gamma^+|$ (c) Varying σ (d) Varying k

Fig. 4. Efficiency of GFC_stream

Table 1. Effectiveness: average accuracy.

Model	YAGO				DBpedia				Wikidata				MAG			
	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1	Pred	Prec	Rec	F_1
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**
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+	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
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

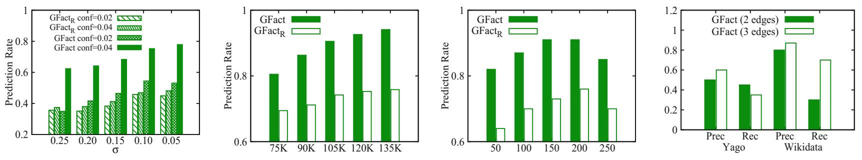
Varying σ. Fixing $|E| = 1M, \theta = 0.005, k = 200$, we varied σ from 0.05 to 0.25. As shown in Fig. 4(c), GFC_batch takes longer time over smaller σ, due to more patterns and GFC candidates need to be verified. On the other hand, GFC_stream is much less sensitive. This is because it terminates early without verifying all patterns.

Varying k. Fixing $G = 1.5M$, $\sigma = 0.1, \theta = 0.005$, we varied k from 200 to 1000. Figure 4(d) shows that GFC_stream is more sensitive to k due to 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 with larger ϵ, less number of patterns are needed, thus GFC_stream takes less time.

Exp-2: Accuracy. We report the accuracy of all the models in Table 1.

Rule-Based Models. We apply the same support threshold $\sigma = 0.1$ for AMIE+ and GFact$_R$. We set $\theta = 0.005$ for GFact$_R$, and set $k = 200$. We sample 20 triple patterns $r(x, y)$ and report the average accuracy. As shown in Table 1, GFact$_R$ constantly improves AMIE+ with up to 21% gain in prediction rate, and with comparable performance for other cases. We found that AMIE+ reports rules with high support but not necessarily meaningful, while GFCs capture more meaningful context (see Exp-3). Both models have relatively high recall but low precision, due to that they have better chance to cover missing facts but may introduce errors when hitting false facts.

Supervised Models. We next compare GFact with supervised link prediction models (Table 1). 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 outperforms KGMiner with 16% gain on precision and 19% recall. Indeed, GFact extracts useful features from GFCs with both high significance and diversity, beyond path features.



(a) Varying σ (b) Varying $|\Gamma^+|$ (c) Varying k (d) Varying # pattern edges

Fig. 5. Impact factors to accuracy

We next evaluate the impact of factors to the model accuracy using Wikidata.

Varying σ and θ. Fixing $|\Gamma^+| = 135K$, we varied σ from 0.05 to 0.25 for both GFact and GFact$_R$. We select 20 patterns with 0.02 confidence and 20 patterns with 0.04 confidence, respectively. Figure 5(a) shows that both GFact and GFact$_R$ has lower prediction rate when support threshold (resp. confidence) is higher (resp. lower). This is because fewer patterns can be discovered with higher support, leading to more "misses" in facts; while higher confidence lead to stronger association of patterns and more accurate predictions. In general, GFact achieves higher prediction rate than GFact$_R$.

Varying $|\Gamma^+|$. Fixing $\sigma = 0.01, \theta = 0.005, k = 200$, we vary $|\Gamma^+|$ from $75K$ to $135K$ as shown in Fig. 5(b). It tells us that GFact and GFact$_R$ have higher prediction rate with more positive examples. Their precisions (not shown) follow the similar trend.

Varying k. Fixing $\sigma = 0.01, \theta = 0.005, |\Gamma^+| = 2500$, we varied k from 50 to 250. Figure 5(c) shows the prediction rate first increases, and then decreases. For rule-based model, more rules increase the accuracy by covering more true facts, while increasing the risk of hitting false facts. For supervised link prediction, the model will be under-fitting with few features for small k, and will be over-fitting with too many features due to large k. We observe that $k = 200$ is a best setting for high prediction rate.

Varying b. Fixing $|E| = 4M, \sigma = 0.01, k = 1000$ and $\theta = 0.005$, we select 200 size-2 patterns and 200 size-3 patterns to train the models. Figure 5(d) verifies an interesting observation: smaller patterns contribute more to recall and larger patterns contribute more to precision. This is because smaller patterns are more likely to "hit" new facts, while larger patterns have stricter constraints for correct prediction of true fact.

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

Accuracy. We show 3 relations and report accuracy in Fig. 6. These relations are non-functional, and may contain incomplete subjects and objects, where PCA may not hold [11]. We found that GFact complements AMIE+ and mitigate such disruptions with context enforced by graph patterns.

$r(x,y)$	GFact		GFact_R		AMIE+	
	Pred	Prec	Pred	Prec	Pred	Prec
r_1(YAGO): *created*(director, movie)	0.81	0.56	0.33	0.13	0.39	0.12
r_2(DBpedia): *derived*(music, music)	0.81	0.53	0.75	0.40	0.32	0.22
r_3(Offshore): *serves*(person, company)	0.90	0.96	0.92	0.91	0.77	0.46

Fig. 6. Accuracy: case study.

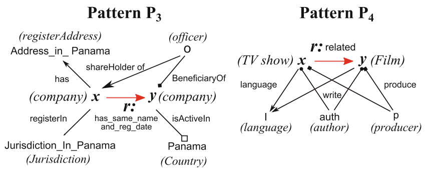


Fig. 7. Real-world GFCs discovered by GFact

Interpretability. We further illustrate two top GFCs in Fig. 7, which contribute to highly important features in GFact with high confidence and significance over a real-world financial network Offshore and DBpedia, respectively.

(1) GFC $\varphi_3 : P_3(x,y) \rightarrow \texttt{has\_same\_name\_and\_reg\_date}(x,y)$ states that two (anonymous) companies are likely to have the same name and registration date if they share shareholder and beneficiary, and one is registered and within jurisdiction in Panama, and the other is active in Panama. This GFC has support 0.12 and confidence 0.0086, and is quite significant. For the same $r(x,y)$, AMIE+ discovers a top rule as $\texttt{registerIn}(x$, Jurisdiction_in_Panama$) \wedge \texttt{registerIn}(y$, Jurisdiction_in_Panama$)$ implies x and y has the same name and registration date. This rule has a low prediction rate.

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

6 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 for observed true and false facts. 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. We are evaluating GFCs with real-world graphs and pattern models. One future topic is to develop scalable GFCs-based models and methods with parallel graph mining and distributed rule learning.

Acknowledgments. This work is supported in part by NSF IIS-1633629 and Huawei Innovation Research Program (HIRP).

References

1. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: massive data summarization on the fly. In: SIGKDD (2014)
2. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI (2010)
3. Chen, Y., Wang, D.Z.: Knowledge expansion over probabilistic knowledge bases. In: SIGMOD (2014)
4. Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A.: Computational fact checking from knowledge networks. PLoS One **10**, e0141938 (2015)
5. Cukierski, W., Hamner, B., Yang, B.: Graph-based features for supervised link prediction. In: IJCNN (2011)
6. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: KDD (2014)
7. Elseidy, M., Abdelhamid, E., Skiadopoulos, S., Kalnis, P.: GraMI: frequent subgraph and pattern mining in a single large graph. PVLDB **7**, 517–528 (2014)
8. Fan, W., Wang, X., Wu, Y., Xu, J.: Association rules with graph patterns. PVLDB **8**, 1502–1513 (2015)
9. Fan, W., Wu, Y., Xu, J.: Functional dependencies for graphs. In: SIGMOD (2016)
10. Finn, S., Metaxas, P.T., Mustafaraj, E., O'Keefe, M., Tang, L., Tang, S., Zeng, L.: TRAILS: a system for monitoring the propagation of rumors on Twitter. In: Computation and Journalism Symposium, New York City, NY (2014)
11. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. **24**, 707–730 (2015)
12. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.: AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In: WWW (2013)
13. Gardner, M., Mitchell, T.M.: Efficient and expressive knowledge base completion using subgraph feature extraction. In: EMNLP (2015)
14. Goodwin, T.R., Harabagiu, S.M.: Medical question answering for clinical decision support. In: CIKM (2016)
15. Hassan, N., Sultana, A., Wu, Y., Zhang, G., Li, C., Yang, J., Yu, C.: Data in, fact out: automated monitoring of facts by FactWatcher. VLDB **7**, 1557–1560 (2014)
16. ICIJ: Offshore dataset. https://offshoreleaks.icij.org/pages/database
17. Jiang, C., Coenen, F., Zito, M.: A survey of frequent subgraph mining algorithms. Knowl. Eng. Rev. **28**, 75–105 (2013)
18. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: EMNLP (2011)
19. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**, 167–195 (2015)
20. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: ACL/HLT (2011)
21. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI (2015)
22. Ma, S., Cao, Y., Fan, W., Huai, J., Wo, T.: Capturing topology in graph pattern matching. VLDB **5**, 310–321 (2011)

23. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-I. Math. Program. **14**, 265–294 (1978)
24. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**, 11–33 (2016)
25. Niu, F., Zhang, C., Ré, C., Shavlik, J.W.: Deepdive: web-scale knowledge-base construction using statistical learning and inference. VLDS **12**, 25–28 (2012)
26. Passant, A.: dbrec—music recommendations using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17749-1_14
27. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. Semant. Web **8**, 489–508 (2017)
28. Shao, C., Ciampaglia, G.L., Flammini, A., Menczer, F.: Hoaxy: a platform for tracking online misinformation. In: WWW Companion (2016)
29. Shi, B., Weninger, T.: Discriminative predicate path mining for fact checking in knowledge graphs. Knowl.-Based Syst. **104**, 123–133 (2016)
30. Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.j.P., Wang, K.: An overview of microsoft academic service (MAS) and applications. In: WWW (2015)
31. Song, C., Ge, T., Chen, C., Wang, J.: Event pattern matching over graph streams. VLDB **8**, 413–424 (2014)
32. Song, Q., Wu, Y.: Discovering summaries for knowledge graph search. In: ICDM (2016)
33. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge. In: WWW (2007)
34. Thor, A., Anderson, P., Raschid, L., Navlakha, S., Saha, B., Khuller, S., Zhang, X.-N.: Link prediction for annotation graphs using graph summarization. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 714–729. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_45
35. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**, 78–85 (2014)
36. Wang, Q., Liu, J., Luo, Y., Wang, B., Lin, C.Y.: Knowledge base completion via coupled path ranking. In: ACL (2016)
37. Wu, Y., Agarwal, P.K., Li, C., Yang, J., Yu, C.: Toward computational fact-checking. PVLDB **7**, 589–600 (2014)
38. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining significant graph patterns by leap search. In: SIGMOD (2008)