Provenance Summaries for Answers and Non-Answers

Seokki Lee
Illinois Institute of Technology
slee195@hawk.iit.edu

Bertram Ludäscher
University of Illinois, Urbana-Champaign
ludaesch@illinois.edu

Boris Glavic
Illinois Institute of Technology
bglavic@iit.edu

ABSTRACT

Explaining why an answer is (not) in the result of a query has proven to be of immense importance for many applications. However, why-not provenance, and to a lesser degree also why-provenance, can be very large, even for small input datasets. The resulting scalability and usability issues have limited the applicability of provenance. We present *PUG*, a system for why and why-not provenance that applies a range of novel techniques to overcome these challenges. Specifically, PUG limits provenance capture to what is relevant to explain a (missing) result of interest and uses an efficient sampling-based *summarization* method to produce compact explanations for (missing) answers. Using two real-world datasets, we demonstrate how a user can draw meaningful insights from explanations produced by PUG.

PVLDB Reference Format:
Seokki Lee, Bertram Ludäscher, Boris Glavic. Providing Provenance Summaries as Explanations for Answers and Non-Answers. *PVLDB*, 11 (12): 1954-1957, 2018.
DOI: https://doi.org/10.14778/3229863.3236233

1. INTRODUCTION

Provenance for relational queries [4] explains how results of a query depend on the query's inputs. Recently, provenance-like techniques have been applied to explain how missing inputs cause a tuple to be missing from a query's result (e.g., [5, 8]). In prior work, we have shown that why and why-not questions can be treated uniformly as provenance for first-order (FO) queries via non-recursive Datalog with negation. We have implemented these techniques in our *PUG* (*P*rovenance *U*nification through *G*raphs) [6] system.

Most approaches for why-not provenance [5, 6] enumerate all failed ways of how a result could have been derived. This type of provenance is often too large to compute, e.g., the why-not provenance graph over a small relation (1000s of tuples) may already consist of billions of nodes [6]. While why provenance is typically much smaller, it may still be too large to be explored manually. PUG overcomes these usability and scalability issues using novel techniques that we explain in the following.

r_1 : T(C) :- C(I, theft, L, C, Y), ¬ A(I)
r_2 : AL(L) :- C(I, arson, L, garfield ridge, Y), ¬ A(I)

Crimes (input)

Id	Type	Location	Community	Year
2446	theft	street	austin	2014
1046	arson	residence	garfield ridge	2016
3144	arson	alley	garfield ridge	2016
4255	theft	street	west lawn	2015
4302	theft	department store	west lawn	2017

Arrest (input)

Id
3144
4255

Thefts (output)

Community
austin
west lawn

Arson**L**oc (output)

Location
residence

Figure 1: Crime database, query r_1 and r_2, and outputs

Adjustable Level of Detail. Detailed types of provenance information are useful for understanding the intricacies of how an outcome came to be, but are potentially an overkill for the initial phase of exploration where a user just wants to get a rough understanding of what has happened. PUG supports several types of provenance that range from simple data dependencies to detailed provenance types which expose *how* results have been derived by a query.

Limit Provenance to What is Relevant. Typically, only a subset of the provenance (which we call an *explanation*) is relevant to explain the existence or absence of a query result. PUG generates explanations for a (missing) result by instrumenting the input query to capture only relevant provenance. This idea has also been employed by *SelP* [2]. The main difference to our work is that we support negation, but no recursion which *SelP* supports.

Sample-based Summarization of Provenance. While explanations are useful, the resulting provenance graphs may still overwhelm users with too much information and waste computational resources. Approaches for summarizing provenance (e.g., [1]) assume that the full provenance is available as an input to the summarization process. While these approaches address some usability issues of provenance, they fail to address scalability issues. PUG uses a sample-based technique to efficiently compute provenance summaries without generating the full (why-not) provenance.

In this demonstration, we present PUG's web-based provenance explorer which enables users to browse provenance summaries for (missing) query results of interest. Using

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236233

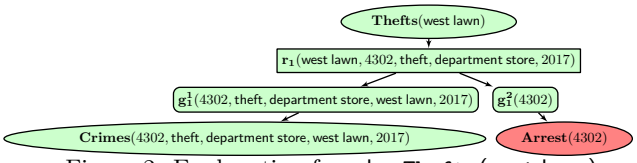

Figure 2: Explanation for why Thefts(west lawn)

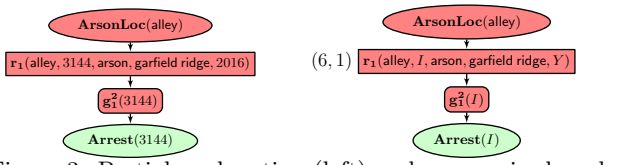

Figure 3: Partial explanation (left) and summarized explanation (right) for whynot ArsonLoc(alley)

Chicago crime[1] ($\sim$ 6M entries) and movielens data[2] ($\sim$ 20M entries), we will demonstrate how to gain new insights using provenance summaries. PUG generates such summaries within 10s of seconds for these datasets, even for why-not questions (preliminary experiments are presented in [7]).

Example 1. *Consider a simplified version of the crime dataset shown in Figure 1 (called* S-Crime *from now on). The* Crimes *table records for each crime an* id, type *and* location *(e.g., theft on a street),* community, *and* year. *Relation* Arrest *stores the* ids *of crimes that led to arrests. Consider the following scenario: Alice wants to identify measures for reducing thefts. She first runs query r_1 in Figure 1 to determine in which communities there were unarrested thefts. For* S-Crime, *r_1 returns the communities austin and west lawn, because some thefts in these communities (ids 2446 and 4302) have not led to arrests. Over the full dataset (which we refer to as* F-Crime*), this query returns all Chicago communities. Alice then tries a different angle by running a query to determine the most common locations of thefts. It turns out that streets account for* 2/3 *of all thefts in* S-Crime *and* $\sim$ 28% *in* F-Crime. *Based on this result, Alice recommends to increase police presence on public streets. With the exception of community west lawn, this turns out to be effective. Alice investigates this outcome by requesting PUG to explain* why Thefts(west lawn).

2. PROVENANCE-BASED EXPLANATIONS

PUG [6] supports several graph-based provenance models which correspond to well known provenance types from the literature. Here, we limit the discussion to two of these types. For positive queries, the most informative type of provenance graphs in PUG corresponds the most general form of provenance in the semiring framework. This type of graph consists of *rule nodes* (boxes with a rule-id and the constant arguments of a rule derivation), *goal nodes* (rounded boxes with a rule-id and the goal's position in the rule body), and *tuple nodes* (ovals). Nodes are either *green* (successful/existing) or *red* (failed/missing). The second type of graph we consider here only records tuple dependencies (only contains tuple nodes). For positive queries, this graph type corresponds to Lineage. Given a question about the existence (absence) of a result, our system generates an *explanation*, i.e., a subgraph of the provenance graph for a query that contains only facts and rule derivations relevant for deriving (or failing to derive) the result.

Example 2. *Continuing with Example 1, Alice requested an explanation for the result* Thefts(west lawn). *This tuple is in the result because some theft(s) in west lawn have not led to arrests yet. In* S-Crime *(Figure 1), there is one such theft, the highlighted crime with id 4302. The provenance graph returned by PUG (Figure 2) contains a single rule node representing the successful derivation of the result through rule r_1. A successful rule node is connected to successful goal nodes (e.g., r_1 is connected to g_1^1, the first goal in the rule's body). Successful positive goals are connected to nodes that represent existing tuples (green) while successful negated goals are connected to missing tuples (red). Figure 5 shows a provenance graph that records data dependencies for Alice's question. This is one of the provenance types supported in PUG.* Thefts(west lawn) *is in the result, because a theft happened in west lawn (the* Crimes *tuple shown on the left), but no arrest was made (the corresponding* Arrest *tuple does not exist). Alice, using* F-Crime *($\sim 6M$ entries), faces the problem that the provenance contains* 7,984 *rule derivations (i.e.,* 7,984 *unarrested thefts in west lawn). The size of this graph motivates the need for summarization.*

Missing Answers. A user may also want to understand why an expected result is not returned. A provenance graph for a missing answer enumerates all potential ways of how the result *could* have been derived by the query's rules and explains for each such alternative *why it failed*. In the provenance graph, only failed goals (which are causes for the failure of a derivation) are connected to the failed rule derivations explaining missing answers. For why-not questions and negation, we allow the user to associate a domain with each attribute in the database. Missing tuples are constructed using values from these associated domains (see [6]).

Example 3. *Alice runs query r_2 from Figure 1 to determine the locations of unsolved arsons in garfield ridge. She knows that several arsons occurred in alleys in this community. She is surprised not to see* alley *in the result and, thus, requests PUG to explain* whynot ArsonLoc*(alley). Part of the explanation for* S-Crime *is shown in Figure 3 (left). The full provenance graph for the example would already contain more than 60 nodes. The subgraph shown in this figure records that there was an arson with id 3144 in an alley in garfield ridge. However, the perpetrator was arrested. That is, the existence of the tuple* Arrest(3144) *caused the rule derivation to fail. For* F-Crime, *the provenance graph contains $\sim 18 \cdot 10^9$ nodes, which would be of little use to Alice, further underlining the need for summarization.*

3. SUMMARIZING PROVENANCE

We address the computational and usability challenges of large provenance graphs by creating summaries based on structural commonalities in the provenance. An initial version of these techniques was presented in [7]. To produce a summary from a provenance graph, we identify subgraphs (derivations) that share a common structure and some common data values, and replace such subgraphs with patterns.

[1] https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
[2] https://grouplens.org/datasets/movielens/20m/

Derivation Patterns. A derivation pattern is a mapping from the variables of a rule to constants and variables. Recall that in PUG the user can associate domains with attributes. These domains are used to define what derivations are represented by a pattern. For simplicity, assume a fixed domain $\mathbb{D}$ for all attributes. Consider a pattern p and let $\bar{X}$ denote the set of variables in p. A valuation ν for p is a mapping from $\bar{X}$ to the elements of $\mathbb{D}$. We use $\nu(p)$ to denote the result of replacing the variables in p based on ν. For a pattern p and a derivation d, we say that p *matches* d if there exists a valuation ν such that $\nu(p) = d$. A pattern p represents the set of all derivations that match p. Note that a rule derivation corresponds to a subgraph in the provenance. Thus, a derivation pattern represents a set of graph fragments - one for each derivation that matches the pattern. For instance, Figure 4 shows a provenance summary corresponding to a single pattern that contains two variables I and Y representing the crime id and year, respectively. One valuation of this pattern is $I = 4302$ and $Y = 2017$ corresponding to the provenance graph in Figure 2.

Measuring Pattern Quality. We measure the quality of a pattern using: 1) **recall**: how much provenance does it cover, and 2) **informativeness**: how much new information is provided by the pattern. To define recall, we have to decide what derivations are considered to be provenance. For why questions, these are all successful derivations. For why-not questions, we group failed rule derivations into subsets based on which goals in their body are failed. The user can (optionally) designate one of these subsets as provenance (e.g., only the first goal is failed). We use $\#Prov(p)$ to denote the number of derivations in the provenance matching pattern p, and $\#Prov$ to denote the total number of derivations in the provenance. For 2), we use $arity(p)$ to denote the arity a pattern p. Intuitively, variables in patterns are not informative since they do not provide any information about the data contained in the provenance. Thus, we consider patterns with more constants to be more informative. However, constants which come from the query or user question have to be part of the pattern and, thus, also do not convey any new information. For a pattern p, we use $C(p)$ to denote the number of arguments of p that are constants and $QC(p)$ to denote the number of arguments that store constants which come from the query or user question. For example, for pattern $p_1 = r_1$(west lawn,I,theft,department store,Y) from Figure 4, we have $arity(p_1) = 5$, $C(p_1) = 3$ (west lawn, theft, and department store), and $QC(p_1) = 2$ (west lawn and theft were provided by the user). Using this notation, we define the recall $\text{rc}(p)$ and informativeness $\text{info}(p)$ of a pattern p:

$$\text{rc}(p) = \frac{\#Prov(p)}{\#Prov} \qquad \text{info}(p) = \frac{C(p) - QC(p)}{arity(p) - QC(p)}$$

Generating Pattern-based Summaries. Ideally, we want provenance summaries to be *concise* (small graphs), *complete* ($\text{rc}(p) = 1.0$), and *informative* ($\text{info}(p) = 1.0$). Obviously, fulfilling all three criteria at the same time is not always possible. Consider two extreme cases: 1) any provenance graph is also a provenance summary (one without variables). Provenance graphs are complete and informative, but not concise; 2) at the other end of the spectrum, we may represent an arbitrary number of derivations of a rule r as a single pattern resulting in a maximally concise summary with recall 1.0. However, unless these rule derivations

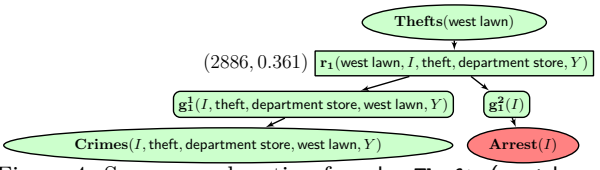

Figure 4: Summ. explanation for why Thefts(west lawn)

are quite homogeneous in terms of constants, such a summary will not be informative at all. The approach employed by PUG [7] guarantees conciseness by returning a summary that contains a configurable number of patterns and tries to maximize recall and informativeness. Specifically, we generate a set of candidate patterns from a representative sample of the provenance, and then return a summary that consists of the top-k patterns ranked based on their geometric mean of recall and informativeness. Note that quality measures in PUG are configurable. Currently, we support any weighted combination of recall and informativeness, but plan to implement additional quality metrics in the future. PUG's candidate generation step is similar to how [3] generates candidate patterns from a dataset. However, a major difference is that we face the additional challenge that it may be computationally infeasible to compute the full provenance as an input to summarization. PUG's provenance sampling technique addresses this challenge.

Example 4. *Given the large size of the explanation for the question* why Thefts(west lawn) *(Example 2), Alice requests PUG to generate a provenance summary for this question. The subgraph of the summary generated by PUG corresponding to the top-1 pattern* $p_1 = r_1$(west lawn,I,theft,department store,Y) *is shown in Figure 4. The pattern* p_1 *represents any derivation of the form* r_1(west lawn,i,theft,department store,y) *for some crime id* i *and year* y. *Alice is surprised to find that* $\sim 36\%$ *of all thefts in* west lawn *happened in department stores (pattern* p_1*) and only* $\sim 15\%$ *happened on public streets (the 2nd ranked pattern omitted from Figure 4). Note that PUG shows the match count and the recall for each pattern (*$(2886, 0.361)$ *in the figure). Given this result, it is obvious why her original recommendation was not effective in* west lawn. *Based on this new insight, Alice recommends to increase surveillance of stores in* west lawn.

Sampling for Approximate Summarization. To generate a summary of a provenance graph for a user question, we need to access successful and failed rule derivations to generate candidate patterns and compute their recall. Instead of generating the full set of derivations which may be computationally infeasible, PUG applies sampling during provenance capture. When generating the sample, we want to ensure that 1) it contains derivations that match representative patterns and 2) it allows us to closely approximate the actual recall. The approach applied by PUG is based on the observation that the union of the successful and failed derivations of a rule r is the cross-product of the associated domains of the attributes accessed by r. Thus, we can compute the number of failed derivations by computing the size of the cross-product of the domains and subtracting the number of successful derivations. Furthermore, the number of the successful derivations is typically several orders of magnitude smaller than the number of failed derivations. Hence, uniform random sampling from the cross-product will yield

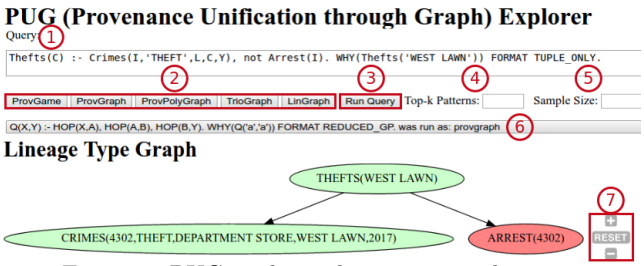

Figure 5: PUG explorer showing an explanation

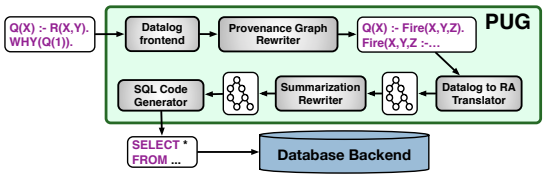

Figure 6: PUG system overview

an unbiased sample of failed derivations with high probability. PUG's sampling algorithm exploits these observations to efficiently generate a representative sample without having to compute all failed rule derivations.

Example 5. *Assume Alice wants to understand whether* alley *is missing from the query result because all arsons in alleys in* garfield ridge *have led to arrests. She requests PUG to summarize the provenance for* whynot `ArsonLoc(alley)` *restricted to the case where only the second goal failed (there was such a crime, but the crime led to an arrest). Alice configures the quality measure to put more weight on recall. Figure 3 (right) shows the summary for this question over* F-Crime. *PUG returns a summarized explanation in about 20 sec (sampling 1000 tuples). This pattern has recall* 100% *((6, 1) in the figure), confirming Alice's conjecture.*

4. THE PUG PROVENANCE EXPLORER

Explorer Interface. Figure 5 shows a screenshot of our PUG explorer. Queries optionally followed by a provenance question are entered into a textbox *Query* (①). The system shows the list of relations in the database on the right-side of the GUI to help the user formulate queries (not shown in the screenshot). Given a provenance question (e.g., `WHY(Thefts(west lawn))`), the user can select the type of provenance graph (②) or manually specify it as part of the request (①). As mentioned above, we support multiple graph types that differ in complexity and informativeness. Furthermore, the user can request PUG to return a summary by entering the number of top ranked patterns that should be included in the summary (④) and the sample size (⑤). For why-not questions, the user can optionally provide the failure pattern (which goals failed) of interest using `FOR FAILURE OF` $(b_1 \ \ldots \ b_m)$ where each b_i is a boolean value indicating whether we are only interested in derivations where the i^{th} goal failed. We maintain a history of user requests. The provenance graph for any of these requests can be restored from the dropdown menu (⑥). The *Run Query* button (③) is used to evaluate a query.

System Implementation. The web-based explorer sends user requests to PUG [6] (available as open source at https://github.com/IITDBGroup/PUG). An overview of PUG's architecture is shown in Figure 6. PUG features a parser for Datalog enriched with provenance requests. The system instruments the input Datalog program query to capture provenance relevant to the user question (see [6] for details). Afterwards, the instrumented Datalog program is translated into relational algebra. If summarization is requested, PUG adds additional instrumentation to generate a provenance summary. The resulting algebra expression is then translated into SQL and executed using a standard relational database backend. The instrumented query computes the edge relation of the provenance graph. We render the result using *graphviz* (https://www.graphviz.org/).

5. DEMONSTRATION OVERVIEW

We will demonstrate the functionality of PUG by executing example queries (e.g., r_1 and r_2 in Figure 1) over the real-life crime and movie datasets ($\sim$ 6M and $\sim$ 20M entries, respectively). Using subsets of these datasets, we first generate detailed explanations for why (e.g., Figure 2) and why-not (e.g., left in Figure 3) questions over the example queries. Afterwards, we generate summarized explanations for the same questions over the full datasets (e.g., Figure 4 and right in Figure 3) to demonstrate the scalability and usability of our sample-based summarization technique. Based on the summaries returned by PUG, we explain the meaning of top-k patterns and what new insights can be derived from them. If interested, attendees can also ask their own provenance questions over our example queries or write their own queries over the crime and movie datasets.

6. CONCLUSIONS

We present a web-based provenance explorer that allows users to browse (summarized) explanations for why and why-not provenance questions. The explorer uses our open-source PUG system to capture parts of the provenance that are relevant for explaining a (missing) answer. PUG integrates sample-based summarization with provenance capture to efficiently produce meaningful and compact explanations.

Acknowledgments. This work was supported by NSF Awards OAC-{1640864,1541450} and SMA-1637155. Opinions and findings expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

7. REFERENCES

[1] D. Deutch, N. Frost, and A. Gilad. Provenance for natural language queries. *PVLDB*, 10(5):577–588, 2017.

[2] D. Deutch, A. Gilad, and Y. Moskovitch. Selective provenance for datalog programs using top-k queries. *PVLDB*, 8(12):1394–1405, 2015.

[3] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, and D. Srivastava. Interpretable and informative explanations of outcomes. *PVLDB*, 8(1):61–72, 2014.

[4] T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[5] M. Herschel and M. Hernandez. Explaining Missing Answers to SPJUA Queries. *PVLDB*, 3(1):185–196, 2010.

[6] S. Lee, S. Köhler, B. Ludäscher, and B. Glavic. A SQL-middleware unifying why and why-not provenance for first-order queries. In *ICDE*, pages 485–496, 2017.

[7] S. Lee, X. Niu, B. Ludäscher, and B. Glavic. Integrating approximate summarization with provenance capture. In *TaPP*, 2017.

[8] A. Meliou, W. Gatterbauer, K. Moore, and D. Suciu. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *PVLDB*, 4(1):34–45, 2010.