Learning Path Queries on Graph Databases

Radu Ciucanu

joint work with
Angela Bonifati and Aurélien Lemay

University of Lille, France
INRIA Lille – Nord Europe
Links Project

EDBT’15
March 24, 2015
Motivation

- Specifying a database query is a challenging task for non-expert users.
  - **Unfamiliar** with language formalisms.

- In the context of **graph databases**, the problem becomes even harder:
  - There is no clear distinction between instances and **schemas**.
  - The instances do not carry proper **metadata**.
  - The instances are usually of large **size** and difficult to visualize.

- Traditional query specification paradigms for non-expert users e.g., query by example\(^1\) become unfeasible.

---

\(^1\)Zloof. Query by example. *AFIPS’75.*
Table of contents

1 Motivating example

2 Learning from a set of examples

3 Learning from user interactions

4 Interactive path query specification
Graph databases

- A **graph database** is a directed, edge-labeled graph.
- Popular in Semantic Web, social networks, scientific, biological and geographical applications, etc.
Path queries on graph databases

- We focus on **path queries** that select nodes having at least one path in the language of a given **regular expression**.
- Example: “select neighborhoods from which one can reach a cinema via public transportation.”

\[(\text{tram} + \text{bus})^* \cdot \text{cinema}\]

\[
\begin{align*}
N_1 \xrightarrow{\text{tram}} N_4 \xrightarrow{\text{cinema}} C_1 \\
N_2 \xrightarrow{\text{bus}} N_1 \xrightarrow{\text{tram}} N_4 \xrightarrow{\text{cinema}} C_1 \\
N_4 \xrightarrow{\text{cinema}} C_1 \\
N_6 \xrightarrow{\text{cinema}} C_2
\end{align*}
\]
Learning path queries on graph databases

- **Input:** positive and negative node examples.
- **Output:** the query that “the user has in mind.”

**Consistent queries:**

\[
(tram + bus)^* \cdot cinema
\]

\[
bus
\]

\[
\vdots
\]
Contributions

Learning from a fixed set of examples

- Polynomial learning algorithm based on grammatical inference.
- Learnability result for path queries.

Learning from user interactions

- Characterization of what means for a node to be informative.
- Practical strategies of presenting nodes to the user.
- Experimental evaluation on real biological and synthetic datasets.

From query learning to query specification

- Validation of the paths of interest by the user.
- A system for interactive path query specification on graphs.
Learning from a set of examples
Learning algorithm

Idea

- For each positive node select the path that “made the user label it.”
- Construct the disjunction of such paths.
- Generalize consistently with the examples.
Learning algorithm

### Idea
- For each positive node select the path that “made the user label it.”
- Construct the disjunction of such paths.
- Generalize consistently with the examples.

### Step 1 – Selecting smallest consistent paths (SCPs)
- For each positive node select its **smallest consistent path (SCP).**

\[ \nu_1 \rightarrow abc \]
\[ \nu_3 \rightarrow c \]
Learning algorithm

Step 2 – Generalizing SCPs by state merges

\[ \nu_1 - abc \]

\[ \nu_3 - c \]

---

Bound the length of the smallest consistent paths (SCPs)

**Problem** – Inconsistent sample.

- We may enumerate an infinite set of paths and never halt.
Bound the length of the smallest consistent paths (SCPs)

**Problem** – Inconsistent sample.
- We may enumerate an infinite set of paths and never halt.

**Consistency checking** is intractable:
- **PSPACE-complete** in general.
- **NP-complete** for restrictions (queries of the form $a_1 \cdots a_n$).
- Proof techniques from **definability** problems$^1$ (binary semantics).

**Solution** – Bound the length of the SCPs.

---

Learnability result

Learning algorithm

1. Select SCPs shorter than a fixed parameter $k$ if they exist.
2. Generalize SCPs.

Assuming that $k$ is fixed, the algorithm is polynomial:

- It returns a consistent query or it abstains from answering.
Learnability result

Learning algorithm

1. Select SCPs shorter than a fixed parameter $k$ if they exist.
2. Generalize SCPs.

Assuming that $k$ is fixed, the algorithm is polynomial:

- It returns a consistent query or it abstains from answering.

Main theoretical result (formally proven in the paper)

For every path query $q$, there exists a graph and a polynomial set of examples (characteristic sample) that guarantees that the algorithm learns $q$ in polynomial time.

- Definition inspired by grammatical inference\(^1\).

What happens without a characteristic sample?

The learning algorithm infers a query that is equivalent on the graph and **indistinguishable** by the user.

**Example**

Goal query: \((a \cdot b)^* \cdot c\)

Learned query: \(a\)
Learning from user interactions
Workflow of the interactive scenario

**Input:** a graph database (empty set of examples).

Is there any **informative** node left?

- **Yes**
  - Choose node $\nu$ w.r.t. a **strategy**.
  - Get neighborhood for $\nu$.
  - Ask label for $\nu$.

- **No**
  - **Output:** learned query.
  - Learn a query from all labels.
  - Propagate label for $\nu$.

What means for a node to be **informative**?

What is a good **strategy** of presenting nodes to the user?
Informative and uninformative nodes

A node is **uninformative** with a label if labeling it otherwise leads to an inconsistent sample.

<table>
<thead>
<tr>
<th>Uninformative with +</th>
<th>Uninformative with −</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Informative nodes</th>
<th>Informative nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Complexity**

Deciding whether a node is informative is PSPACE-complete.
Practical strategies

Idea

- Look at \( k \)-paths – paths of length bounded by \( k \).
- A node is \( k \)-uninformative if all its \( k \)-paths are covered by negatives.
  - no \( \rightarrow \) the node is informative and becomes a candidate next node.
  - yes \( \rightarrow \) the current \( k \) does not permit to decide the informativeness.

Strategies

1. A **randomly** chosen \( k \)-informative node.
2. A node with **minimal** number of non-covered \( k \)-paths (hope \( + \)).
3. A node with **maximal** number of non-covered \( k \)-paths (hope \( - \)).
4. A node with **average** number of non-covered \( k \)-paths (compromise).
Experiments

Scenarios

- **Static** experiments
  - Take randomly some nodes, label them, and run algorithm on them.
  - Measure the **F1 score** and the **learning time**.

- **Interactive** experiments
  - Start with an empty set of examples.
  - Measure the **number of examples** (+/−) and the **time** necessary for F1 score 1.
Experiments

Scenarios

- **Static** experiments
  - Take randomly some nodes, label them, and run algorithm on them.
  - Measure the **F1 score** and the **learning time**.

- **Interactive** experiments
  - Start with an empty set of examples.
  - Measure the **number of examples** (+/−) and the **time** necessary for F1 score 1.

Datasets

- **Biological** – graph of 3k nodes, 6 queries from biological research.
- **Synthetic**:  
  - Generate **scale-free** graphs (as Internet, social, and biological graphs).
  - Varying sizes: 10k, 20k, 30k.

Queries

- Of the form $A \cdot B^* \cdot C \cdot \ldots$  
  ($A, B, C$ are disjunctions, sometimes with overlapping symbols).
Experimental results – biological dataset

[Graphs showing F1 score and learning time for different biological datasets with and without interactions.]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>F1 Score 0</th>
<th>F1 Score 1</th>
<th>Learning Time 0</th>
<th>Learning Time 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bio1</td>
<td>7%</td>
<td>0.06%</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>Bio2</td>
<td>7%</td>
<td>1.78%</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>Bio3</td>
<td>66%</td>
<td>1.24%</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>Bio4</td>
<td>12%</td>
<td>1.32%</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>Bio5</td>
<td>87%</td>
<td>7.7%</td>
<td>3.45</td>
<td></td>
</tr>
<tr>
<td>Bio6</td>
<td>12%</td>
<td>1.18%</td>
<td>0.24</td>
<td></td>
</tr>
</tbody>
</table>
Experimental results – biological dataset

### Biological Dataset

#### Labels needed for F1 score = 1

<table>
<thead>
<tr>
<th>Bio query</th>
<th>F1 score without interactions</th>
<th>F1 score with interactions</th>
<th>Time between interactions (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bio1</td>
<td>7%</td>
<td>0.06%</td>
<td>0.19</td>
</tr>
<tr>
<td>Bio2</td>
<td>7%</td>
<td>1.78%</td>
<td>0.26</td>
</tr>
<tr>
<td>Bio3</td>
<td>66%</td>
<td>1.24%</td>
<td>0.34</td>
</tr>
<tr>
<td>Bio4</td>
<td>12%</td>
<td>1.32%</td>
<td>0.23</td>
</tr>
<tr>
<td>Bio5</td>
<td>87%</td>
<td>7.7%</td>
<td>3.45</td>
</tr>
<tr>
<td>Bio6</td>
<td>12%</td>
<td>1.18%</td>
<td>0.24</td>
</tr>
</tbody>
</table>
Experimental results – synthetic dataset

<table>
<thead>
<tr>
<th>Time between interactions (seconds)</th>
<th>F1 score with interactions</th>
<th>F1 score without interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>51%</td>
<td>0.15%</td>
</tr>
<tr>
<td>20000</td>
<td>26%</td>
<td>0.07%</td>
</tr>
<tr>
<td>30000</td>
<td>22%</td>
<td>0.04%</td>
</tr>
</tbody>
</table>

Radu Ciucanu (INRIA Lille)
Experimental results – synthetic dataset

<table>
<thead>
<tr>
<th>Synthetic graph size</th>
<th>Labels needed for ( F1 ) score = 1 without interactions</th>
<th>Labels needed for ( F1 ) score = 1 with interactions</th>
<th>Time between interactions (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>51%</td>
<td>0.15%</td>
<td>1.33</td>
</tr>
<tr>
<td>20000</td>
<td>26%</td>
<td>0.07%</td>
<td>5.83</td>
</tr>
<tr>
<td>30000</td>
<td>22%</td>
<td>0.04%</td>
<td>13.5</td>
</tr>
</tbody>
</table>

Radu Ciucanu (INRIA Lille)
Learning Path Queries on Graph Databases
EDBT’15
Interactive path query specification
From query learning to query specification

Goal: \((\text{tram} + \text{bus})^* \cdot \text{cinema}\)

SCP for \(N_2\): \(\text{bus}\)
SCP for \(N_6\): \(\text{bus}\)
Learned query: \(\text{bus}\)

It may not be equivalent to the user’s query on a different graph.
From query learning to query specification

Goal: \((\text{tram} + \text{bus})^* \cdot \text{cinema}\)

SCP for \(N_2\): \text{bus}
SCP for \(N_6\): \text{bus}
Learned query: \text{bus}

It may not be equivalent to the user’s query on a different graph.

Solution – an additional type of interaction – **path validation**

If the used validates
– for \(N_2\): \text{bus} \cdot \text{tram} \cdot \text{cinema}
– for \(N_6\): \text{cinema}

their generalization is the goal \((\text{tram} + \text{bus})^* \cdot \text{cinema}\).
Workflow for interactive query specification

**Input:** a graph database (empty set of examples).

Is there any **informative** node left?

- **Yes:**
  - Choose node $\nu$ w.r.t. a **strategy**.
  - Get neighborhood for $\nu$.
  - Ask label for $\nu$.

- **No:**
  - **Output:** learned query.

**Actions:**
- learn a query from all labels
- propagate label for $\nu$
- visualize
- ask for zoom
- label $\nu$ with + or −
- validate relevant path

Radu Ciucanu (INRIA Lille)  Learning Path Queries on Graph Databases  EDBT'15  23 / 25
Our system

**GPS** – “a system for interactive Graph Path query Specification”

Three **types of interactions** between the user and the system:

1. The user provides a fixed set of examples.
2. The user interactively labels +/− examples.
3. The user interactively labels +/− examples and **validate paths**.  
   - We propose a path and the user can validate or **correct** it.
Our system

GPS – “a system for interactive Graph Path query Specification”

Three **types of interactions** between the user and the system:

1. The user provides a fixed set of examples.
2. The user interactively labels +/– examples.
3. The user interactively labels +/– examples and **validate paths**.
   - We propose a path and the user can validate or **correct** it.

Additional **features**:

1. **Zoom out** the neighborhood of a proposed node.
2. **Visualize** the current **query** and run it on the graph.

**EDBT’15 System Demo.**
Conclusions and future work

Contributions

- We studied the problem of learning path queries on graphs:
  - Learnability result based on grammatical inference.
  - Practical strategies of interactively presenting nodes to the user.
  - Experimental evaluation on biological and synthetic datasets.

- We developed a system for interactive path query specification.
  - EDBT’15 System Demo.

Directions of future work

- Design a benchmark for graph queries defined by regular expressions.
- Sample the initial graph and learn on a representative subgraph.