Learning Queries for Relational, Semi-structured, and Graph Databases

Radu Ciucanu

University of Lille & INRIA, France
Supervised by Angela Bonifati & Sławek Staworko

SIGMOD’13 PhD Symposium
June 23, 2013
Table of contents

1. Introduction
2. Extending twig learning
3. Learning relational and graph queries
4. Conclusions
Introduction

- Web applications store their data within various database models:
  - Relational,
  - Semi-structured,
  - Graph databases.

- We study learning algorithms for queries for these models.

- We aim to apply the results to learn cross-model database mappings.
Introduction

Query learning algorithm

**Input:** A database instance annotated by the user.

**Output:** A query consistent with the user’s annotations.
Introduction

Query learning algorithm

**Input:** A database instance annotated by the user.

**Output:** A query consistent with the user’s annotations.

There are several parameters to consider for the learning algorithms:

- **Type of annotations:**
  - Positive examples only,
  - Both positive and negative examples.

- **Learning protocol:**
  - There is a fixed set of examples,
  - The algorithm interactively ask the user to annotate more examples.
Introduction

- We assume large database instances, so the algorithms should minimize the amount of effort that the user must invest.
- We require our algorithms to be able to learn from a small number of annotations.
- Depending on the existing results in the literature:
  - XML queries – extend existing algorithms.
  - Relational and graph queries – investigate an interactive framework.
Extending twig learning
Learning XML queries

Learning XML queries and transformations have been studied for the task of data extraction.

- *n*-ary XML queries captured with tree automata [Carme et al. ’07], [Lemay et al. ’06],
- Tree transformations captured with tree transducers [Lemay et al. ’10].

Tree automata and transducers are very expressive, but in practice XPath and XQuery are in widespread use.

- **Learning twig queries** [Staworko and Wieczorek ’12].
Learning twig queries from positive examples

Figure: Annotation of a fragment of the DBLP database.
Learning twig queries from positive examples

The algorithms proposed in [Staworko and Wieczorek ’12] are able to learn from few examples (generally 2), but they:

- Do not take advantage of the schema,
- Use only positive examples because consistency checking for positive and negative examples is intractable.

We are interested in extending their algorithms in several directions:
- Take advantage of the schema to improve the quality of the learned queries,
- Consider richer query classes e.g., union of twigs, for which the consistency checking is in PTIME,
- Approximate learning i.e., the learned query may select some negative examples and omit some of the positive ones.
Learning twig queries from positive examples

The algorithms proposed in [Staworko and Wieczorek ’12] are able to learn from few examples (generally 2), but they:

- Do not take advantage of the schema,
- Use only positive examples because consistency checking for positive and negative examples is intractable.

We are interested in extending their algorithms in several directions:

- Take advantage of the schema to improve the quality of the learned queries,
- Consider richer query classes e.g., union of twigs, for which the consistency checking is in PTIME,
- Approximate learning i.e., the learned query may select some negative examples and omit some of the positive ones.
Overspecialization

- The documents typically follow a schema: DTD, XML Schema,
- The algorithms may return queries which include fragments implied by the schema.

![Diagram showing the schema and examples of overspecialization]

**Figure:** Overspecialization.
Schema formalisms for unordered XML

- Since the twig queries are order-oblivious, we are interested in schema formalisms for unordered XML.
- The schema formalisms in widespread use, DTD and XML Schema, are better fitter towards ordered content.

We propose a new schema formalism: DMS

\[
\begin{align*}
    dblp & \rightarrow \text{article}^* \parallel \text{book}^* \parallel \text{inproceedings}^* \\
    \text{article} & \rightarrow \text{title} \parallel \text{year}? \parallel \text{author}^+ \\
    \text{inproceedings} & \rightarrow \text{title} \parallel \text{year}? \parallel \text{author}^+ \\
    \text{book} & \rightarrow \text{title} \parallel \text{year}? \parallel (\text{author}^+ \mid \text{editor}^+) \\
    \ldots & \rightarrow \#\text{PCDATA}
\end{align*}
\]
Learning twig queries in the presence of the schema

- Investigate unordered schema learning:
  - Previous research on schema learning consists typically on learning restricted classes of regular expressions,
  - We developed learning algorithms for DMS.

- Use the inferred DMS to avoid overspecialization:
  - Study query minimization in the presence of DMS,
  - Propose improved learning algorithms.
Learning relational and graph queries
Related work on learning relational queries

- **BP-completeness**: given a pair of relational instances, decide whether there exists a relational algebra expression which maps the first instance to the second one [Bancilhon '78], [Paredaens '78].
- The results were later extended to the nested relational model [Van Gucht '87] and sequences of pairs [Fletcher et al. '09].
- **Query by output**: given a database instance and the output of some query, construct an equivalent query to the initial one [Tran et al. '09].
- **View definitions**: given a database instance and a corresponding view instance, find the most succinct view definition [Das Sarma et al. '10].
- **Conditional functional dependencies** [Fan et al. '11].
- **Inductive logic programming** for relational algebra expressions [Gillis and Van den Bussche '09].
Our approach

- We assume a large database instance and a user annotating it.
- We want to develop an interactive framework where we ask the user to annotate tuples.
- There are several parameters to consider:
  - The query language (natural join, semijoin, chain of joins).
  - Presence of NULL values in the database.
  - Knowledge of integrity constraints, which can be used to boost the inference of the goal query (key dependencies, inclusion dependencies).
Preliminary results

- We investigated an interactive framework for learning *natural joins* and *semijoins*.
- We studied the following decision problems:
  - whether a tuple is *uninformative* w.r.t. the previously annotated tuples,
  - consistency checking of a set of positive and negative examples,
  - minimality of a learned query w.r.t. the database instance.
- The problems are tractable for natural joins, but they become intractable for semijoins.
Learning graph queries

- Learning graph queries is a novel research topic.
- We want to investigate an interactive framework similar to our approach for the relational queries.
- In the context of RDF, the standard querying language is SPARQL, too computationally complex for our purposes:
  - Evaluation of SPARQL patterns is intractable [Pérez et al. '09].
- Our priority is to identify a query language for graphs expressive enough and also learnable.
Application to cross-model database mappings

- Applications using different data models need to exchange data among them.
- The solutions are based on defining mappings i.e., logical assertions between elements in the source and target schema.
- The mappings are typically defined by an expert user.
- Learning schema mappings has been studied for relational databases [Ten Cate et al. ’12], but no research has been done on learning cross-model database mappings.
Application to cross-model database mappings

- Applications using different data models need to exchange data among them.
- The solutions are based on defining *mappings* i.e., logical assertions between elements in the source and target schema.
- The mappings are typically defined by an expert user.
- Learning schema mappings has been studied for relational databases [Ten Cate et al. ’12], but no research has been done on learning cross-model database mappings.

- A mapping can be seen as a rule having queries as body and head.
- As a further goal, our contributions can be used to address the problem of learning cross-model database mappings.
Cross-model database mappings

**Figure:** Data exchange between heterogeneous data models, using XML as intermediate model.

1. Relational query
2,3. XML query
4. Graph query
Conclusions
The main goal of the thesis is to develop learning algorithms for queries for different database models:

- Relational,
- Semi-structured,
- Graph databases.

We assume large database instances, so the algorithms should be able to learn from a small number of examples.

Depending on the existing results in the literature, we identify two directions of research:

- Extend twig learning,
- Investigate an interactive learning framework for relational and graph queries.
Conclusions

- **Extend twig learning:**
  - √ Propose practical unordered schema formalisms (DMS),
  - √ Investigate the problem of learning unordered schemas from document examples,
    - Use the inferred schema to avoid overspecialization,
    - Other possible extensions: allow negative examples, consider richer query classes, design a practical system, etc.

- **Learning relational and graph queries:**
  - √ Analyze several problems of interest for natural joins and semijoins,
    - Propose interactive learning algorithms,
    - Extend for other relational operators,
    - Investigate a similar framework for graph databases.