Continuous Data Cleaning

M. Volkovs, F. Chiang, J. Szlichta and R. J. Miller

ICDE 2014

UNIVERSITY OF WATERLOO

uwaterloo.ca

Presenter: Nabiha Asghar

Outline

- Introduction and motivation
- Main contributions of the paper
- Description of architecture and techniques
- Experimental evaluation (brief)
- Concluding thoughts

Introduction & Motivation

So far we have seen **data repair algorithms:**

- → assume that a given set of constraints is correct
- → search for least cost repairs satisfying the constraints
- → typically use heuristics, sampling and statistical inference to reduce the space of possible solutions
- → e.g. papers 2.1.2 (Mustafa), 2.2.1 (Qi), 2.2.3 (Prateek), 2.2.4 (Hella), 2.2.7 (Udit), 2.3.2, 2.5.2

UNIVERSITY OF WATERLOO

Introduction & Motivation

So far we have seen **data repair algorithms:**

- → assume that a given set of constraints is correct
- → search for least cost repairs satisfying the constraints
- → typically use heuristics, sampling and statistical inference to reduce the space of possible solutions
- → e.g. papers 2.1.2 (Mustafa), 2.2.1 (Qi), 2.2.3 (Prateek), 2.2.4 (Hella), 2.2.7 (Udit), 2.3.2, 2.5.2

We have NOT looked at **constraint repair algorithms:**

- → aim to identify stale constraints and modify the data/constraints
- → e.g. papers 2.4.1, 2.4.2

UNIVERSITY OF WATERLOO

Introduction & Motivation

So far we have seen **data repair algorithms:**

- → assume that a given set of constraints is correct
- → search for least cost repairs satisfying the constraints
- → typically use heuristics, sampling and statistical inference to reduce the space of possible solutions
- → e.g. papers 2.1.2 (Mustafa), 2.2.1 (Qi), 2.2.3 (Prateek), 2.2.4 (Hella), 2.2.7 (Udit), 2.3.2, 2.5.2

We have NOT looked at **constraint repair algorithms:**

- → aim to identify stale constraints and make modifications
- → e.g. papers 2.4.1, 2.4.2

We have NOT looked at **data cleaning systems:**

- → consider *static* data (snapshot) and *fixed* constraints
- → e.g. papers 2.6.1 (AJAX), 2.6.2 (Potter's Wheel), 2.6.3 (NADEEF), 2.6.4 (LLUNATIC), 2.6.5 (Data Tamer)

UNIVERSITY OF WATERLOO

Introduction & Motivation

- Need a new system for cleaning the data *and* the constraints in *dynamic* environments
- Do incremental cleaning
- Involve the users (domain experts)

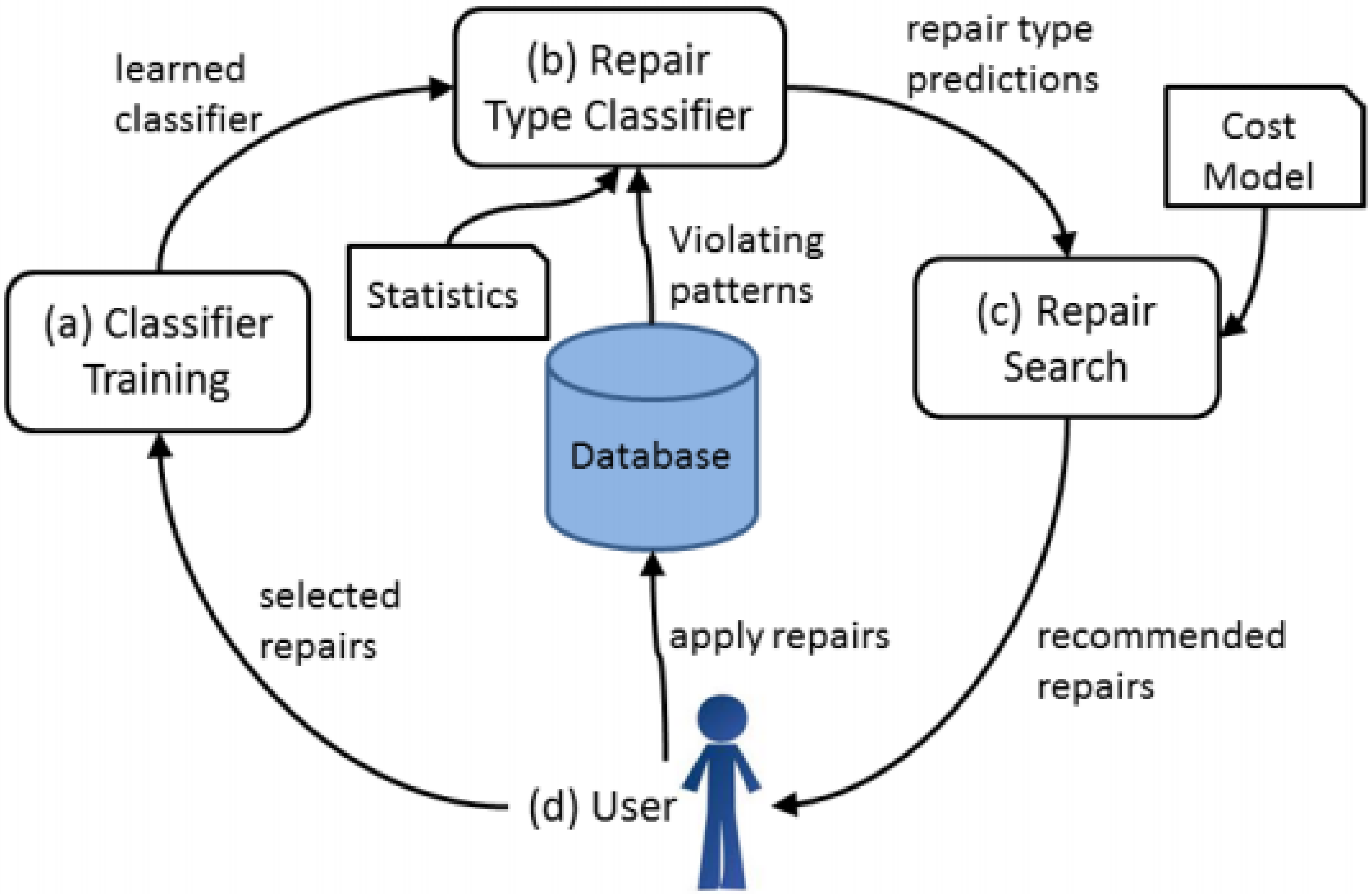
Main Contributions

A cleaning framework that enables continuous data cleaning where both the data and constraints change.

- → A **logistic classifier** to predict the type of repair needed (data, constraint, or both)
- → Input features for the classifier: **22 statistics** over the data and constraints to capture the changing dynamics. Can be updated incrementally
- → Labels: repairs suggested by the **user**

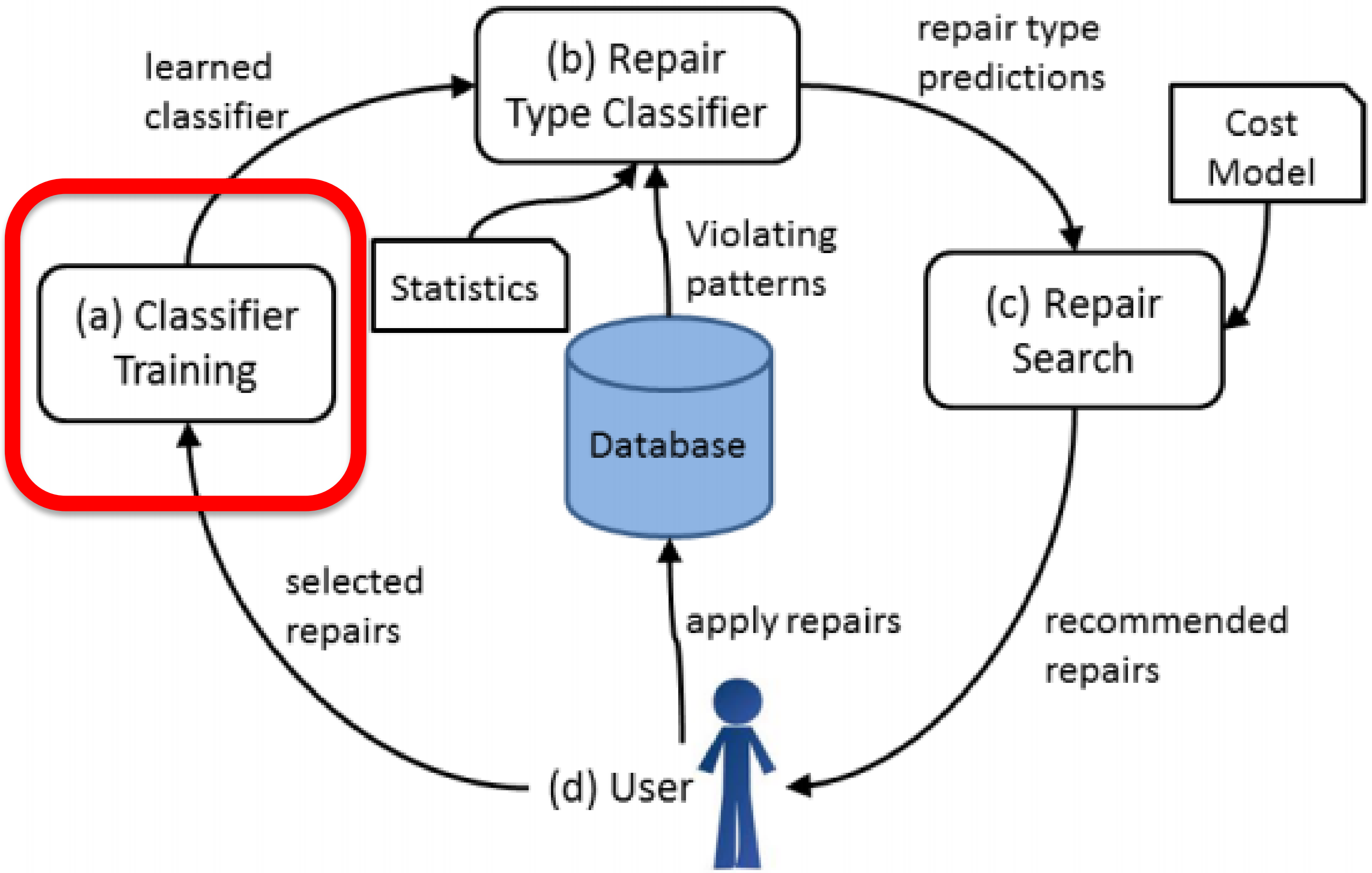
UNIVERSITY OF WATERLOO

Architecture



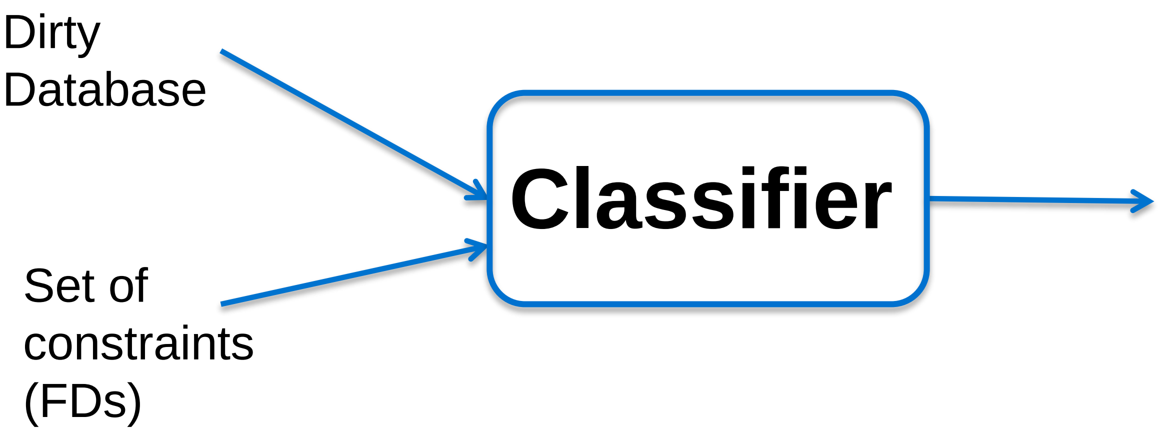
WATERLOO

Architecture



WATERLOO

The Classifier



One of:

1) Not repaired
2) Repaired completely by FD repairs
3) Repaired completely by Data repairs
4) Repaired completely by Data and FD repairs
5) Repaired partially by FD repairs
6) Repaired partially by data repairs
7) Repaired partially by FD and data repairs

(a) Classifier Training

For a given database **I**, a set of FDs **F** and a set of repairs **R**:

1. Create the set **P** of all patterns that violate one or more FDs
2. For each pattern *p* in **P**, compute a 22 x 1 feature vector *G(p)* via 22 statistics
3. Training set = {(*G(p)*, *class*(**R(***p***)**)) for all *p* in **P** }

UNIVERSITY OF WATERLOO

(a) Classifier Training: Computing the Feature Vectors (incrementally)

Compute 22 statistics for pattern p that violates FD F ($X \rightarrow A$):

- Proportion of violating tuples in F
- Proportion of tuples that match p
- Mean({ *overlap*(F', p) } where $F' \neq F$)
- Min({ *fix*(p, $F \rightarrow F^{repaired}$) } for all repairs of F)
- Frequency-based entropy stats of F-satisfying patterns of X:

$$-\sum_{p' \in S_{\mathbf{X}}} \frac{\text{freq}(p')}{|S_{\mathbf{X}}|} \log\left(\frac{\text{freq}(p')}{|S_{\mathbf{X}}|}\right)$$

UNIVERSITY OF WATERLOO

(a) Classifier Training: Computing the Feature Vectors (incrementally)

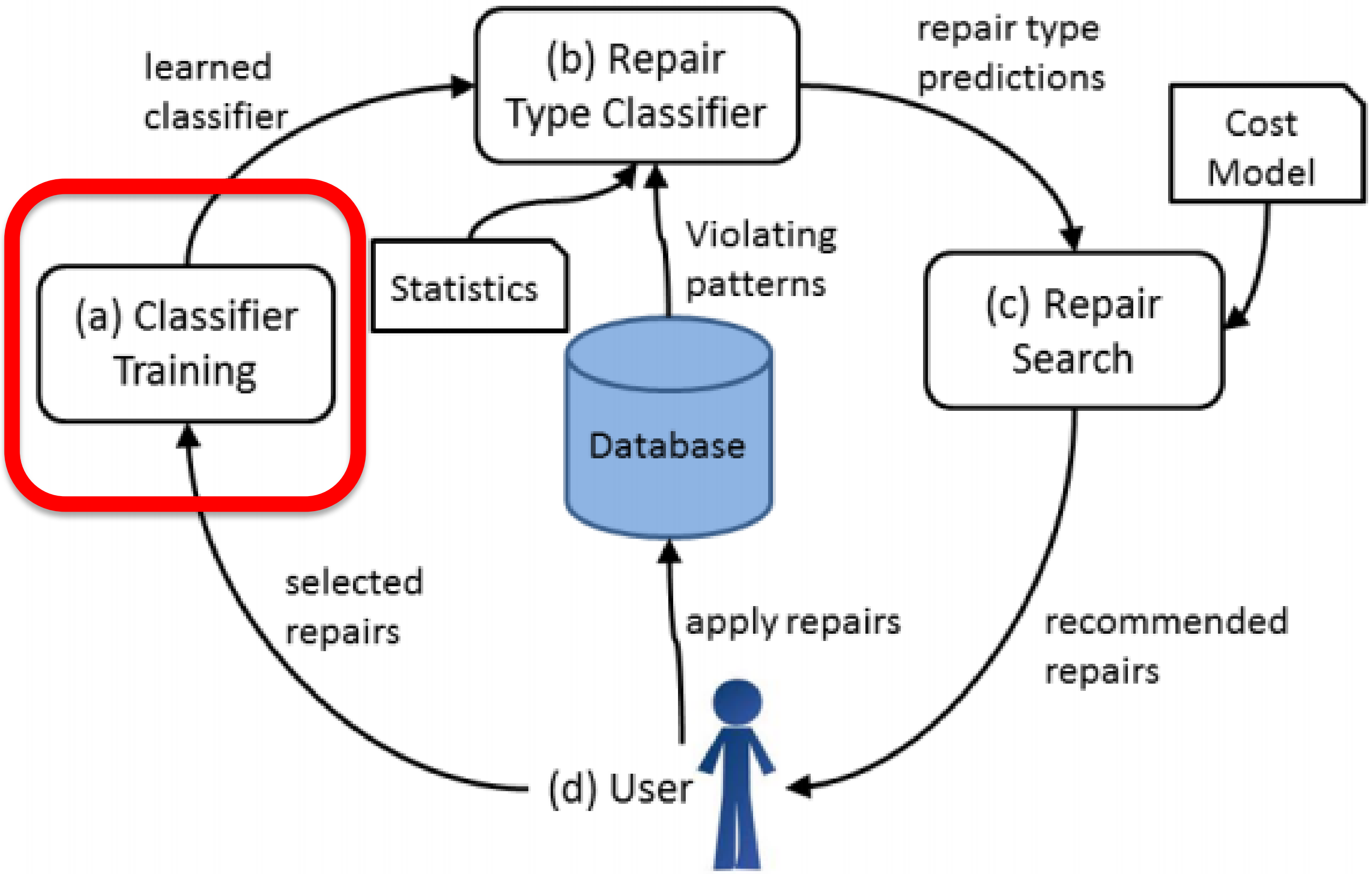
Compute 22 statistics for pattern p that violates FD F ($X \rightarrow A$):

- Proportion of violating tuples in F
- Proportion of tuples that match p
- Mean({ $overlap(F', p)$ } where $F' \neq F$)
- Min({ $fix(p, F \rightarrow F^{repaired})$ } for all repairs of F)
- Frequency-based entropy stats of F-satisfying patterns of X:

$$-\sum_{p' \in S_{\mathbf{X}}} \frac{\text{freq}(p')}{|S_{\mathbf{X}}|} \log\left(\frac{\text{freq}(p')}{|S_{\mathbf{X}}|}\right)$$

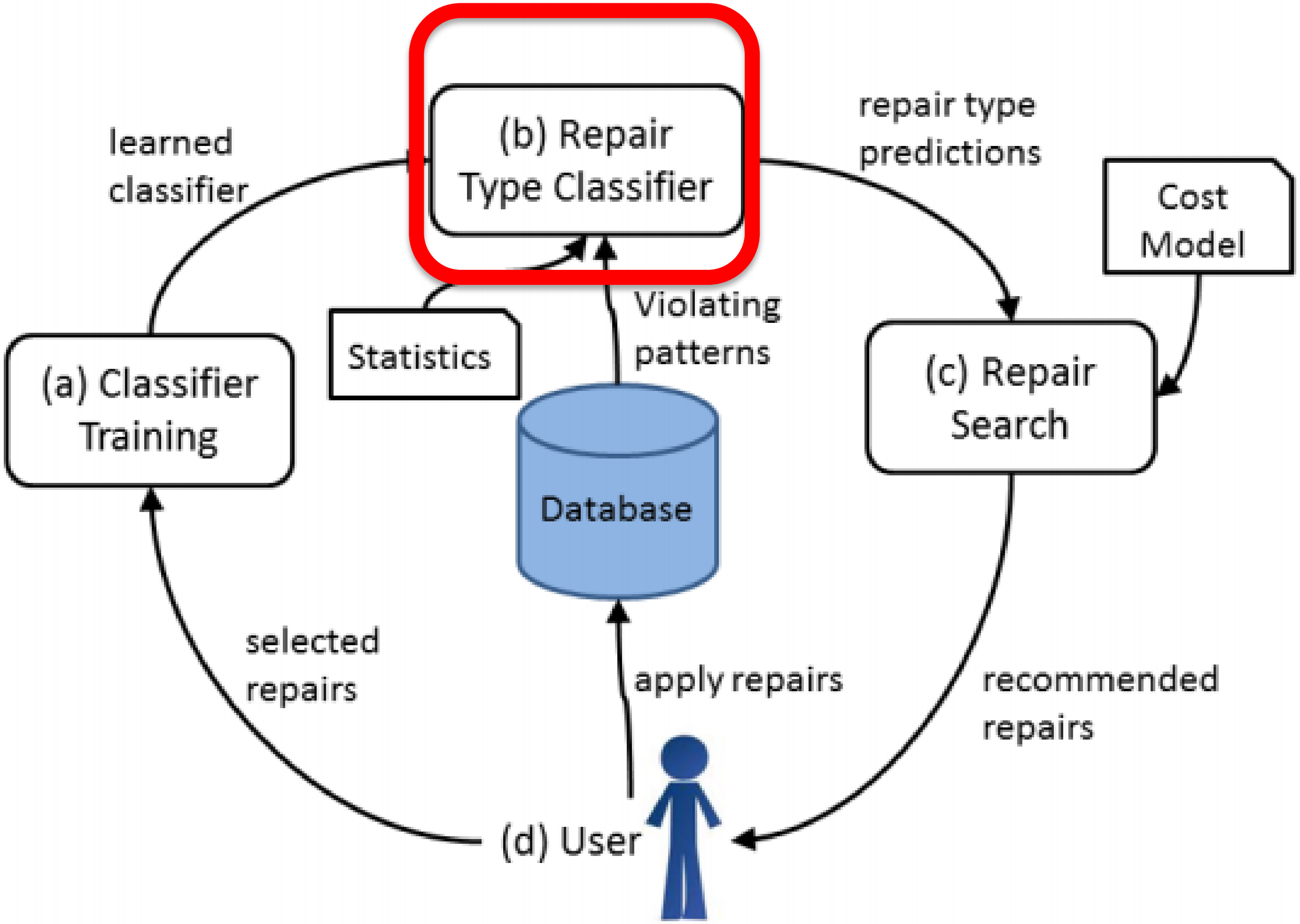
UNIVERSITY OF WATERLOO

Architecture



WATERLOO

Architecture



WATERLOO

(b) Repair-Type Classifier

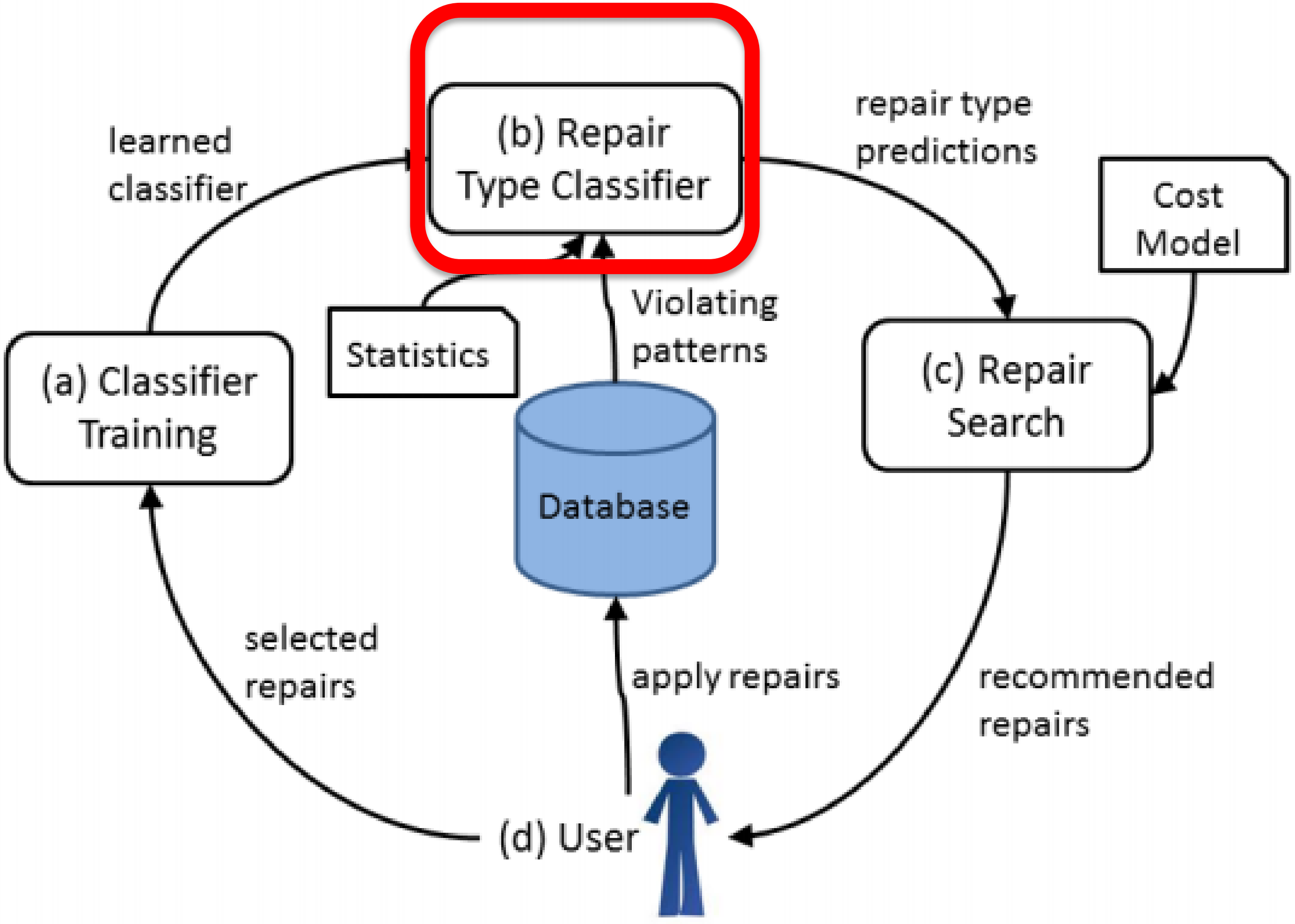
New violating pattern *p'* → **Weighted Logistic Regression** →

For each pattern, one of:

1) Not repaired
2) Repaired completely by FD repairs
3) Repaired completely by Data repairs
4) Repaired completely by Data and FD repairs
5) Repaired partially by FD repairs
6) Repaired partially by data repairs
7) Repaired partially by FD and data repairs

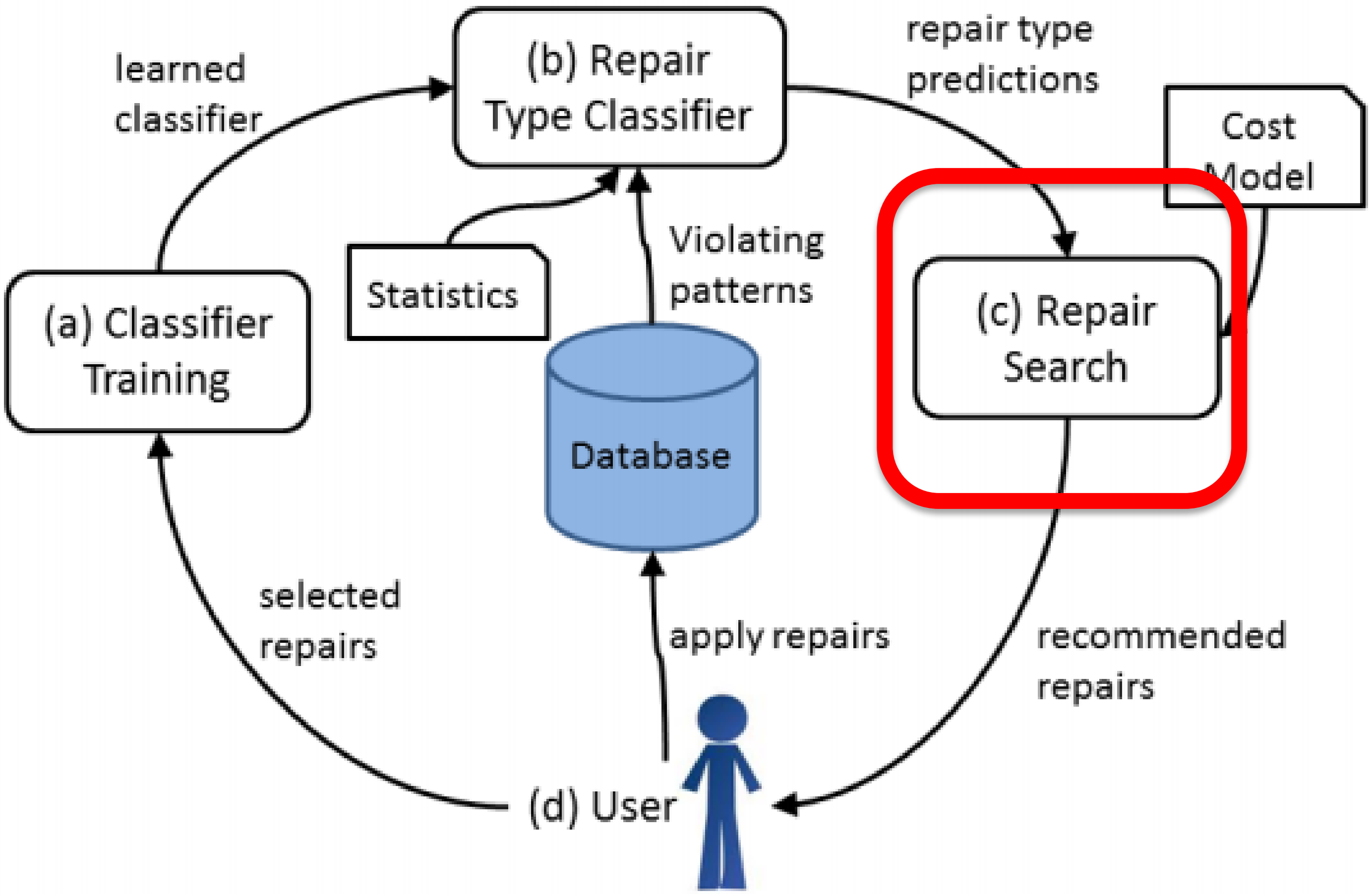
UNIVERSITY OF WATERLOO

Architecture

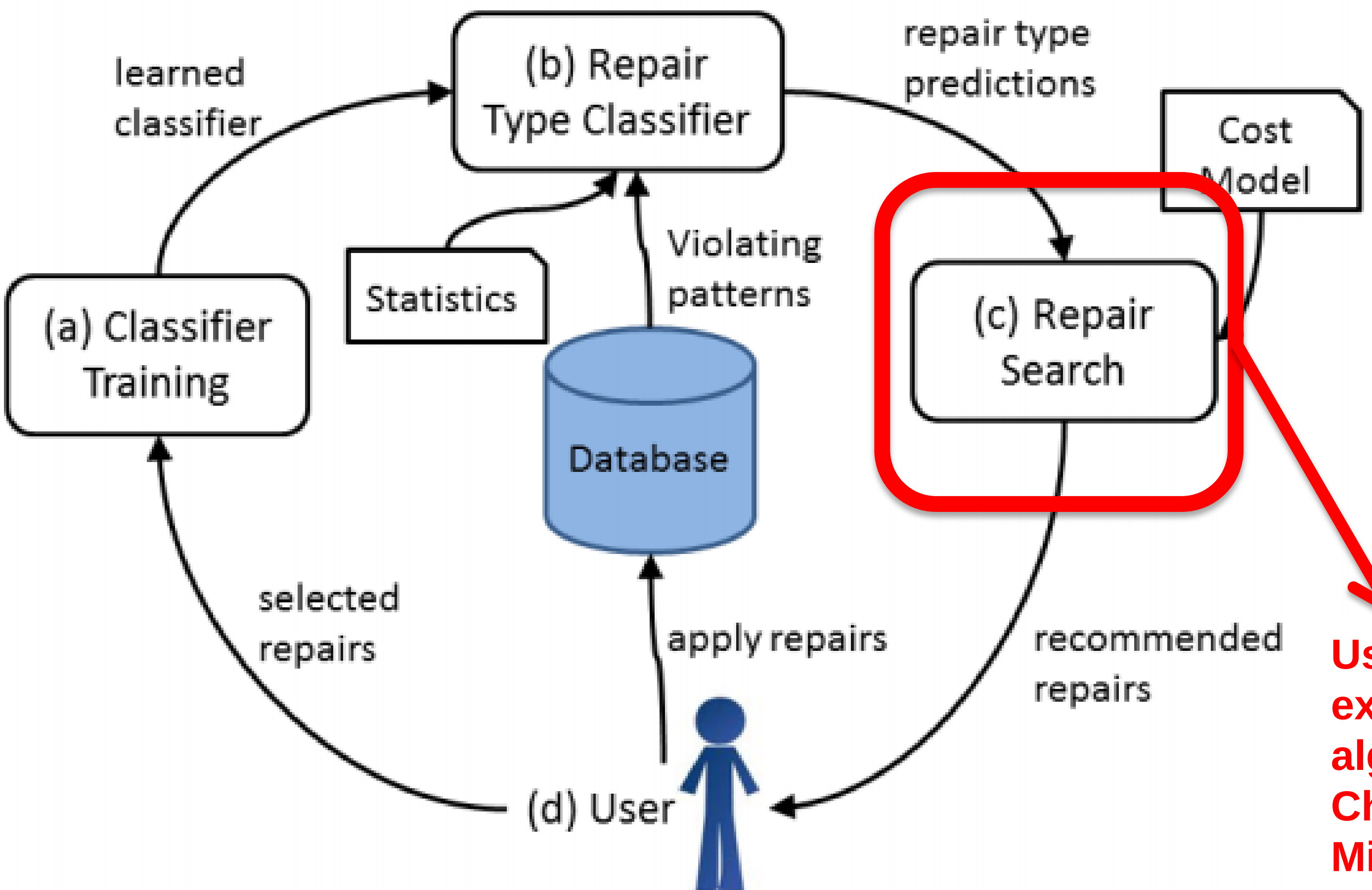


WATERLOO

Architecture



Architecture



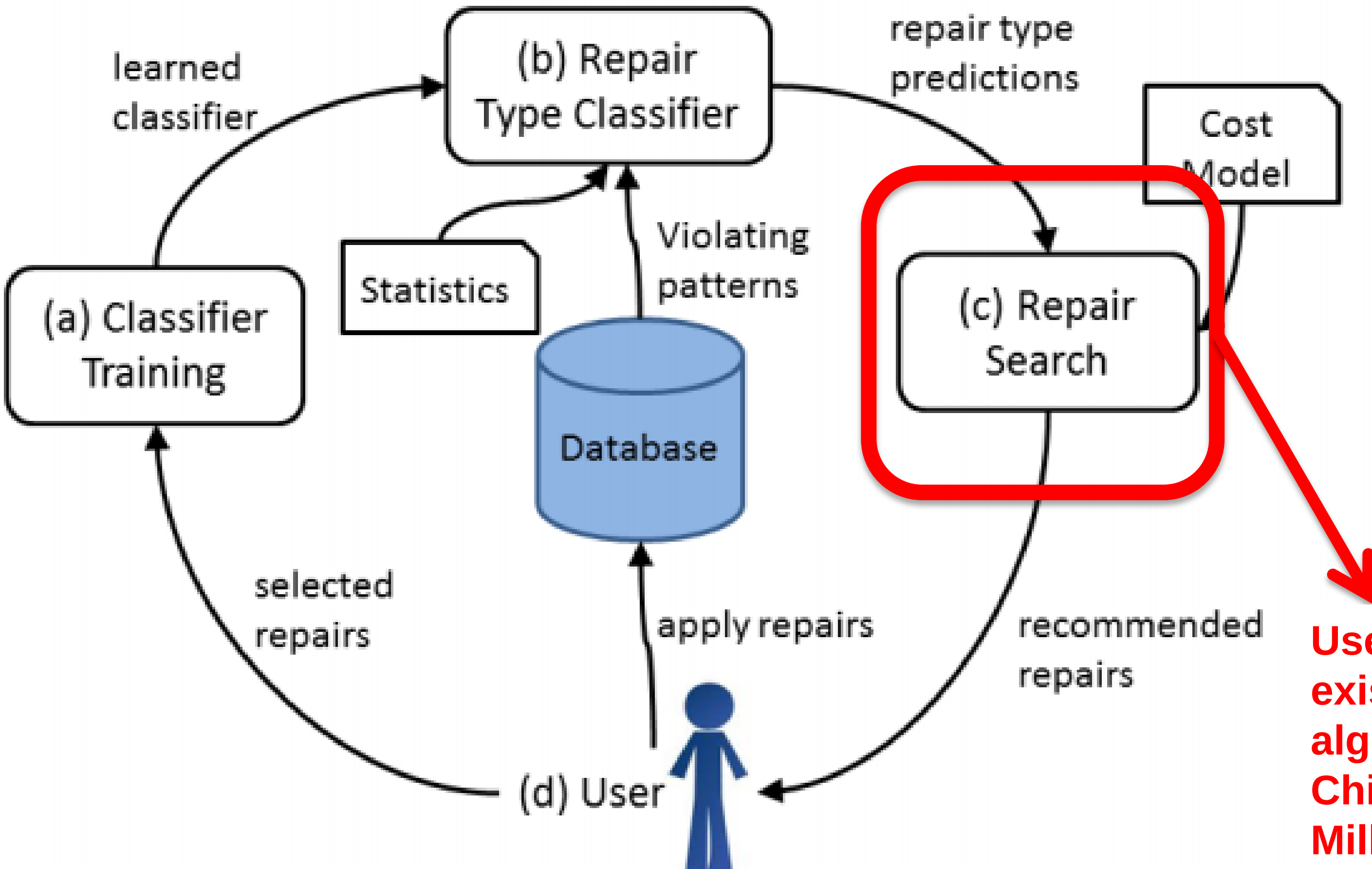
Use an existing algorithm by Chiang and Miller (2011) (paper # 2.4.1)

(c) Repair search (paper # 2.4.1)

- A data repair algorithm that searches for data modifications such that the constraints hold and repair cost is minimal
- A constraint repair algorithm that determines which attributes to add to a constraint to resolve the inconsistency
- A new cost model that quantifies the trade-off of when an inconsistency is a data error (needing a data repair) versus an update to the model (justifying a constraint repair)

UNIVERSITY OF WATERLOO

Architecture



Use an existing algorithm by Chiang and Miller (2011) (paper # 2.4.1)

Evaluation

Four main ways of evaluation:

- Accuracy of classification

    → around 11 to 15 % gain

- Utility of each of the 22 statistics

    → Found 3 statistics to be more useful than others

- Comparison with existing cleaning solutions (precision of repair, and running time)

    → around 20% gain in precision, 13 to 19% in runtime

- Scalability with number of tuples

Summary

- A new data cleaning system that looks for data repairs AND constraint repairs in a continuously changing environment
- Harness the dual power of machine learning and user involvement to prune the search space of repairs
- Achieves better accuracy than existing techniques that only handle static data and fixed constraints
- Is scalable for vertically-expanding data

UNIVERSITY OF WATERLOO

Thoughts

- A nice well-rounded approach to data cleaning: combines machine learning with user expertise to reduce the search space for existing automatic data+constraint repair algorithms

- The numbers for accuracy suggest some room for improvement:

  → try other repair algorithms (paper # 2.4.2, ICDE 2013)

  → try other statistics as features

  → try other classifiers (e.g. decision trees, SVMs)

- Need to test scalability with respect to the number of attributes

- Test the possibility of evolution into a generalized, extensible and easy-to-deploy system like NADEEF

UNIVERSITY OF WATERLOO