Mapping Maintenance for Data Integration Systems

R. McCann, B. AlShebli, Q. Le, H. Nguyen, L. Vu, A. Doan
University of Illinois at Urbana-Champaign
VLDB 2005

Laurent Charlin

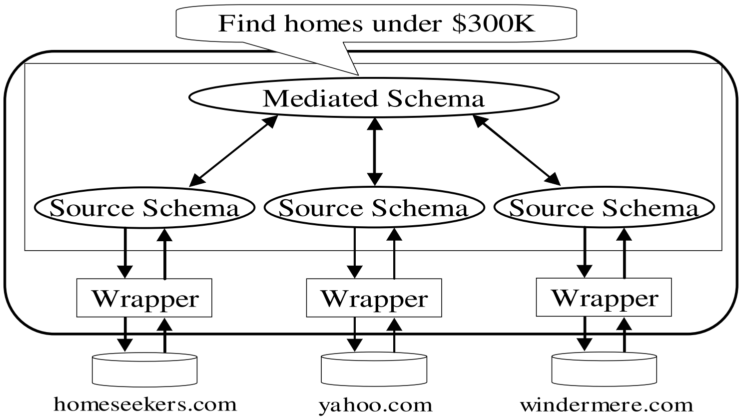
October 26, 2005

Outline

- Problem Definition
- Previous Work in the domain & Background information
- Maveric, automatic mapping verification system
  - Sensor Ensemble
  - Perturber
  - Multi-Source Trainer
  - Filter
- Results from the paper
- Critique & Discussion

Overall

- They Assume a typical schema integration instance



Statement

- They want to solve the Semantic mapping verification problem (ie: answer the question, Is a given mapping broken ?)
  - Assume that the Semantic mapping has been done
- Motivation: They have found that the dominating cost is often the mapping maintenance (detect and repair).

Background

- The authors have a strong AI (machine learning) background
- They are very active in this domain
- From Doan's Ph.D. thesis "First, it introduced machine learning as an indispensable component of matching solutions. Second, it articulated a multi-component, highly extensible architecture for schema matching. Third, it showed how to learn from past matching efforts (to improve accuracy of subsequent matching tasks)."

Some Perspective

- First, Regression tester, relies on formating regularities
- Kushmerick (2000), RAPTURE system, syntaxic only
  - Could be 1 sensor in the current system
- Lerman *et al.* (2003), syntaxic measure as well (results in the experiments section).
  - Learned structural information
  - Positive Data only

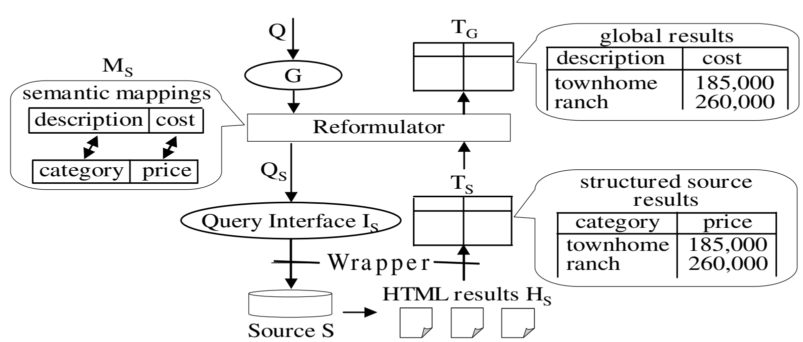
Some Perspective

- Activity monitoring (for example, Fawcett *et al.*, '99)
- It might be formulated as a stream problem
  - Data is continuously arriving from querying the sources
  - You even have some control on the stream since you're controlling the queries.

Typical Machine Learning - Online learning algorithm

- Given a m experts (sensors)
- At each time step (iteration)
  - the sensors predict $score_j \in [0, 1]$
  - the learner, based on all the $score_j$ predicts $score_{cum} \in [0, 1]$
  - compare $score_{cum}$ with the actual label of the example (update the sensor weights based on this).
- In verification (testing): You simply predict with the learned weights
- (*this is taken from Robert Schapire's lectures*)

Back to the current problem



1 - The Sensors

- Computational modules which capture specific characteristics of a source S
- Idea
  1. Train them on data from S
  2. Deploy them to monitor the data returned by queries

1- The Sensors

- Generate examples from querying "valid" semantic mappings
- Two types of parameters to learn
  - The parameters of each sensor (Gaussian mean and variance)
  - The weight of the sensors (in the Winnown algorithm)

1 - Winnow Algorithm

Train the Sensor Combiner

Input: examples $R_1,...,R_n$ labeled with $+$ or , alarm threshold
sensors $s_1,...,s_m$ (already trained on $R_1,...,R_n$)
Output: sensor weights $w_1,\ \ ,w_n$
1. Initialize each weight w_i to 1
2. Repeat: for each example R_i
   for each sensor s_j, $score_j$ the score of s_j when applied to R_i
   $score_{comb}$ = the combined score of all sensors using $w_1,\ \ ,w_n$
   if ($score_{comb} \geq \theta$ and $R_i.label$ = -) // false alarm
   $\boxed{w_j = w_j / 2 \text{ for each } score_j \geq \theta}$
   else if ($score_{comb} < \theta$ and $R_i.label = +$) // missed alarm
   $\boxed{w_j = w_j / 2 \text{ for each } score_j < \theta}$
   until a stopping criterion is reached
3. Return $w_1,\ \ ,w_n$

- Final classifier is given by

$$vote_{valid} = \sum_{i}^{m} w_i * score_i$$

$$vote_{invalid} = \sum_{i}^{m} w_i * (1 - score_i)$$

1 - The Sensor Types

- Value Sensors
  - Monitor features of attributes
  - Data modeled according to a Gaussian distribution
  - Density Scoring $score_s = 1 - P(v)$
  - Normalized Density scoring $score_s = Pr[P(v') \geq P(v)]$
- Trend Sensors
  - Work much like the Value sensors
- Layout Sensors
  - Monitors the HTML layouts
- Constraint Sensors
  - Monitors pre-defined attribute constraints

2 - Perturbation

- Problem: No Negative Example
- Solution: Generate negative examples from current source (S) data
- Corrolary: They are also trying to generate more diversified examples
- They Simulate the following situations
  - Change in the Source Query Interface
  - Change Source Data
  - Change the Presentation Format
- In training they incorporate this new data (both positive and negative) into the examples R.

2 - Perturbation

- The addition of positive (invalid data) changed the way the score is calculted
- It's due to the fact that they use Gaussian modeling of the data (they cannot incorporate both positive and negative examples in their distributions).
- They have two Gaussians
- $score_{cum} = score_{-}/(score_{-} + score_{+})$

3 - Multi-Source training

- Usually you have to train on a single source S
- What about using other sources S' from the same domain which have equivalent attributes
- Example (two attributes which are tied by the semantic schema):
  - S: price $\$185,000$
  - S': amount $185,000 USD$

4 - Filtering

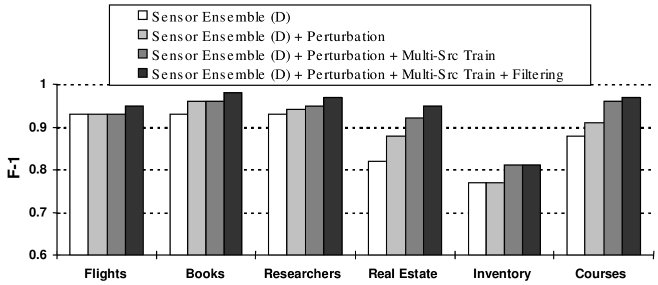
- Motivation : Have to find a balance between false positve and false negative (it cannot be attained by changing the value of the threshold θ).
- 3 filters
  - Each as the ability to silence attribute
  - If some are not silenced after passing the filters then you raise an alarm

4 - Filtering

1. Syntaxic Recognizer (much in the style of previous papers on the subject)
2. Exploiting External Sources
   - Trains a new sensor on newly acquired data from a different source
   - Lets you learn from other previously broken mappings
3. Learning from the web (*Google is your friend !*)
   - Is "185,000 USD" a cost ?
   - Search for:
     3.1 "185,000 USD"
     3.2 "cost 185,000 USD"
   - If the ratio is high enough, then it's valid (ie: silence the attribute)

- This is the most semantic it gets

Results



Results

Domain	Lerman System		Sensor Ensemble (D)		Sensor Ensemble (ND)	
	P / R	F-1	P / R	F-1	P / R	F-1
Flights	0.81 / 1.00	0.85	0.93 / 0.98	0.93	0.93 / 0.98	0.93
Books	0.83 / 1.00	0.89	0.90 / 0.99	0.93	0.90 / 0.99	0.93
Researchers	0.77 / 0.99	0.84	0.90 / 0.99	0.93	0.90 / 0.99	0.93
Real Estate	0.45 / 0.90	0.63	0.80 / 0.82	0.82	0.82 / 0.82	0.80
Inventory	0.52 / 0.89	0.67	0.75 / 0.90	0.77	0.71 / 0.90	0.75
Courses	0.49 / 0.94	0.66	0.92 / 0.88	0.88	0.88 / 0.87	0.85

Discussion

- They improve previous work by
  - Broader collection of evidences
  - The ensemble of sensors allow to use these evidences
  - Weighted combining of sensors
- They still have to improve
  - Unrecognized formats (not seen in training and not on the web)
  - Mixed same type attributes when they were switched

Final remarks

- Why not use a more powerful meta-learning algorithm (ie: AdaBoost (Schapire '90s)) ?
- How far can we push the stream analogy ?
- Why does (D) perform as well as (ND) ?

Final remarks

- Previous work do mostly syntaxic analysis
- This paper does it a little better
  - They have the same fundamental problems as other papers
  - Very adhoc sensors
  - Needs to train on every source independently
  - Filtering is a good idea but it needs to be pushed further
- Next step, try to understand semantically the query returns
- AI techniques would become even more important