Iteratively Learning Conditional Statements in Transforming Data by Example

Bo Wu and Craig A. Knoblock

University of Southern California
Introduction
Motivation

Data Source A  Data Source B  Data Source C

Data Transformation

Data in the ready format
# A Data Table

<table>
<thead>
<tr>
<th>Accession</th>
<th>Credit</th>
<th>Dimensions</th>
<th>Medium</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.2</td>
<td>Gift of the artist</td>
<td>5.25 in HIGH x 9.375 in WIDE</td>
<td>Oil on canvas</td>
<td>John Mix Stanley</td>
</tr>
<tr>
<td>05.411</td>
<td>Gift of James L. Edison</td>
<td>20 in HIGH x 24 in WIDE</td>
<td>Oil on canvas</td>
<td>Mortimer L. Smith</td>
</tr>
<tr>
<td>06.1</td>
<td>Gift of the artist</td>
<td>Image: 20.5 in. HIGH x 17.5 in. WIDE</td>
<td>Oil on canvas</td>
<td>Theodore Scott Dabo</td>
</tr>
<tr>
<td>06.2</td>
<td>Gift of the artist</td>
<td>9.75 in</td>
<td>16 in HIGH x 13.75 in</td>
<td>19.5 in WIDE</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09.8</td>
<td>Gift of the artist</td>
<td>12 in</td>
<td>14 in HIGH x 16 in</td>
<td>18 in WIDE</td>
</tr>
</tbody>
</table>
## Programming by Example

<table>
<thead>
<tr>
<th></th>
<th>Raw Value</th>
<th>Target Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>5.25 in HIGH x 9.375 in WIDE</td>
<td>9.375</td>
</tr>
<tr>
<td>R2</td>
<td>20 in HIGH x 24 in WIDE</td>
<td>24</td>
</tr>
<tr>
<td>R3</td>
<td>Image: 20.5 in. HIGH x 17.5 in. WIDE</td>
<td>17.5</td>
</tr>
<tr>
<td>R4</td>
<td>9.75 in</td>
<td>16 in HIGH x 13.75 in</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td>12 in</td>
<td>14 in HIGH x 16 in</td>
</tr>
</tbody>
</table>
Problem:
Learn accurate conditional statements efficiently for data with heterogeneous formats using few examples
Previous Approach

Compatibility score \(O(n^3)\)

Few Training Data

Examples

Data Preprocessing

Cluster

Learn classifier

Partitions

Derive branch program

Conditional

Branch Transformation program

Combine

Transformation Program

Examine Results and provide examples

Get examples and provide results

GUI
Transformation Program

```javascript
function Transform(value) {
  label = classify(value);
  switch label:
    case "format1":
      pos1 = value.indexOf('BNK', 'NUM', -1);
      pos2 = value.indexOf('NUM', 'BNK', 2);
      output = value.substr(pos1, pos2);
      break;
    case "format2":
      pos3 = value.indexOf('l','NUM',2);
      pos4 = value.indexOf('NUM','BNK',-1);
      output = value.substr(pos3, pos4);
      break;
  return output;
}
```

Example: 9.75 in | 16 in HIGH x 13.75 in | 19.5 in WIDE ➔ 19.5
Our Approach
Main Idea

Learning the conditional statement iteratively

Input: 5.25 in HIGH x 9.375 in WIDE
Output: 9.375

Input: 9.75 in | 16 in HIGH x 13.75 in | 19.5 in WIDE
Output: 13.35

Time

Running Information
Our Approach

Convert data into feature vectors

Utilize previous constraints

Utilize Unlabeled Data

Examples

→

Data Preprocessing

Cluster

→

Partitions

Learn classifier

Branch Transformation program

Derive branch program

Conditional

→

Combine

Transformation Program
# Data Preprocessing

<table>
<thead>
<tr>
<th>String</th>
<th>9.75 in</th>
<th>16 in HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokens</td>
<td>START NUM(9) Period(.) NUM(75) BNK LWRD(in) VBAR(</td>
<td>) NUM (16) BNK LWRD(in) BNK UWRD(H) UWRD(I) UWRD(G) UWRD(H) ...</td>
</tr>
<tr>
<td>Token counts</td>
<td>NUM</td>
<td>UWRD</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Feature Vector</td>
<td>LWRD</td>
<td>NUM</td>
</tr>
<tr>
<td></td>
<td>0.21</td>
<td>0.29</td>
</tr>
</tbody>
</table>
Our Approach

- Convert data into feature vectors
- Utilize previous constraints
- Utilize Unlabeled Data

Diagram:

1. Examples
2. Data Preprocessing
3. Cluster
4. Partitions
5. Learn classifier
6. Derive branch program
7. Conditional
8. Branch Transformation program
9. Combine
10. Transformation Program

Steps:
- Convert data into feature vectors
- Utilize previous constraints
- Utilize Unlabeled Data
Constraints

- Two Types of Constraints:
  - Cannot-merge Constraints:
    - Ex:
      | Constraint                          | Value |
      |-------------------------------------|-------|
      | 5.25 in HIGH x 9.375 in WIDE        | 9.375 |
      | 9.75 in |16 in HIGH x 13.75 in| 19.5 in WIDE | 13.75 |
      | 20 in HIGH x 24 in WIDE             | 24    |
  - Must-merge Constraints:
    - Ex:
      | Constraint                          | Value |
      |-------------------------------------|-------|
      | 5.25 in HIGH x 9.375 in WIDE        | 9.375 |
      | 20 in HIGH x 24 in WIDE             | 24    |
      | 9.75 in |16 in HIGH x 13.75 in| 19.5 in WIDE | 13.75 |
      | Image: 20.5 in. HIGH x 17.5 in. WIDE | 17.5  |
Constrained Agglomerative Clustering

Update constraints
Learn distance metric
Distance Metric Learning

- **Distance Metric (Weighted Euclidean) Learning**

\[ d(x, y) = \|x - y\|_w = \sqrt{\sum_i w_i (x_i - y_i)^2} \]

- **Objective Function**

\[
\arg\min_{w > 0} \sum_i \|x_i - e_{x_i}\|_w + a * g(w) - b * h(w)
\]

\[
g(w) = \ln(\sum_{X_m} \sum_{x_i, x_j \in X_m, i \neq j} \|x_i - x_j\|_w)
\]

\[
h(w) = \ln(\sum_{X_r} \max_{x_i, x_j \in X_r} \|x_i - x_j\|_w)
\]

Close to each other

Too far away
Our Approach

- Convert data into feature vectors
- Utilize previous constraints
- Utilize Unlabeled Data

Diagram:

1. **Examples**
   - Data Preprocessing
     - Cluster
     - Partitions
     - Learn classifier
     - Derive branch program
     - Conditional
     - Branch Transformation program
   - Combine
     - Transformation Program
Utilize Unlabeled data in Learning Classifier

Filter unlabeled data
1. Filter unlabeled data on the boundary
2. Only choose top K unlabeled data

Learn a SVM classifier

<table>
<thead>
<tr>
<th>Partition 1</th>
<th>Examples</th>
<th>5.25 in HIGH x 9.375 in WIDE</th>
<th>9.375</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 in HIGH x 24 in WIDE</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image: 20.5 in. HIGH x 17.5 in. WIDE</td>
<td>17.5</td>
<td></td>
</tr>
<tr>
<td>Unlabeled</td>
<td>26 in. HIGH x 23 in. WIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19.75 in HIGH x 22.75 in WIDE x 0.25 in DEEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>33.5 in HIGH x 39 in WIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partition 2</th>
<th>Examples</th>
<th>9.75 in</th>
<th>16 in HIGH x 13.75 in</th>
<th>19.5 in WIDE</th>
<th>13.75</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12 in</td>
<td>14 in HIGH x 16 in</td>
<td>18 in WIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlabeled</td>
<td>20.25 in</td>
<td>19.75 in HIGH x 15.75 in</td>
<td>15.875 in WIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55 in HIGH x 46 in</td>
<td>290 in WIDE</td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Results
Evaluation

• Dataset: 30 editing scenarios
  – Museum
  – Google Refine and Excel user forums

• Comparing Methods:
  – **SP**
    • The state-of-the-art approach that uses compatibility score to select partitions to merge
  – **SPIC**
    • Utilize previous constraints besides using compatibility score
  – **DP**
    • Learn distance metric
  – **DPIC**
    • Utilize previous constraints besides learning distance metric
  – **DPICED**
    • Our approach in this paper
Results

Success Rates:

<table>
<thead>
<tr>
<th></th>
<th>DPICED</th>
<th>DPIC</th>
<th>DP</th>
<th>SPIC</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SccRate</td>
<td>1</td>
<td>1</td>
<td>0.97</td>
<td>0.77</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Time and Examples:

<table>
<thead>
<tr>
<th></th>
<th>Total Time (seconds)</th>
<th>Examples</th>
<th>Constraint Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPICED</td>
<td>3.9</td>
<td>5.4</td>
<td>6.1</td>
</tr>
<tr>
<td>DPIC</td>
<td>6.4</td>
<td>6.8</td>
<td>6.6</td>
</tr>
<tr>
<td>DP</td>
<td>8.3</td>
<td>6.8</td>
<td>17.6</td>
</tr>
<tr>
<td>SPIC</td>
<td>21.3</td>
<td>6.8</td>
<td>260.1</td>
</tr>
<tr>
<td>SP</td>
<td>26.5</td>
<td>6.9</td>
<td>305.8</td>
</tr>
</tbody>
</table>
Related Work

• Wrapper induction approaches
  – WIEN [Kushmerick, 1997], SoftMealy [Hsu et al., 1998], STALKER [Muslea et al., 1999]

• Programming-by-example approaches
  – FlashFill[Gulwani, 2011][Perelman et al., 2014], Data Wrangler [Kandel et al., 2011], SmartEditor [Lau et al. 2003]

• Clustering with constraints
  – Clustering with constraints [Xing et al., 2002][Bilenko et al., 2004][Bade et al., 2006][Zhao et al., 2010] [Zheng et al., 2011]
Discussion

• Iteratively learn conditional statements in PBE setting
  – Improve the efficiency
  – Learn more accurate conditional statements
  – generate a small number of branches.

• Incorporate ML tools as external functions in inductive programming
Future Work

• Integrate the partitioning and classification steps
  – Reduce accumulated errors
• Improve GUI to help user verifying the data
  – Identify unseen formats
  – Identify incorrectly classified records
• Thanks