Domain Adaptation for Deep Entity Resolution

Jianhong Tu
Renmin University, China

Ju Fan
Renmin University, China

Nan Tang
QCRI, Qatar

Peng Wang
Renmin University, China

Chengliang Chai
Tsinghua University, China

Guoliang Li
Tsinghua University, China

Ruixue Fan
Renmin University, China

Xiaoyong Du
Renmin University, China
Entity Resolution (ER)

- As a core problem of data integration, entity resolution (ER) is to determine whether two data instances refer to the same real-world entity.
  - E.g., matching products from two e-commerce websites

<table>
<thead>
<tr>
<th>Table A</th>
<th>Table B</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>a₁</td>
<td>samsung 52'' series 7 black flat ...</td>
</tr>
<tr>
<td>a₂</td>
<td>sony 46'' bravia ...</td>
</tr>
<tr>
<td>a₃</td>
<td>linksys wirelessn ...</td>
</tr>
</tbody>
</table>

- Existing methods:
  - Rule-based methods: disjunctive normal form, general boolean formula, etc.
  - Machine Learning (ML)-based methods: SVM, random forests, etc.
  - Deep Learning (DL)-based methods: DeepMatcher, DeepER, Ditto, etc.

DL-based methods achieve the state-of-the-art (SOTA) results of ER.
DL-based Entity Resolution (Deep ER)

The Framework of Deep ER:

- **Feature Extractor** converts entity pair \((a, b)\) into \(d\)-dimensional vector-based representation (feature).
- **Matcher** takes the feature of entity pair as input, and predicts whether they match or not.

\[
\hat{y} = M(x) = M(\mathcal{F}(a, b))
\]

<table>
<thead>
<tr>
<th>Table A: entity a</th>
<th>Table B: entity b</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
</tr>
<tr>
<td>(a_1)</td>
<td>bait wheasel ...</td>
</tr>
<tr>
<td>(a_2)</td>
<td>kodak esp ...</td>
</tr>
<tr>
<td>(a_3)</td>
<td>hp q3675a ...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>title</td>
</tr>
<tr>
<td>(b_1)</td>
<td>bait inc. ...</td>
</tr>
<tr>
<td>(b_2)</td>
<td>kodak esp 7 ...</td>
</tr>
<tr>
<td>(b_3)</td>
<td>hewlett ...</td>
</tr>
</tbody>
</table>

Problem: DL-based methods need a large amount of labeled training data.
Opportunity of Reusing Well-Labeled ER Datasets

- There are many well-labeled ER datasets, either public on the Web or available in enterprises
  - E.g., Magellan datasets and WDC datasets

Magellan datasets

WDC datasets

Can we reuse these labeled ER datasets for a new unlabeled ER dataset?
Directly Reusing Feature Extractor and Matcher Trained on Labeled Source?

Step 1: Getting the Feature Extractor $\mathcal{F}$ and the Matcher $M$ trained by labeled source data.

Step 2: Mapping the unlabeled target data into the feature space.

Step 3: Predicting the target data with $M$ directly. The $M$ fails to predict the target data due to the distribution change or domain shift of feature space.
Distribution Change or Domain Shift

- **Similar domains**
  - Source ( Citation): DBLP-ACM (Title, Authors, Venue, Year)
  - Target ( Citation): DBLP-Scholar (Title, Authors, Venue, Year)

- **Different domains**
  - Source (Music): iTunes-Amazon (Album Name, Artist Name, Song Name, Album Price, ...)
  - Target ( Citation): DBLP-Scholar (Title, Authors, Venue, Year)

Training only with source vs. Training with target (F1)

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Training only with source</th>
<th>Training with target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar domains</td>
<td>DBLP-ACM</td>
<td>77.8</td>
<td>95.6</td>
</tr>
<tr>
<td>Different domains</td>
<td>iTunes-Amazon</td>
<td>68.2</td>
<td>95.6</td>
</tr>
</tbody>
</table>

Can we better reuse the source?
Domain Adaptation (DA) Comes to Rescue

- One example of **DA for CV**.

[1] Reducing distribution differences

- Train domain classifier
- Update features
- Deceive Domain Classifier

[2] Improving performance

- Source
- Gain of DA
- Directly reusing source


Domain Adaptation (DA) for Deep ER

- Learn domain-invariant and discriminative features.

Domain-invariant: reducing distribution change.

Discriminative: extracting discriminative information.

Whether DA can be used for ER tasks?
DADER Framework

- Feature Extractor and Matcher
- Feature Aligner: the key module to realize domain adaptation.

1. Training $\mathcal{F}$ and $\mathcal{M}$ with labeled source data.
2. Utilizing unlabeled target data.
3. Adjusting $\mathcal{F}$ and $\mathcal{M}$ to align distributions of source and target, such that $\mathcal{F}$ and $\mathcal{M}$ can work on target.
## DADER Design Space

- Feature Extractor: RNN, LMs
- Matcher: MLP
- Feature Aligner: Discrepancy-based, Adversarial-based, Reconstruction-based

<table>
<thead>
<tr>
<th>Modules</th>
<th>Categorization</th>
</tr>
</thead>
</table>
| **Feature Extractor** $(F)$ | (I) Recurrent neural network (RNN)  
                           | (II) Pre-trained language models (LMs)         |
| **Matcher** $(M)$       | Multi-layer Perceptron (MLP)                   |
| **Feature Aligner** $(A)$ | (1) Discrepancy-based  
                           | (a) MMD  
                           | (b) K-order  
                           | (c) GRL  
                           | (d) InvGAN  
                           | (e) InvGAN+KD  
                           | (3) Reconstruction-based  
                           | (f) ED |
Representative Method: MMD (Discrepancy-based)

- Feature Aligner is a function to measure maximum mean discrepancy.

During training, the Maximum Mean Discrepancy of source and target feature spaces is computed and reduced. The smaller the MMD, the more similar the distributions.

\[
\mathcal{L}_{MMD} = \sup_{\| \phi \|_H \leq 1} \| E_{x^s \sim p_s} [\phi(x^s)] - E_{x^t \sim p_t} [\phi(x^t)] \|_H^2
\]
Representative Method: InvGAN (Adversarial-based)

- Feature Aligner is a **binary domain classifier** to discriminate source/target dataset.

During training, the optimization objective of Feature Aligner is to **minimize the domain classification loss**, while Feature Extractor is to generate the **indistinguishable features** that confuse Feature Aligner.

\[
\min_{\mathcal{F}, \mathcal{M}} \max_{\mathcal{A}} V(\mathcal{F}, \mathcal{M}, \mathcal{A}) = L_M(\mathcal{F}, \mathcal{M}) + \beta L_A(\mathcal{F}, \mathcal{A}),
\]

\[
L_A = E_{x^s \sim \mathcal{D}_s} \log \mathcal{A}(\mathcal{F}(x^s)) + E_{x^t \sim \mathcal{D}_t} \log (1 - \mathcal{A}(\mathcal{F}(x^t))),
\]
Representative Method: ED (Reconstruction-based)

- Feature Aligner is a decoder to reconstruct the initial data for source and target.

During training, the auxiliary reconstruction task can ensure the shared Feature Extractor (encoder) to extract important and shared information from both domains.

One example of Encoder-Decoder (ED) Architecture: Bart

\[
\mathcal{L}_{REC} = E_{x \sim D_S \cup D_T} [\mathcal{L}_{CE}(\mathcal{A}(\mathcal{F}(x)), x)]
\]
Datasets (DeepMatcher, Magellan, and WDC)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Domain</th>
<th>#Pairs</th>
<th>#Matches</th>
<th>#Attrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walmart-Amazon (WA)</td>
<td>Product</td>
<td>10,242</td>
<td>962</td>
<td>5</td>
</tr>
<tr>
<td>Abt-Buy (AB)</td>
<td>Product</td>
<td>9,575</td>
<td>1,028</td>
<td>3</td>
</tr>
<tr>
<td>DBLP-Scholar (DS)</td>
<td>Citation</td>
<td>28,707</td>
<td>5,347</td>
<td>4</td>
</tr>
<tr>
<td>DBLP-ACM (DA)</td>
<td>Citation</td>
<td>12,363</td>
<td>2,220</td>
<td>4</td>
</tr>
<tr>
<td>Fodors-Zagats (FZ)</td>
<td>Restaurant</td>
<td>946</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>Zomato-Yelp (ZY)</td>
<td>Restaurant</td>
<td>894</td>
<td>214</td>
<td>3</td>
</tr>
<tr>
<td>iTunes-Amazon (IA)</td>
<td>Music</td>
<td>532</td>
<td>132</td>
<td>8</td>
</tr>
<tr>
<td>RottenTomatoes-IMDB (RI)</td>
<td>Movies</td>
<td>600</td>
<td>190</td>
<td>3</td>
</tr>
<tr>
<td>Books2 (B2)</td>
<td>Books</td>
<td>394</td>
<td>92</td>
<td>9</td>
</tr>
<tr>
<td>WDC-Computers (CO)</td>
<td>Product</td>
<td>1,100</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>WDC-Cameras (CA)</td>
<td>Product</td>
<td>1,100</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>WDC-Watches(WT)</td>
<td>Product</td>
<td>1,100</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>WDC-Shoes (SH)</td>
<td>Product</td>
<td>1,100</td>
<td>300</td>
<td>2</td>
</tr>
</tbody>
</table>

Similar domains: Partially different attributes

Similar domains: Partially different textual styles

Different domains: Totally different attributes

Similar domains: different categories within the same website.
Baselines

- NoDA
  - Using the model trained on source directly.

- DeepMatcher [1]
  - State-of-the-art deep ER method.

- Ditto [2]
  - State-of-the-art ER method with pre-trained model.

Overall Results of DA for ER

- DA works well on the datasets from:
  - Similar domains, e.g., WA (product) → AB (product).
  - Different domains, e.g., RI (movie) → AB (product).

![F1 Score Graphs]

1. **Similar domains**: WA → AB: 65.8, AB → WA: 56.9, DS → DA: 97.2, DA → DS: 77.8, ZY → FZ: 85.4, FZ → ZY: 47.6


- **NoDA** and **Gain of DA** are indicated in different colors.
Results of Different Methods

- Discrepancy-based and Adversarial-based methods perform well.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Source</th>
<th>Target</th>
<th>NoDA</th>
<th>Discrepancy-based</th>
<th>Adversarial-based</th>
<th>Reconstruction-based</th>
<th>ΔF1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MMD</td>
<td>K-order</td>
<td>GRL</td>
<td>InvGAN</td>
</tr>
<tr>
<td>Similar Domain</td>
<td>Walmart-Amazon</td>
<td>Abt-Buy</td>
<td>65.8</td>
<td>72.6</td>
<td>68.3</td>
<td>68.4</td>
<td>56.0</td>
</tr>
<tr>
<td></td>
<td>Abt-Buy</td>
<td>Walmart-Amazon</td>
<td>56.9</td>
<td>71.1</td>
<td>62.0</td>
<td>66.3</td>
<td>47.5</td>
</tr>
<tr>
<td></td>
<td>DBLP-Scholar</td>
<td>DBLP-ACM</td>
<td>97.2</td>
<td>97.2</td>
<td>96.2</td>
<td>96.9</td>
<td>97.1</td>
</tr>
<tr>
<td></td>
<td>DBLP-ACM</td>
<td>DBLP-Scholar</td>
<td>77.8</td>
<td>91.5</td>
<td>88.9</td>
<td>84.2</td>
<td>92.1</td>
</tr>
<tr>
<td></td>
<td>Zomato-Yelp</td>
<td>Fodors-Zagats</td>
<td>85.4</td>
<td>92.2</td>
<td>87.7</td>
<td>89.1</td>
<td>94.5</td>
</tr>
<tr>
<td></td>
<td>Fodors-Zagats</td>
<td>Zomato-Yelp</td>
<td>47.6</td>
<td>64.5</td>
<td>72.6</td>
<td>49.6</td>
<td>29.7</td>
</tr>
<tr>
<td>Different Domain</td>
<td>RottenTomatoes-IMDB</td>
<td>Abt-Buy</td>
<td>40.6</td>
<td>43.6</td>
<td>41.4</td>
<td>42.7</td>
<td>23.8</td>
</tr>
<tr>
<td></td>
<td>RottenTomatoes-IMDB</td>
<td>Walmart-Amazon</td>
<td>38.4</td>
<td>41.5</td>
<td>41.9</td>
<td>49.0</td>
<td>35.1</td>
</tr>
<tr>
<td></td>
<td>iTunes-Amazon</td>
<td>DBLP-ACM</td>
<td>80.3</td>
<td>94.5</td>
<td>86.9</td>
<td>92.1</td>
<td>57.7</td>
</tr>
<tr>
<td></td>
<td>iTunes-Amazon</td>
<td>DBLP-Scholar</td>
<td>68.2</td>
<td>86.9</td>
<td>80.4</td>
<td>85.4</td>
<td>59.6</td>
</tr>
<tr>
<td></td>
<td>Book2</td>
<td>Fodors-Zagats</td>
<td>49.6</td>
<td>91.5</td>
<td>78.2</td>
<td>84.2</td>
<td>93.5</td>
</tr>
<tr>
<td></td>
<td>Book2</td>
<td>Zomato-Yelp</td>
<td>67.4</td>
<td>73.0</td>
<td>68.0</td>
<td>54.0</td>
<td>63.3</td>
</tr>
</tbody>
</table>
Effect of Domain Shift Reduction

- Datasets: Abt-Buy (Source) → Walmart-Amazon (Target).
- Distributions of source and target are much closer after DA (b) than without DA (a).
Comparison with a Few Labels.

- The performance of the model after DA can always be maintained at a high level with some labeled data.
We define the problem of DA for deep ER.

We propose the DADER framework consisting of three modules, namely Feature Extractor, Matcher and Feature Aligner.

We systematically explore the design space of DADER and develop six representative methods.

We conduct a thorough evaluation and show that DADER can significantly advance deep ER by applying DA.
Thanks
Comparison with Instance-level DA Methods

- DADER that learns domain-invariant and discriminative features is better than instance-based method that reweight source data.