Large-Scale Deduplication with Constraints using Dedupalog

Stavros Sintos, Yahui Xiong

Paper is by Arvind Arasu, Christopher Re, Dan Suciu

Some of the slides are taken from a presentation found online.
Take-away points

- Sophisticated constraints can help the deduplication procedure.
- Using Dedupalog we can write complex constraints.
- Dedupalog is scalable to very large datasets.
- The main algorithm is $O(1)$-approximation of the optimum deduplication result.
- Dedupalog handles both hard and soft constraints.
- Empirical linear time on the number of soft constraints.
Collective deduplication

- A generalization in which one wants to find types of real-world entities in a set of records that are related.
- Output: a set of several partitions of the input records (by entity type)
Example of a sophisticated constraint:

“author references that do not share any common coauthors do not refer to the same author”

Related work

- Collective deduplication in the presence of constraints

- Markov Logic Networks for collective deduplication with constraints
  - [Singla et al. 2006] #P-complete to evaluate Markov logic network even for simple constraints, do not scale to large data sets.
Issues of existing approaches

- Lack of scalability: complicated inference
- Allow clustering of only a single entity type in isolation
- Ignore Constraints
- Use Constraints in an *ad-hoc* way
Motivation and formal model

1. Provide a set of input tables that contain references to be deduplicated and other useful info (e.g. results of similarity computation)
2. Define a list of entity references to deduplicate (e.g. authors, papers, publishers)
3. Define a Dedupalog program
4. Execute the Dedupalog program
Contributions

- Motivate the Dedupalog language by showing how to construct a sophisticated clustering program; specify the semantics of Dedupalog.
- Main algorithm with theoretical guarantees; physical optimizations to achieve scalability.
- Algorithm achieves high precision and recall on standard datasets such as Cora, \( p = .97, r = .93 \); adding constraints to increase precision and recall on large, real datasets; interactive deduplication system
- Two practical extensions
Outline

- Preliminaries
- Algorithms
- Experiments
- Discussion
- Conclusion
Preliminaries
Definitions

- Declare a list of entity reference relations. $R! (x)$
- Dedupalog creates clustering relations $R \ast (x, y)$
Rules via an example

<table>
<thead>
<tr>
<th>id</th>
<th>pos</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>A. Gionis</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>H. Manilla</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>P. Tsaparas</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>A. Gionis</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>I. Bhattacharya</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>L. Getoor</td>
</tr>
</tbody>
</table>

\[ \text{Write}(id, pos, author) \]

<table>
<thead>
<tr>
<th>id</th>
<th>Title</th>
<th>Publication Venue</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cluster Aggregation</td>
<td>&quot;ICDE&quot;</td>
<td>2005</td>
</tr>
<tr>
<td>2</td>
<td>Clustering Aggregations</td>
<td>&quot;Conference on Data Engineering&quot;</td>
<td>2005</td>
</tr>
<tr>
<td>3</td>
<td>Collective entity resolution in relational data</td>
<td>&quot;Data Eng. Bull.&quot;</td>
<td>2007</td>
</tr>
<tr>
<td>4</td>
<td>Collective entity resolution in relational data</td>
<td>&quot;Data Engineering&quot;</td>
<td>2007</td>
</tr>
</tbody>
</table>

\[ \text{PaperRefs}(id, title, venue, year) \]
Entity Reference Tables

\[
\text{Paper}!(id) := \text{PaperRefs}(id, -, -, -)
\]
\[
\text{Publisher}!(p) := \text{PaperRefs}(-, -, p, -)
\]
\[
\text{Author}!(id, pos) := \text{Wrote}(id, pos, -)
\]
Soft-complete Rules

- “papers with similar titles are likely duplicates”

\[
\text{Paper} \ast (id, id') \leftrightarrow \text{PaperRefs}(id, t, -, -, -), \\
\text{PaperRefs}(id', t', -, -, -), \\
\text{TitleSimilar}(t, t')
\]

- Paper references whose titles appear in TitleSimilar are likely to be clustered together.
- Paper references whose titles do not appear in TitleSimilar are not likely to be clustered together.
Soft-incomplete Rules

“papers with very similar titles are likely duplicates”

\[
\text{Paper} \ast (id, id') \leftarrow \text{PaperRefs}(id, t, -, -, -), \\
\text{PaperRefs}(id', t', -, -, -), \\
\text{TitleVerySimilar}(t, t')
\]

✓ Paper references whose titles appear in TitleVerySimilar are likely to be clustered together.

✓ *This rule says nothing about paper references whose titles do not appear in TitleVerySimilar.
Hard Rules

- “the publisher references listed in the table PublisherEQ must be clustered together”
- “the publisher references in PublisherNEQ must not be clustered together”

\[
\text{Publisher} \ast (x, y) \leq \text{PublisherEQ}(x, y) \\
\text{\neg Publisher} \ast (x, y) \leq \text{PublisherNEQ}(x, y)
\]

must-link
cannot-link

✓ They must be satisfied in any legal clustering
Complex Hard Rules

“whenever we cluster two papers, we must also cluster the publishers of those papers”

\[ \text{Publisher} \ast (x, y) \leq \text{Publishes}(x, p_1), \]
\[ \text{Publishes}(y, p_2), \text{Paper} \ast (p_1, p_2) \]

✓ Such constraints are central to collective deduplication

✓ At most one entity reference is allowed in the body of the rule as in this example.
Complex Negative Rules

- “two distinct author references on a single paper cannot be the same person”

\[ \neg \text{Author} (x, i, y, j) \leq \text{Wrote}(p, x, i), \text{Wrote}(p, y, j), i \neq j \]
Recursive Rules

“Authors that do not share common coauthors are unlikely to be duplicates”

\[ \neg \text{Author} \star (x, i, y, j) \leftarrow (\text{Wrote}(x, i, -), \text{Wrote}(y, j, -), \text{Wrote}(x, p, -), \text{Wrote}(y, p', -), \text{Author} \star (x, p, y, p')) \]
Cost model

- Cost of clustering $J$: Number of tuples in the output of soft-rules that are violated.

If $\gamma$ is soft-complete,
\[
\text{Cost}(\text{HEAD}(i, j) \leftrightarrow \text{BODY}, J^*) \overset{\text{def}}{=} \{| \{i, j\} | i \neq j \text{ and } J^* \not\models \text{HEAD}(i, j) \leftrightarrow \text{BODY}(i, j)\|\}
\]

else, $\gamma$ is soft-incomplete and its cost on $J^*$ is
\[
\text{Cost}(\text{HEAD}(i, j) \leftarrow \text{BODY}, J^*) \overset{\text{def}}{=} \{| \{i, j\} | i \neq j \text{ and } J^* \not\models \text{HEAD}(i, j) \leftarrow \text{BODY}(i, j)\|\}
\]

- Cost of a valid clustering $J^*$
\[
\text{Cost}(\Gamma, J^*) \overset{\text{def}}{=} \sum_{\gamma \in \Gamma_{\text{Soft}}} \text{Cost}(\gamma, J^*)
\]
Cost model

Example 2.1: Consider a program with a single soft-complete rule $R \ast (p1, p2) \leftrightarrow E(p1, p2)$ where $I$ is such that $R! = \{a, b, c, d\}$ and let $E$ be the symmetric closure of $\{(a, b), (a, c), (c, d)\}$. This is graphically represented in Fig. 2. Any partition of $R!$, e.g., $\{\{a, b, c\}, \{d\}\}$, is a valid clustering; this partitioning is illustrated in Fig. 2(b).

The cost is 2 because of two violations:

1) $d$ belongs in the same cluster as $c$

2) $c$ does not belong in the same cluster as $b$
Problem Definition

**Deduplication Evaluation Problem:** Given a constraint program $\Gamma$ and an input instance $I$, construct a valid clustering $J^*$ of $I$ such that $J^*$ minimizes $\text{Cost}(\Gamma, J^*)$.

- NP-hard, even for a single soft-complete constraint.
- In many cases the problem is hard to approximate.
- Next, we show a $O(1)$-approximation for a large fragment of Dedupalog.
Algorithms
Clustering graph

- A graph $G(V, E)$
- A symmetric label function on edges: $\phi: \binom{V}{2} \rightarrow \{[+], [-], [=], [\neq]\}$
  - Nodes $V$: entity references
  - Soft-plus, hard-plus, soft-minus, hard-minus
- Goal: Find a clustering of $G$ such that $\forall e = (u, v) \in E$
  - $u, v$ belong in the same cluster if $\phi(e) = [=]$
  - $u, v$ belong in different clusters if $\phi(e) = [\neq]$

Minimizing

$$\sum_{e} 1[\phi(e) = [+] \&\& C(u) \neq C(v)]||\phi(e) = [-] \&\& C(u) = C(v)]$$
Clustering graph Algorithm

- If no hard constraints then Clustering Graph = correlation clustering.
  - \( O(1) \)-approximation algorithm

**New Algorithm**

- Uniformly choose a random permutation of the nodes
- This gives a partial order on the edges
- Harden each edge in order:
  - Change soft edges into hard edges
  - Apply these two rules:
    1) If \( \phi(u, v) = [\neq] \) and \( \phi(v, w) = [\neq] \), set \( \phi(u, w) = [\neq] \)
    2) If \( \phi(u, v) = [\neq] \) and \( \phi(v, w) = [\neq] \), set \( \phi(u, w) = [\neq] \)
- A clustering is all \([=] \) connected components
Clustering Graphs Algorithm

**Theoretical guarantee:**

\[
E_{\pi}[\text{Cost}(\mathcal{R}^*, V, \phi)] \leq 3 \text{ Cost}(\text{Opt}, V, \phi)
\]
From Clustering Graph to Deduplication

- 2-stages Algorithm
  - Forward-voting: From $\Gamma$, produces a list of clustering graphs $G_1, \ldots, G_n$ where $n$ is the number of entity references.
  - Backward-propagation: Take the $n$ graphs found by Forward-voting and using the Clustering Graph algorithm produce the clusterings $R_1*, \ldots, R_n*$. 
Overall Algorithm

Fix a constraint program $\Gamma$ with $\Gamma_E = \{R_1!, \ldots, R_n!\}$ and instance $\mathcal{I}$

CLUSTER\textsc{Many}($G_L = \{G_1, \ldots, G_{i-1}\}$, $E_R = \{R_1!, \ldots, R_n!\}$)

Input A set of compiled graphs $G_L$
Input A set of to-be-compiled entity references $E_R$
Output A clustering $R_i*$ of each $G_j$ for $j = i, \ldots, n$.

1. If $E_R = \emptyset$ then return $\emptyset$.
2. Else use $R_1!$ to perform Forward-voting to produce $G_i$
4. Let $\mathcal{R} = \text{CLUSTER\textsc{Many}}(G_L \cup G_i, E_R - \{R_1!\})$.
5. Backward-propagate $\mathcal{R}$ to $G_i$ and cluster to produce $R_i*$
6. Return $\{R_i*\} \cup \mathcal{R}$
Forward-voting (assume no recursive rule)

- For \(i = 1, \ldots, n\) let \(\Gamma^{(i)}\) be the set of rules in \(\Gamma\) that have \(R_i \ast\) in the HEAD.
- Order the entity references \(R!_1, \ldots, R!_n\) such that:
  - If \(\exists \gamma \in \Gamma\) with \(R_i \ast \in BODY\gamma\) and \(R_j \ast \in HEAD\gamma\) then \(i \leq j\)
- Example: \(R_3 \ast (x, y) < \rightarrow \cdots, R_2 \ast (x, y)\)
- Goal: produce the clustering graph \(G_i = (V_i, \phi_i)\) corresponds to \(R_i!\)
  - Nodes \(V_i\) are the entity references in \(R_i!\)
  - How to label the edges? Assign \(\phi_{-i}(\cdot)\) values?
Assign $\phi_i()$ labeling

- Intuition: Each constraint in $\Gamma^{(i)}$ offers a vote for the label assigned to an edge $e$ in $G_i$.

- For each rule $\gamma \in \Gamma^{(i)}$ and for each entry s.t. $\gamma$ satisfies the conditions listed in the first two columns

<table>
<thead>
<tr>
<th>Head</th>
<th>Hard/Soft</th>
<th>Voting Datalog Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>All</td>
<td>$q_{\gamma}^{(+1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(+1)}]$</td>
</tr>
<tr>
<td>Positive</td>
<td>Soft</td>
<td>$q_{\gamma}^{(+1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(+1)}]$</td>
</tr>
<tr>
<td>Positive</td>
<td>Soft-C</td>
<td>$q_{\gamma}^{(-1)}(\bar{x}, \bar{y})$ :- $q_{\gamma}^{(+1)}(\bar{x}, \bar{y}), R_1!(\bar{x}), R_1!(\bar{y})$</td>
</tr>
<tr>
<td>Positive</td>
<td>Hard</td>
<td>$q_{\gamma}^{(-1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(-1)}]$</td>
</tr>
<tr>
<td>Negative</td>
<td>All</td>
<td>$q_{\gamma}^{(-1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(-1)}]$</td>
</tr>
<tr>
<td>Negative</td>
<td>Soft</td>
<td>$q_{\gamma}^{(-1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(-1)}]$</td>
</tr>
<tr>
<td>Negative</td>
<td>Soft-C</td>
<td>$q_{\gamma}^{(+1)}(\bar{x}, \bar{y})$ :- $q_{\gamma}^{(-1)}(\bar{x}, \bar{y}), R_1!(\bar{x}), R_1!(\bar{y})$</td>
</tr>
<tr>
<td>Negative</td>
<td>Hard</td>
<td>$q_{\gamma}^{(+1)}(\bar{x}, \bar{y})$ :- BODY$<em>{\gamma}[R_3 \star \rightarrow E</em>{\gamma}^{(+1)}]$</td>
</tr>
</tbody>
</table>

- Create a voting query
Example

Publisher * (x, y) <= Publishes(x, p_1), Publishes(y, p_2), Paper * (p_1, p_2)

Replace any occurrence of $R_j *$ for $j < i$ in the BODY of $\gamma$ with either $E_j^{[+]_i}$ or $E_j^{[=]}$.

$E_j^{[+]_i} = \{ e | \phi_j(e) = [+])$, the current [+] edges in $j$. 
Electing edge labels

- Intuition: If there is vote for a hard label $h$ then $h$ is the label. Otherwise $\phi(u, v)$ takes the majority level.

```
Input A set $V_i$ of nodes and voting queries for each $\gamma \in \Gamma^{(i)}$
Foreach $\{u, v\} \in \binom{V_i}{2}$
If exists $\gamma \in \Gamma^{(i)}$ such that $\{u, v\} \in q_\gamma^h$ for $h \in \{=, \neq\}$
then $\phi(u, v) = h.$
else (* No hard edge, take the majority vote *)

PLUS VOTES$(u, v) = |\{\gamma | \{u, v\} \in q_\gamma^{[+]\gamma}\}|$
MINUS VOTES$(u, v) = |\{\gamma | \{u, v\} \in q_\gamma^{[-]}\}|$
if PLUS VOTES$(u, v) \geq$ MINUS VOTES$(u, v)$ then
$\phi(u, v) = [+]$ else $\phi(u, v) = [-].$
```
Backward-propagation

- Opposite order of Forward-voting
- For $i = 1, ..., n$ let $\Gamma_{(i)}$ be the set of rules in $\Gamma$ that have $R_i \ast$ in the BODY.
- For $\gamma \in \Gamma_{(i)}$, $R_k \ast = \text{HEAD}_\gamma$ for some $k > i$.
- If $\{x, y\} \notin R_k \ast$ then for any pair $\{u, v\}$ such that $\gamma(x, y)$ holds whenever $\{u, v\} \in R_i \ast$, then clustering $u, v$ together would violate a hard constraint.
- Set $\phi_i(u, v) = \neq$
- Cluster $G_i$ using ClusterGraph algorithm.
Recursive constraints

- Are confined to a single graph.
- Are always soft.
- Executing Clustering Graph algorithm, keep track of the votes for each edge and which rule cast the vote.
- Use classical techniques to incrementally evaluate Datalog.
Theorem 3.2: If there exists a valid clustering for a constraint program $\Gamma$ on entity relations $R_1!, \ldots, R_n!$ and input instance $I$, then CLUSTERMANY (Fig. 4) returns a valid clustering $J^*$. Let $\text{Opt}$ be the optimal clustering of $\Gamma$ and $I$. If $\Gamma$ is such that for any hard rule $\gamma \in \Gamma$, $\text{BODY}_\gamma$ contains no clustering relations, then CLUSTERMANY returns a clustering that has cost within a constant factor of the optimal. Formally,

$$E[\text{Cost}(\Gamma, J^*)] \leq k \cdot \text{Cost}(\Gamma, \text{Opt})$$

where $k = 6 \max_i |\Gamma^{(i)}|$ and $E$ is over the choices of CLUSTERMANY.
Physical Implementation and Optimization

- Implicit Representation of Edges
  - Quadratic running time storing all edges explicitly.
  - Explicitly store $[+]$ edges but implicitly represent $[-]$ edges (complement).

- Choosing edge orderings
  - Optimize Clustering Graph Algorithm
  - Process all $[+]$ neighbors of a node $i$ and then all $[-]$.
  - Choose $\pi$ such that all $[+]$ edges before $[-]$.
  - Why? $[-]$ neighbors will never be included in the same cluster as $i$.
  - $[-]$ edges will be converted to $\neq$ edges

- Sort optimization
  - Assume no hard constraints.
  - Sort $[+]$ edges according to $\pi$.
  - Find clustering directly in this order running the Clustering Graph Algorithm.
Experiments
Experiments: Measures

- Two standard measures for clustering, precision and recall.
- Let $T$ be the ground truth:

$$\text{Precision}(J^*, T) \overset{\text{def}}{=} \frac{|J^* \cap T|}{|J^*|} \text{ and Recall}(J^*, T) \overset{\text{def}}{=} \frac{|J^* \cap T|}{|T|}$$
Experiments: Implementation details

- **Hardware:**
  - Intel Core2 6600 at 2.4Ghz with 2GB of RAM
  - Windows Vista Enterprise
  - SQL Server 2005
  - Performance: average of 5 runs, standard deviation < 3% of the total execution time in all cases.
    - Warm-cache, the sole process executing on the system.

- **Implementation:**
  - A main clustering library and console application written in approximately 3000 lines of C#.
  - A GUI application built to aid in deduplication, written in approximately 1700 lines of C#.

- **Similarity scores:** Standard TF-IDF scoring
Quality: Cora

- Standard: matching titles and running correlated clustering
- NEQ: Standard + an additional hard rule constraint—lists conference papers that were known to be distinct from their journal papers
Quality: Cora

- Standard: Clustering based on string similarity
- Soft Constraint: Standard + soft constraint—”papers must be in a single conference”
- Hard Constraint: Standard + hard constraint—”papers must be in a single conference”
Quality: ACM dataset

- No Constraints: String similarity and correlation clustering
- Constraints: No Constraints + hard constraint—”references with different years, do not refer to the same conference”
ACM dataset

- Helps catches errors in records.
- Added a hard rule that says: “If two references refer to the same paper, then they must refer to the same conference”
- On the subset it found 5 references that contained incorrect years
- On the full dataset it found 152 suspect papers
Performance

<table>
<thead>
<tr>
<th>Scale ($p$)</th>
<th>Nodes ($n$)</th>
<th>$n^2$</th>
<th>[+] Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Conf</td>
</tr>
<tr>
<td>0.001</td>
<td>570</td>
<td>324k</td>
<td>4</td>
</tr>
<tr>
<td>0.01</td>
<td>5403</td>
<td>29.2M</td>
<td>742</td>
</tr>
<tr>
<td>0.1</td>
<td>52k</td>
<td>2.81B</td>
<td>78k</td>
</tr>
<tr>
<td>0.5</td>
<td>266k</td>
<td>70.9B</td>
<td>2.0M</td>
</tr>
<tr>
<td>1.0</td>
<td>531k</td>
<td>282B</td>
<td>7.6M</td>
</tr>
</tbody>
</table>

Fig. 10. Statistics about the data. The number of nodes in each dataset $n$, the number of nodes squared $n^2$ and the number of positive edges for three entities: Conferences, Paper titles and Publishers.
Performance

• **Vanilla**: Clustering of the references by conference with a single soft constraint

• **[=]**: Vanilla with two additional hard constraints

• **HMorphism**: Vanilla + [=] + Cluster conferences and papers with the constraint—”conference papers appear in only one conference”

• **NoStream**: Vanilla with sort-optimization off
Interactive Deduplication

- Manual clustering of Cora took a couple of hours—98% precision and recall.
- Obtaining ground truth for the ACM subset to only 4 hours
Discussion

- Weight and Don’t care edges
  - Assign weights to the rules
  - Likely, it does not have a constant approximation.
- Clean Entity Lists
  - List of clean conference names.
  - Use cleaned list and the dirty data in Dedupalog program.
  - Problem is NP-hard.
Conclusion

😊 Proposed dedupalog, a language for deduplication
😊 Efficiently cluster large datasets w/ high-precision recall
😊 Novel theoretical analysis and implementation

😊 Different datasets?
😊 Do not show all the proofs.
Thank you

Questions?