

Blocking Schemes for Record Linkage

Matthew Michelson

CSCI 548

2006

Record Linkage – Finding Matches

Census Data

<i>First Name</i>	<i>Last Name</i>	<i>Phone</i>	<i>Zip</i>
Matt	Michelson	555-5555	12345
Jane	Jones	555-1111	12345
Joe	Smith	555-0011	12345

A.I. Researchers

<i>First Name</i>	<i>Last Name</i>	<i>Phone</i>	<i>Zip</i>
Matthew	Michelson	555-5555	12345
Jim	Jones	555-1111	12345
Joe	Smeth	555-0011	12345

match



match



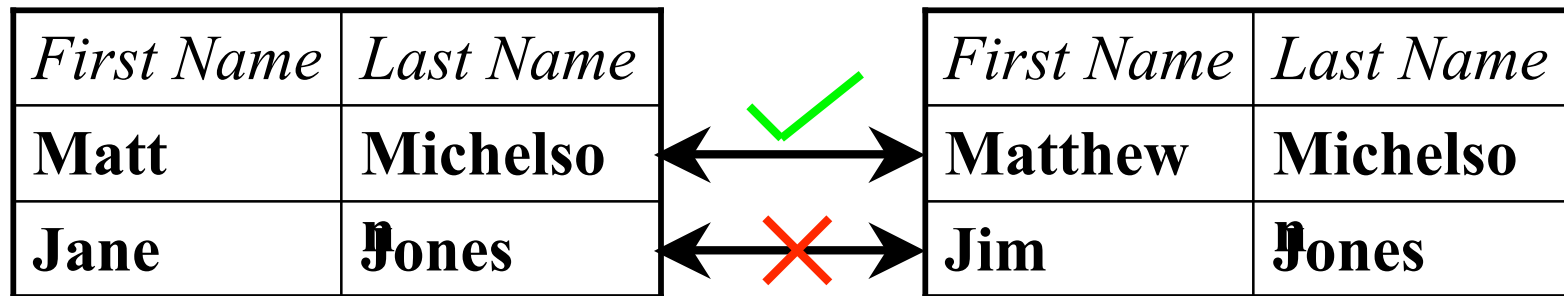


Record Linkage – Finding Matches

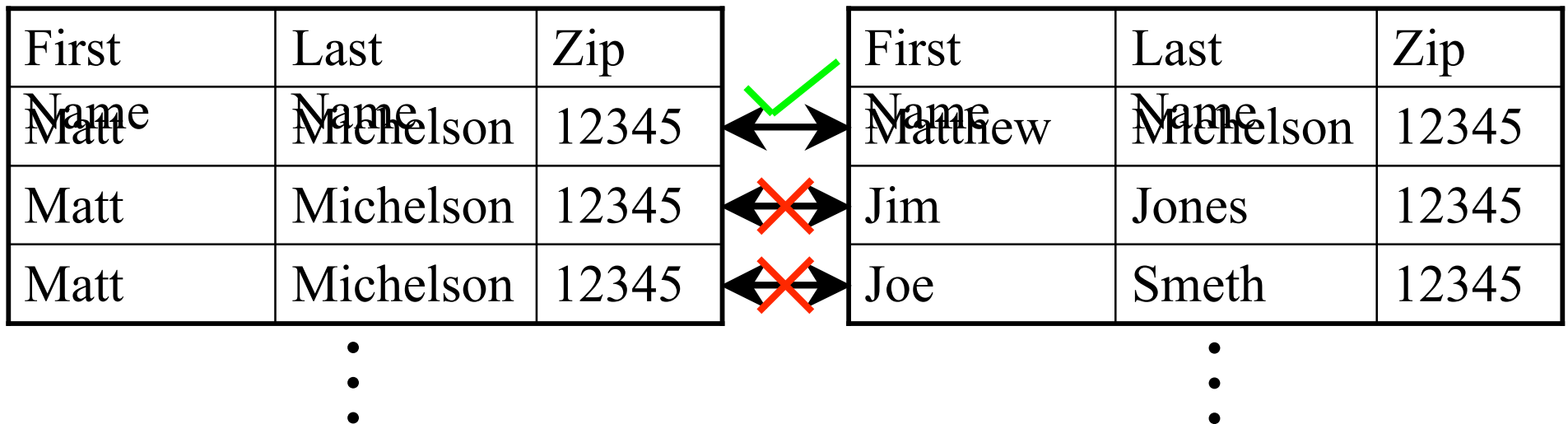
- Can't compare all records!
 - Just 5,000 to 5,000 → 25,000,000 comparisons!
 - At 0.01s/comparison → 250,000 s → ~3 days!
- Need to use a subset of comparisons
 - “Candidate matches”
 - Want to cover true matches
 - Want to throw away non-matches

Blocking – Generating Candidates

(token, last name) AND (1st letter, first name) = block-key



(token, zip)

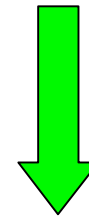


Blocking - Intuition

Census Data

<i>First Name</i>	<i>Last Name</i>	<i>Zip</i>
Matt	Michelso	12345
Jane	Jones	12345
Joe	Smith	12345

Zip = '12345'



1 Block of **12345** Zips

→ Compare to the “block-key”

Group & Check to reduce Checks



Bi-Gram Indexing

- Can we use a better blocking key than tokens?
 - What about “fuzzy” tokens?
 - Matt → Matthew, William → Bill? (similarity)
 - Michael → Mychael (spelling)
- Bi-Gram Indexing
 - Baxter, Christen, Churches, **A Comparison of Fast Blocking Methods for Record Linkage**, ACM SIGKDD, 2003

Bi-Gram indexing

- **Step 1:** Take token and break it into bigrams
 - Token: **matt**
 - ('**ma**,' '**at**,' '**tt**,')
- **Step 2:** Generate all sub-lists
 - (# bigrams) x (threshold) = sub-list length
 - **3** x **.7** = 2
- **Step 3:** Sort sub-lists and put them into inverted index
 - ('**at**' '**ma**') ('**at**' '**tt**') ('**ma**' '**tt**') → record w/ **matt**
 - └── Block key



BiGram Indexing: Properties

- Threshold properties
 - lower = shorter sub-lists → more lists
 - higher = longer sub-lists → less lists, less matches
- Now we can find spelling mistakes, close matches, etc...



Blocking – Multi-pass

- Sort neighborhoods on block keys
- Multiple independent runs using keys
 - runs capture different match candidates
- Attributed to (Hernandez & Stolfo, 1998)
- E.g.) 1st → (token, last name)
2nd → (token, first name) &
(token, phone)



Blocking – Multi-pass

- Can we make blocks without sorting?
 - Yes! We can cluster...










Blocking – Canopies Method

McCallum, Nigam, Ungar, **Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching**, 2000, KDD

Idea: form clusters around certain key values, within some threshold value

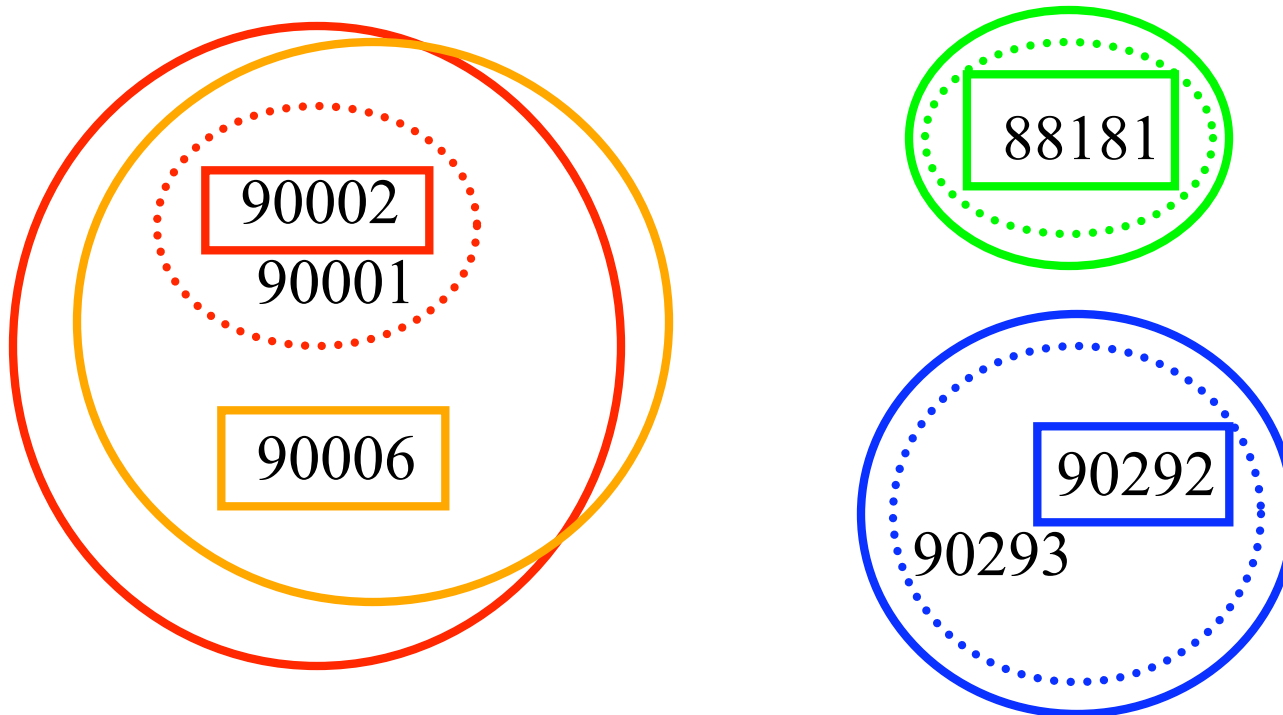
Blocking – Canopies Method

-  Start with 2 threshold values, $T1$ and $T2$, s.t. $T1 > T2$
 -  based on similarity function, hand picked or learned thresholds
-  Select a random record from list of records and calculate it's similarity to all other records
 -  Very cheap in some cases: inverted index
-  Create “Canopy” for all records where similarity less than $T1$
-  Remove all records form the list of records where similarity less than $T2$
-  Repeat 1-4 until your list is empty

Blocking – Canopies Method

- Sim. function = abs. zip distance, $T1 = 6$, $T2 = 3$

List of records: ~~90001~~, ~~90002~~, ~~90006~~, ~~88181~~, ~~90292~~, ~~90293~~





Blocking – Multi-pass

- Back to the world of multi-pass...
- Terminology:
 - Each pass is a “conjunction”
 - (token, first) AND (token, phone)
 - Combine passes to form “disjunction”
 - [(token, last)] OR [(token, first) AND (token, phone)]
 - Disjunctive Normal Form rules
 - form “Blocking Schemes”



Blocking Effectiveness

- Determined by rules
 - Determined by choices for attributes and methods
 - (token, zip) captures all matches, but all pairs too
 - (token, first) AND (token, phone) gets half the matches, and only 1 candidate generated
 - Which is better? Why?
 - How to quantify??

Blocking Effectiveness

$$\text{Reduction Ratio (RR)} = 1 - \|C\| / (\|S\| * \|T\|)$$

S,T are data sets; C is the set of candidates

$$\text{Pairs Completeness (PC) [Recall]} = S_m / N_m$$

S_m = # true matches in candidates,

N_m = # true matches between S and T

Examples: **(token, last name) AND (1st letter, first name)**

$$\text{RR} = 1 - 2/9 \approx 0.78$$

$$\text{PC} = 1 / 2 = 0.50$$

(token, zip)

$$\text{RR} = 1 - 9/9 = 0.0$$

$$\text{PC} = 2 / 2 = 1.0$$



Multi-Pass Blocking Schemes

Old Techniques: Ad-hoc rules

New Techniques: Learn rules!

Learned rules justified by quantitative effectiveness

Michelson & Knoblock, Learning Blocking Schemes for Record Linkage, 2006, AAAI



How to choose methods and attributes?

- Blocking Goals:
 - Small number of candidates (High **RR**)
 - Don't leave any true matches behind! (High **PC**)
- Previous approaches:
 - Ad-hoc by researchers or domain experts
- New Approach:
 - Blocking Scheme Learner (BSL) – modified Sequential Covering Algorithm

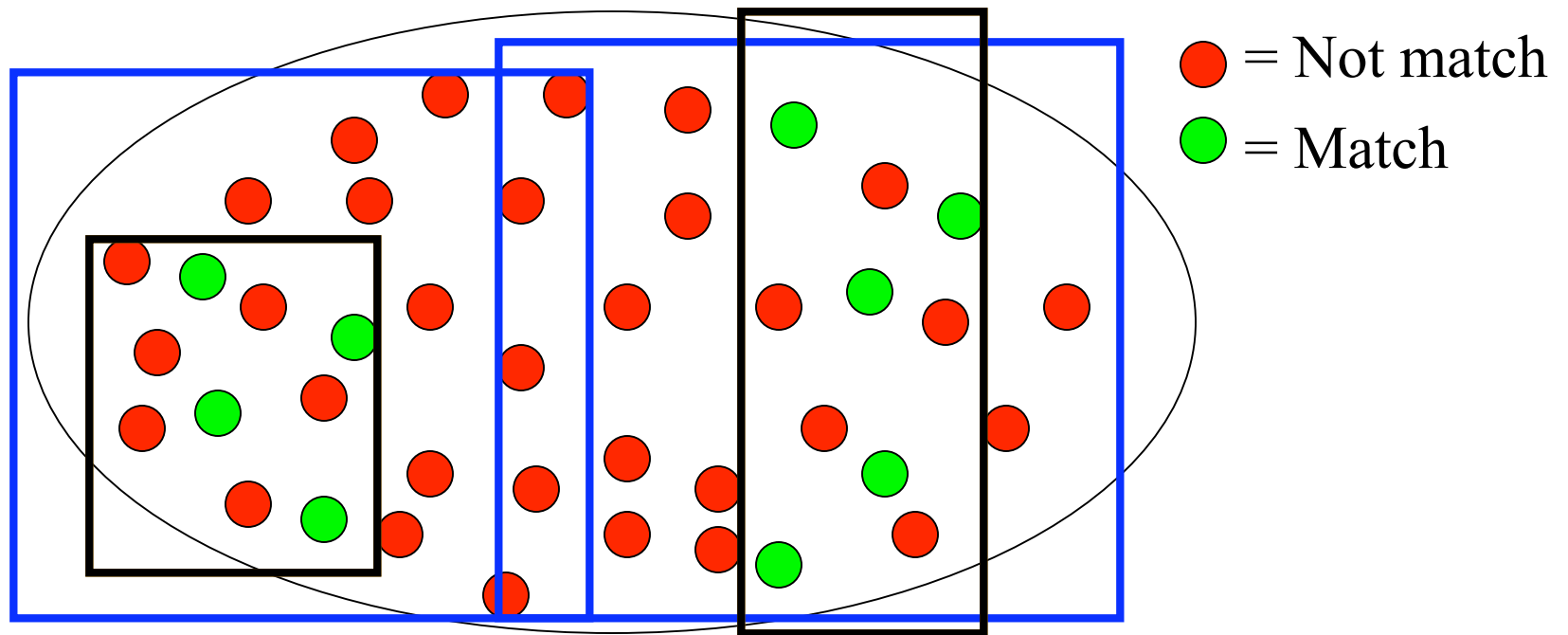


Learning Schemes – Intuition

- Learn restrictive conjunctions
 - partition the space → minimize False Positives
- Union restrictive conjunctions
 - Cover all training matches
 - Since minimized FPs, conjunctions should not contribute many FPs to the disjunction

Example to clear things up!

Space of training examples



Rule 1 :- (zip|token) & (first|token)
Final Rule :- [(zip|token) & (first|token)] UNION [(last|1st Letter)
& (first|1st Letter)] & (last|1st Letter) & (first|1st Letter)

SCA: propositional rules

- ❑ Multi-pass blocking = disjunction of conjunctions
- ❑ Learn conjunctions and union them together!
- ❑ Cover all training matches to maximize **PC**

```
SEQUENTIAL-COVERING( class, attributes, examples, threshold)
```

```
LearnedRules ← {}
```

```
Rule ← LEARN-ONE-RULE(class, attributes, examples)
```

```
While examples left to cover, do
```

```
    LearnedRules ← LearnedRules U Rule
```

```
    Examples ← Examples – {Examples covered by Rule}
```

```
    Rule ← LEARN-ONE-RULE(class, attributes, examples)
```

```
    If Rule contains any previously learned rules, remove them
```

```
Return LearnedRules
```



SCA: propositional rules

□ LEARN-ONE-RULE is greedy

- rule containment as you go, instead of comparison afterward

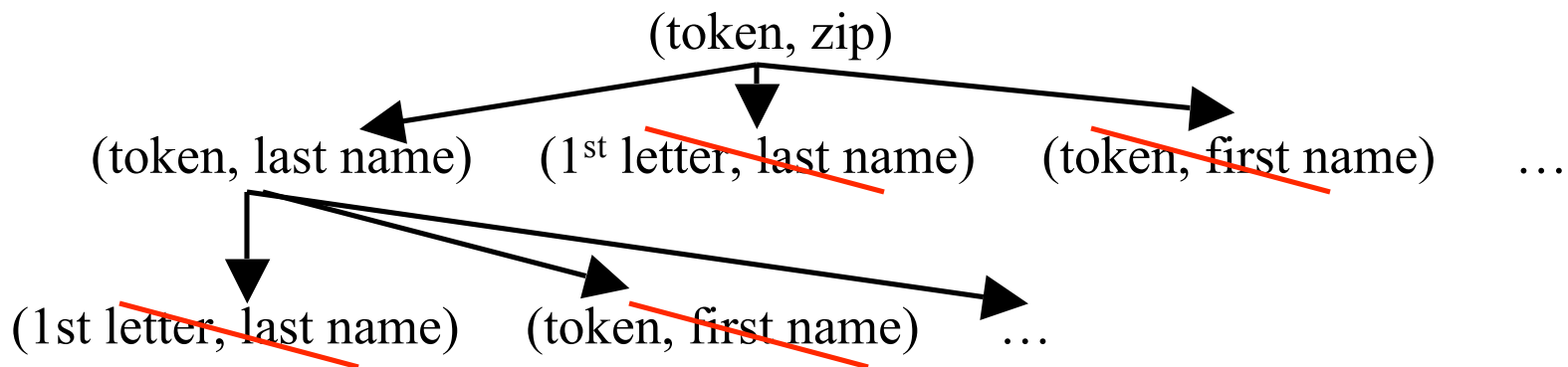
- Ex) rule: (token|zip) & (token|first)

(token|zip) CONTAINS (token|zip) & (token|first)

- Guarantee later rule is less restrictive – If not how are there examples left to cover?

Learn-One-Rule

- Learn conjunction that maximizes **RR**
 - General-to-specific beam search
 - Keep adding/intersecting (attribute, method) pairs
 - Until can't improve **RR**
 - Must satisfy minimum **PC**
-



Experiments

Cars	RR	PC
HFM	47.92	99.97
BSL	99.86	99.92
BSL (10%)	99.87	99.88

HFM = ($\{\text{token, make}\} \cap \{\text{token, year}\} \cap \{\text{token, trim}\}$)

$\cup (\{\text{1st letter, make}\} \cap \{\text{1st letter, year}\} \cap \{\text{1st letter, trim}\})$

$\cup (\{\text{synonym, trim}\})$

BSL = ($\{\text{token, model}\} \cap \{\text{token, year}\} \cap \{\text{token, trim}\}$)

$\cup (\{\text{token, model}\} \cap \{\text{token, year}\} \cap \{\text{synonym, trim}\})$

Census	RR	PC
Best 5 Winkler	99.52	99.16
Adaptive	99.9	92.7
Filtering	98.12	99.85
BSL	99.50	99.13
BSL (10%)	99.50	99.13

Restaurants	RR	PC
Marlin	55.35	100.00
BSL	99.26	98.16
BSL (10%)	99.57	93.48

Summary

	Attr, Method	Learning
Canopie	Ad-hoc	--
Bi-Gram	Ad-hoc	--
BSL	Learn	SCA (iterative)

- Tradeoffs: Learning vs. Non
 - Need to label (but already labeled for RL!), but get well justified, productive blocking
- Choice: Choose a learning method!
 - Maybe use canopies within a learning method!



Conclusions

- Automatic Blocking Schemes using Machine Learning
 - Not created by hand
 - cheaper
 - easily justified
 - Better than non-experts ad-hoc and comparable to domain expert's rules
 - Nice reductions – scalable record linkage
 - High coverage – don't hinder record linkage