

Blocking Schemes for Record Linkage

Matthew Michelson

CSCI 548

2006

Record Linkage – Finding Matches

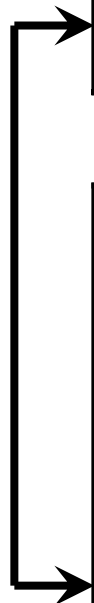
Census Data

<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

<i>First Name</i>	<i>Last Name</i>
Matt	Michelson
Jane	Jones

<i>First Name</i>	<i>Last Name</i>
Matthew	Michelson
Jim	Jones

(token, zip)

First Name	Last Name	Zip
Matt	Michelson	12345
Matt	Michelson	12345
Matt	Michelson	12345
⋮		

First Name	Last Name	Zip
Matthew	Michelson	12345
Jim	Jones	12345
Joe	Smeth	12345
⋮		

Blocking - Intuition

Census Data

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

Zip = '12345'



1 Block of **12345** Zips

→ Compare to the “block-key”

Group & Check to reduce Checks

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

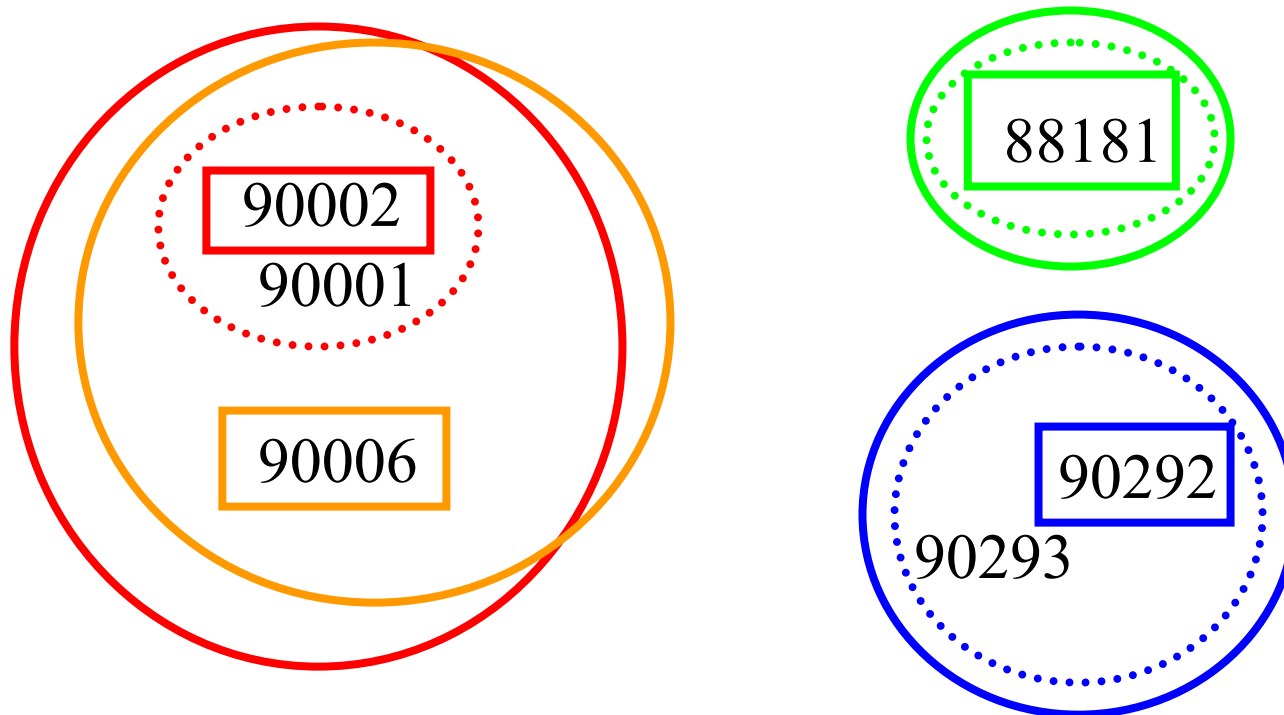
Blocking – Canopies Method

1. Start with 2 threshold values, $T1$ and $T2$, s.t. $T1 > T2$
 1. based on similarity function, hand picked or learned thresholds
2. Select a random record from list of records and calculate it's similarity to all other records
 1. Very cheap in some cases: inverted index
3. Create “Canopy” for all records where similarity less than $T1$
4. Remove all records form the list of records where similarity less than $T2$
5. 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

**Bilenko, Kamath, Mooney, Adaptive Blocking:
Learning to Scale Up Record Linkage,
2006, ICDM**

Bilenko, et. al.

blocking function \rightarrow set of (method, attribute) pairs (scheme) that cover records i and j

$$f_p^* = \arg \min_{f_p} \sum_{(x_i, x_j) \in R} f_p(x_i, x_j)$$

R is set of non-matches

$$\text{s.t. } |B| - \sum_{(x_i, x_j) \in B} f_p(x_i, x_j) < \varepsilon$$

B is the set of matches

small error threshold

Optimal
blocking
function

What does it mean? \rightarrow Select the set of blocking functions that minimize the coverage of non-matches, such that we cover as many true matches as we can, leaving only epsilon true matches behind!

Bilenko, et. al. – DNF Blocking

Algorithm: APPROXDNF

Input: Training set $\mathcal{B} = \{b_1, \dots, b_\beta\}$ and $\mathcal{R} = \{r_1, \dots, r_\rho\}$ where
each b_i is a pair of coreferent records (x_{i1}, x_{i2}) s.t. $y_{i1} = y_{i2}$
each r_i is a pair of non-coreferent records (x_{i1}, x_{i2}) s.t. $y_{i1} \neq y_{i2}$
Set of blocking predicates $\mathcal{P} = \{p_1, \dots, p_t\}$
Maximum number of coreferent pairs allowed to be uncovered ε
Maximum number of pairs that any predicate may cover η
Maximum conjunction length, k

Output: A DNF blocking function based on \mathcal{P} :

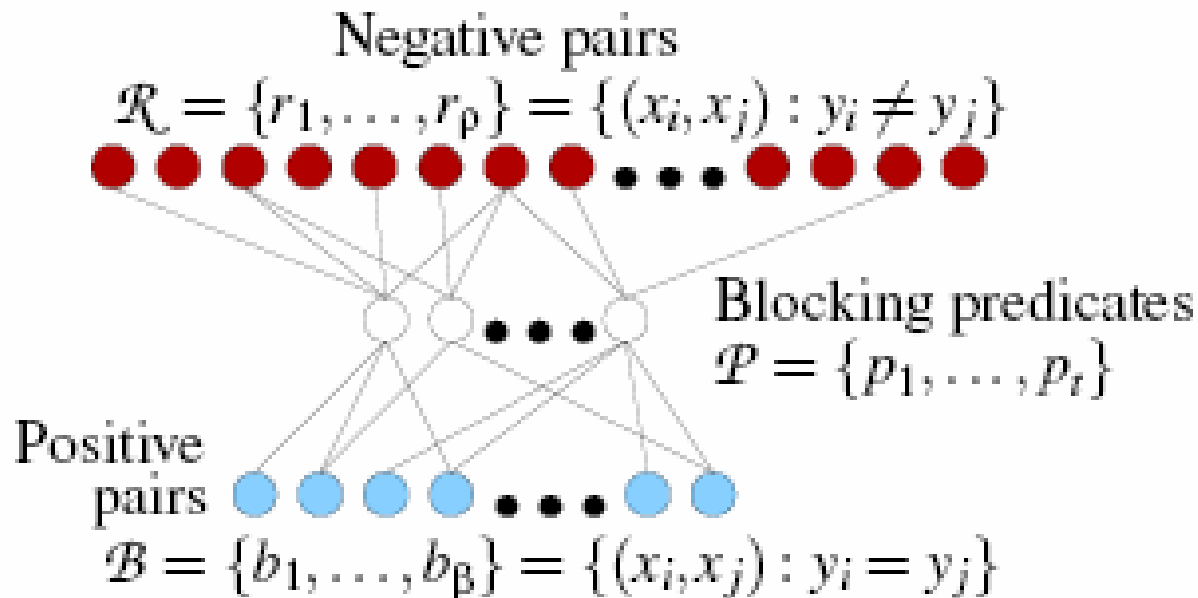
$$(p_{i_1} \wedge \dots \wedge p_{i_{j'}}) \vee \dots \vee (p_{i_n} \wedge \dots \wedge p_{i'_n}), \text{ each } i'_j \leq k$$

Method:

1. Discard from \mathcal{P} all predicates p_i for which $r(p_i) \geq \eta$:
 $\mathcal{P} \leftarrow \{p_i \in \mathcal{P} \mid r(p_i) \leq \eta\}$.
2. $\mathcal{P}^{(c)} = \emptyset$
3. For each $p_i \in \mathcal{P}$
4. Construct $k - 1$ candidate conjunctions $p_i^{(c)} = p_i \wedge \dots \wedge p_{i_k}$
by iteratively selecting p_{i_j} that maximizes cover $b(p_i^{(c)})/r(p_i^{(c)})$,
adding each $p_i^{(c)}$ to $\mathcal{P}^{(c)}$.
5. Return APPROXRBSSETCOVER($\mathcal{R}, \mathcal{B}, \mathcal{P} \cup \mathcal{P}^{(c)}, \varepsilon, \eta$).

Bilenko, et. al. – DNF Blocking

- `ApproxRBSetCover` = Red/Blue Set Cover



Optimal RB Covering = selecting subset of predicate vertices s.t.
at least $(B-\epsilon)$ blue vertices have 1 incident edge with predicates AND
number of red vertices with 1 incident edge is minimized

Bilenko, et. al. – DNF Blocking

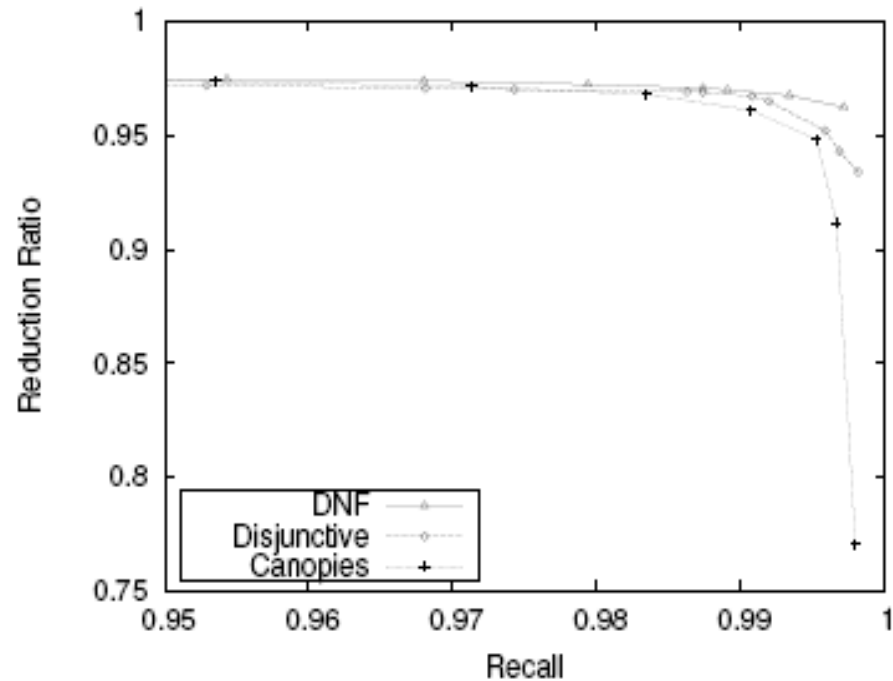


Figure 5. Blocking results for the *Cora* dataset

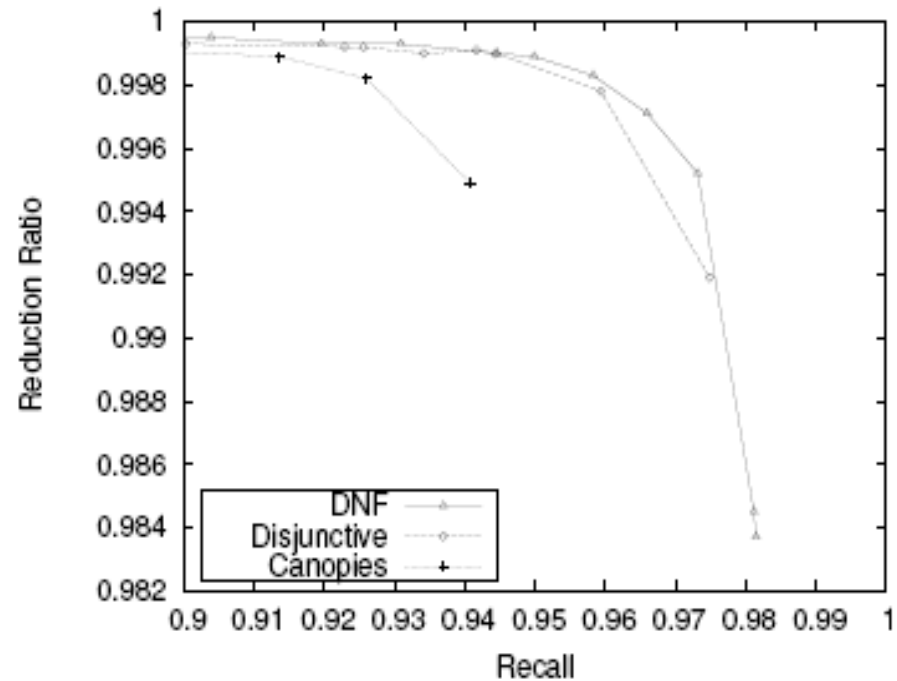


Figure 6. Blocking results for the *Addresses* dataset



Multi-Pass Blocking Schemes

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

How to choose methods and attributes?

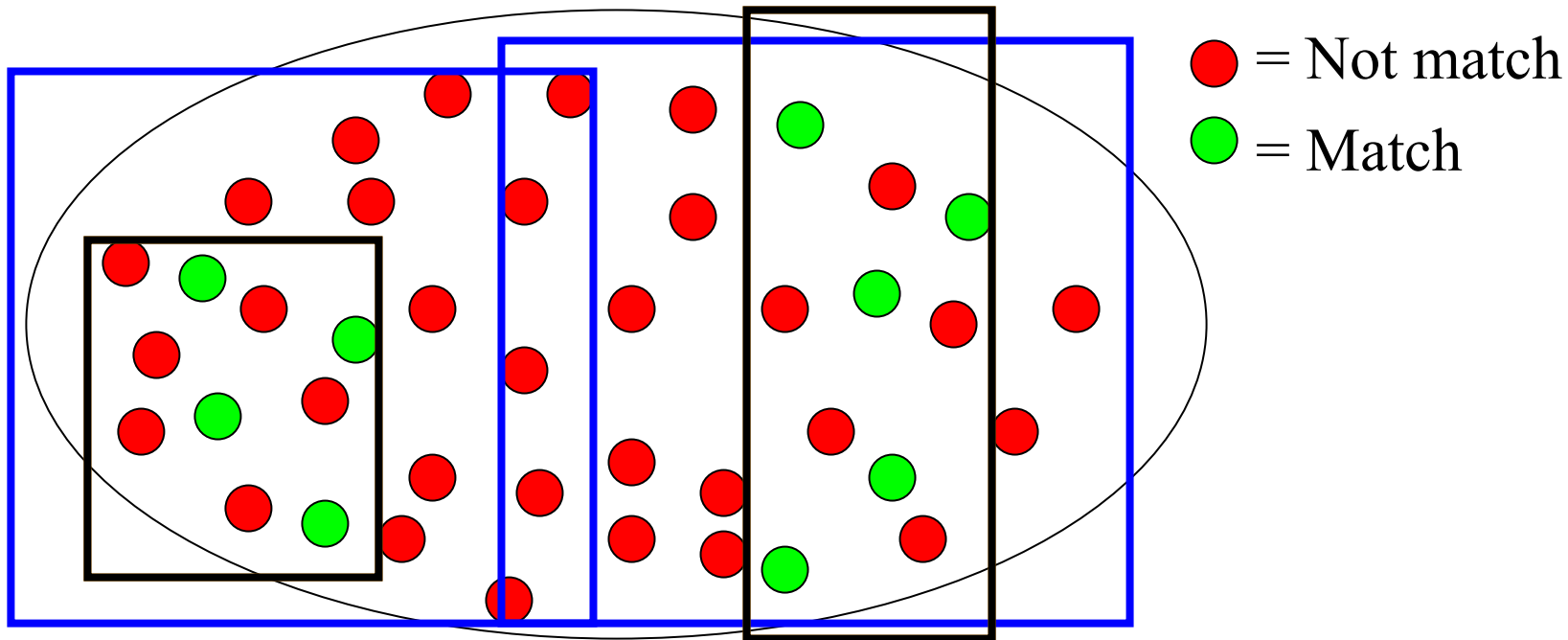
- Blocking Goals:
 - Small number of candidates (High **RR**)
 - Don't leave any true matches behind! (High **PC**)
- Previous approaches:
 - Ad-hoc by researchers or domain experts
- New Approach:
 - Blocking Scheme Learner (BSL) – modified Sequential Covering Algorithm

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
```

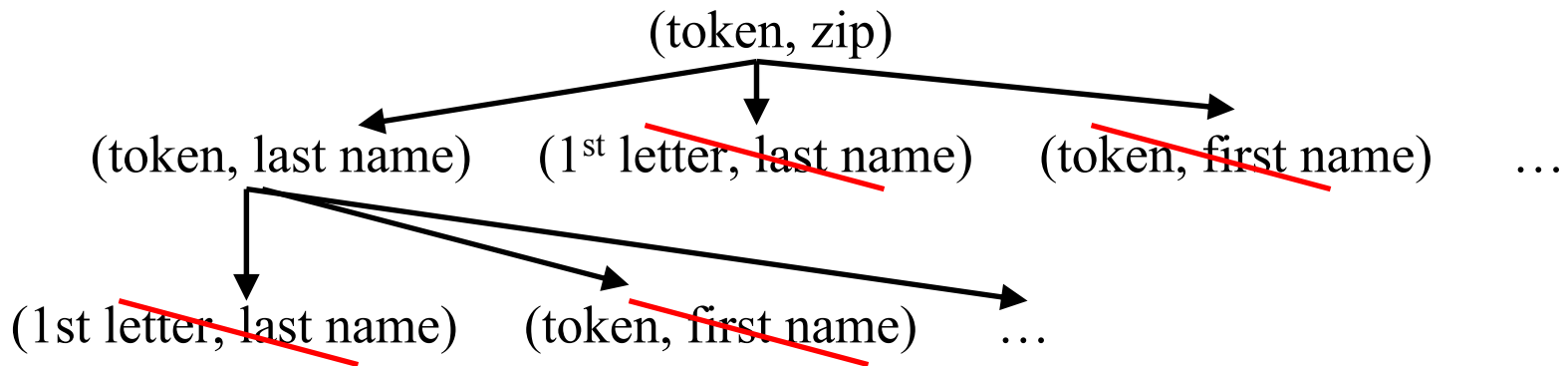
```
Return LearnedRules
```

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 Filtering	99.9	92.7
BSL	98.12	99.85
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
Canopies	Ad-hoc	--
Bilenko	Learn	RB Set Cover
BSL	Learn	SCA (iterative)

- Tradeoffs: Learning vs. Non
 - Need to label (but already labeled for RL!), but get well justified, productive blocking
 - Bilenko/BSL essentially the same
 - (developed independently at same time.)
- Choice: Choose a learning method!
 - Maybe use canopies within a learning method!

Conclusions

- Automatic Blocking Schemes using Machine Learning (Bilenko, et. al. & BSL)
 - 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