DRAFT: Improvements in Adaptive Blocking

Forest Gregg*

April 16, 2022

Record linkage paradigms based on pairwise comparisons scales like $O(n^2)$. In the worst case, every record has to be compared with every other record, leading to $\frac{n(n-1)}{2}$ comparisons, which is quickly impractical for moderately sized n. Therefore, linking large datasets depends on co-referent records being empirically rare and learning a blocking strategy that can take advantage of that sparseness.

Since we cannot control the distribution of truly co-referent records, our only course to scaling up classic record linkage is to get better at learning blocking strategies.

In this paper we introduce a block learning approach that produces blocking that significantly reduces the number of overall comparisons while still covering true co-referent pairs, compared to current methods.

The Block Learning Problem

The block learning problem is the task of finding a set of blocking predicates that produce blocks that group together, or cover, all co-referent records while minimizing the total number comparisons.

Let $\mathcal{B}$ be the set of co-referent pairs of records and $\mathcal{T}$ be the set of all pairs. Let $\mathcal{P}$ be the set of blocking predicate. Let $C(p, \mathcal{X})$ be all the pairs in $\mathcal{X}$ covered by predicate p. Let $\mathcal{P}^*$ be a subset of $\mathcal{P}$. Then our objective is:

$$\underset{\mathcal{P}^* : \bigcup_{p \in \mathcal{P}^*} C(p,\mathcal{B}) = \mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p \in \mathcal{P}^*} C(p, \mathcal{T}) \right| \tag{1}$$

This problem is equivalent to the Red-Blue Set Cover problem (Appendix A), and is NP-HARD.[Carr et al., 2000] Thus, we will usually not be able to solve the problem but only find a solution that approximates the optimal set.

*Thanks to Stefano Allesina and Jacopo Grilli for providing a derivation of the grid sampling estimator, and to Tarak Shah and Patrick Ball for many stimulating conversation on block learning methods.

Improving Block Learning

We can improve block learning by two means. First, we can improve our approximation of the optimal $\mathcal{P}^*$ given a set of candidate predicates. Second, we can add accurate and precise predicates to the candidate set $\mathcal{P}$. Our approach includes both types of improvements.

Branch and Bound

First, we introduce our algorithm for approximating the optimal predicate set $\mathcal{P}^*$: a branch and bound algorithm for the block learning problem (Algorithm 1).[1] The algorithm operates as follows.

Outline of Algorithm

The algorithm is provided with training data in the form of a set of co-referent record pairs, $\mathcal{B}$ and a sample of all record pairs $\mathcal{S}_\mathcal{T}$. We use a recursive algorithm `Search` to explore combinations of predicates to find a combination that covers $\mathcal{B}$ while covering a minimum of total comparisons.

In the first call of `Search`, we start with an empty partial solution $\mathcal{Q}$. At the branching stage, we find a predicate p amongst the candidate set $\mathcal{P}$ that maximizes a heuristic condition: the predicate that has the best ratio of coverage of still uncovered co-referent pairs to total estimated comparisons (Line 13). We remove this predicate from the candidate set $\mathcal{P}$ and explore a branch where p is added to $\mathcal{Q}$ (Line 14) and one where it is not (Line 15).

The current best solution set $\mathcal{P}^*$ is initialized to be the full set of candidate predicates. If a partial solution set $\mathcal{Q}$ covers enough co-referent pairs and has fewer total comparisons than the best solution set becomes $\mathcal{Q}$ (Lines 6-8)

The key bounding step comes from observing that we need only consider adding a predicate to our partial solution $\mathcal{Q}$ if adding that predicate will not make the total comparisons in the partial solution larger than our current best solution (Lines 10-11). After discarding predicates, we check if the remaining predicates combined with the candidate set could cover enough co-referent pairs. If not, we do not explore that branch futher (Line 12).

Note that we use a sample $\mathcal{S}_\mathcal{T}$ instead of all possible comparisons $\mathcal{T}$ because using the full set is usually not computationally feasible and a sample can be an effective proxy (Appendix B).

Refinements

There are two more opportunities for pruning the search space, after the next predicate p^* has been chosen (Line 13).

For the first branch, if we add the predicate p^* to the partial solution set $\mathcal{Q}$, then we need not consider adding any other predicate unless it can add at least one novel co-referent record pair to the coverage of $\mathcal{Q} \cup \{p^*\}$.

[1] See the DEDUPE library for a Python implementation of this algorithm.[Gregg et al., 2021]

Algorithm 1: BranchAndBoundSetCover

Input: Training set $\mathcal{B}$ and $\mathcal{S}_{\mathcal{T}}$ where each $b \in \mathcal{B}$ is a pair of co-referent records and $\mathcal{S}_{\mathcal{T}}$ is a sample of all possible record pairs.
Set of blocking predicates $\mathcal{P}$.
Maximum number of calls of the `Search` routine, L.

Output: Set of blocking predicates $\mathcal{P}^* \subseteq \mathcal{P}$

$\mathcal{P}^* \leftarrow \mathcal{P}$;
$l \leftarrow 0$;
Function `Search`($\mathcal{P}$, $\mathcal{Q}$) **is**
    if $l < L$ **then**
        $l \leftarrow l + 1$;
        if $\bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{B}) = \mathcal{B}$ **then**
            if $\left|\bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{S}_{\mathcal{T}})\right| < \left|\bigcup_{p^* \in \mathcal{P}^*} \mathrm{C}(p^*, \mathcal{S}_{\mathcal{T}})\right|$ **then**
                $\mathcal{P}^* \leftarrow \mathcal{Q}$
        else
            $\delta \leftarrow \left|\bigcup_{p^* \in \mathcal{P}^*} \mathrm{C}(p^*, \mathcal{S}_{\mathcal{T}})\right| - \left|\bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{S}_{\mathcal{T}})\right|$;
            $\mathcal{P} \leftarrow \left\{p \in \mathcal{P} : \left|\mathrm{C}(p, \mathcal{S}_{\mathcal{T}}) - \bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{S}_{\mathcal{T}})\right| < \delta\right\}$;
            if $\bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{B}) \cup \bigcup_{p \in \mathcal{P}} \mathrm{C}(p, \mathcal{B}) = \mathcal{B}$ **then**
                $p^* \leftarrow \mathrm{argmax}_{p \in \mathcal{P}} \frac{\left|\mathrm{C}(p, \mathcal{B}) - \bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{B})\right|}{\left|\mathrm{C}(p, \mathcal{S}_{\mathcal{T}}) - \bigcup_{q \in \mathcal{Q}} \mathrm{C}(q, \mathcal{S}_{\mathcal{T}})\right|}$;
                `Search`($\mathcal{P} - \{p^*\}$, $\mathcal{Q} \cup \{p^*\}$);
                `Search`($\mathcal{P} - \{p^*\}$, $\mathcal{Q}$);
`Search`($\mathcal{P}$, $\{\}$);
return $\mathcal{P}^*$

For the second branch, if we do not add p^* to $\mathcal{Q}$, then we can discard all predicates that are strictly worse than p^*. Specifically, we do not need to consider any p in $\mathcal{P}$ such that $\mathrm{C}(p, \mathcal{B} - \bigcup_{q\in\mathcal{Q}} \mathrm{C}(q, \mathcal{B})) \subseteq \mathrm{C}(p^*, \mathcal{B} - \bigcup_{q\in\mathcal{Q}} \mathrm{C}(q, \mathcal{B}))$ and $\mathrm{C}(p, \mathcal{S}_\mathcal{T} - \bigcup_{q\in\mathcal{Q}} \mathrm{C}(q, \mathcal{S}_\mathcal{T})) \supseteq \mathrm{C}(p^*, \mathcal{S}_\mathcal{T} - \bigcup_{q\in\mathcal{Q}} \mathrm{C}(q, \mathcal{S}_\mathcal{T}))$.

Approximation Bounds

This branch and bound algorithm is an extension of block learning algorithm proposed by Bilenko. Until a first solution is found, the algorithm will follow the first branch and repeatedly add the heuristic-proposed predicate to the partial solution set. This is the same as the greedy procedure at the heart of the algorithm proposed by Bilenko.[Bilenko et al., 2006].

As such, the branch and bound algorithm has the same approximation ratio as Bilenko's algorithm.[Peleg, 2000] The ratio between the number of comparisons produced by the branch and bound solution and the comparison produced by the optimal solution is bounded by the maximum number of individual predicates that cover any sample record pair multiplied by the number of provided co-referent pairs:

$$\max \left\{ \sum_{p\in\mathcal{P}} |\mathrm{C}(p, s)| : s \in \mathcal{S}_\mathcal{T} \right\} \cdot |\mathcal{B}| \tag{2}$$

Conjunctive Predicates

In the previous sections, we discussed improvements to the problem of finding the best subset of blocking predicates from a candidate set. Now, we turn to enriching the candidate set.

Given a set of atomic blocking rules, we can almost always generate better predicates by forming the conjunctions of these atomic rules. However, the combinatorial explosion of unique conjunctions quickly makes exploring even a small fraction of all possible conjunctions infeasible.

We propose an approach inspired by Govind's observation that we can view a set of blocking predicates as a random forest.[Govind et al., 2018] Taking this observations seriously, we use the typical random forest scheme to generate k-conjunctions (Algorithm 2).

We will use the depth of the trees of the forest, K, as a hyperparameter that will regularize our candidate set and so avoid overfitting the training date

Experiments

We compare the performance of our approach to block learning approaches proposed by Bilenko and by Michelson and Knoblock.[Bilenko et al., 2006, Michelson and Knoblock, 2006]

We will evaluate the different approaches on the `restaurant` and `cora` test datasets[Primpeli and Bizer, 2020, McCallum, 2017]. For the `restaurant` dataset, which has only 112 true co-referent pairs, we will use a random sample

Algorithm 2: BlockForest

Input: Training set $\mathcal{B}$ where $b \in \mathcal{B}$ is a pair of co-referent records.
Set of simple blocking predicates $\mathcal{P}$.
Number of iterations N.

Output: Set of blocking predicates $\mathcal{P}^{(c)}$

$\mathcal{P}^{(c)} \leftarrow \{\}$;
for $i \leftarrow 1$ **to** N **do**
  $\mathcal{B}' \leftarrow$ a random sample, with replacement, from $\mathcal{B}$ of size $|\mathcal{B}|$.;
  $\mathcal{P}' \leftarrow$ a random sample, without replacement, from $\mathcal{P}$ of size m.;
  `// Initialize` $p^{(c)}$ `as a predicate that always covers all pairs.`
  $p^{(c)} \leftarrow p_{\varnothing}$;
  for $k \leftarrow 1$ **to** K **do**
    $p^* \leftarrow \text{argmax}_{\mathcal{P}'} \frac{\left|\text{C}(p^{(c)} \wedge p, \mathcal{B}')\right|}{\left|\text{C}(p^{(c)} \wedge p, \mathcal{S}_{\mathcal{T}})\right|}$;
    $p^{(c)} \leftarrow p^{(c)} \wedge p^*$;
    $\mathcal{P}^{(c)} \leftarrow \mathcal{P}^{(c)} \cup \{p^{(c)}\}$;
return $\mathcal{P}^{(c)}$

of 20, 50 and 100 true co-referent pairs for training.[2]. For the `cora` dataset, we will use samples of 20, 50, 100, and 1000 true co-referent pairs. For these small datasets, we provide the full set of all possible comparisons $\mathcal{T}$ for $\mathcal{S}_{\mathcal{T}}$. Each configuration will replicated 100 times. We will evaluate our approach with a `BlockForest` tree depth of 1 and 2.

Our metrics will be recall and precision. We use the precision of blocking rules, $\frac{\text{covered pairs that are co-referent}}{\text{covered pairs}}$, instead of the reduction ratio, $1 - \frac{\text{covered pairs}}{\text{all possible pairs}}$, because precision is not dependent on the size of the dataset and is a more sensitive measure of performance.

Conclusion

Our `BranchAndBoundSetCover + BlockForest` approach dominates the other approaches in precision and, with a tree depth of 1, our approach provides similar recall.

Our approach does not continue to improve recall when we provide more than 100 co-referent pairs as training data. This may be because with more training data, the bootstrap sampling we use to generate the trees in the random forest do not have adequate sample variance to to generate sufficently diverse trees.

The block learning approach described in this paper substantially outperforms existing methods in precision, while maintaining good recall. Further

[2]The Bilenko approaches also uses distinct pairs for training and we will supply an equal number of distinct pairs

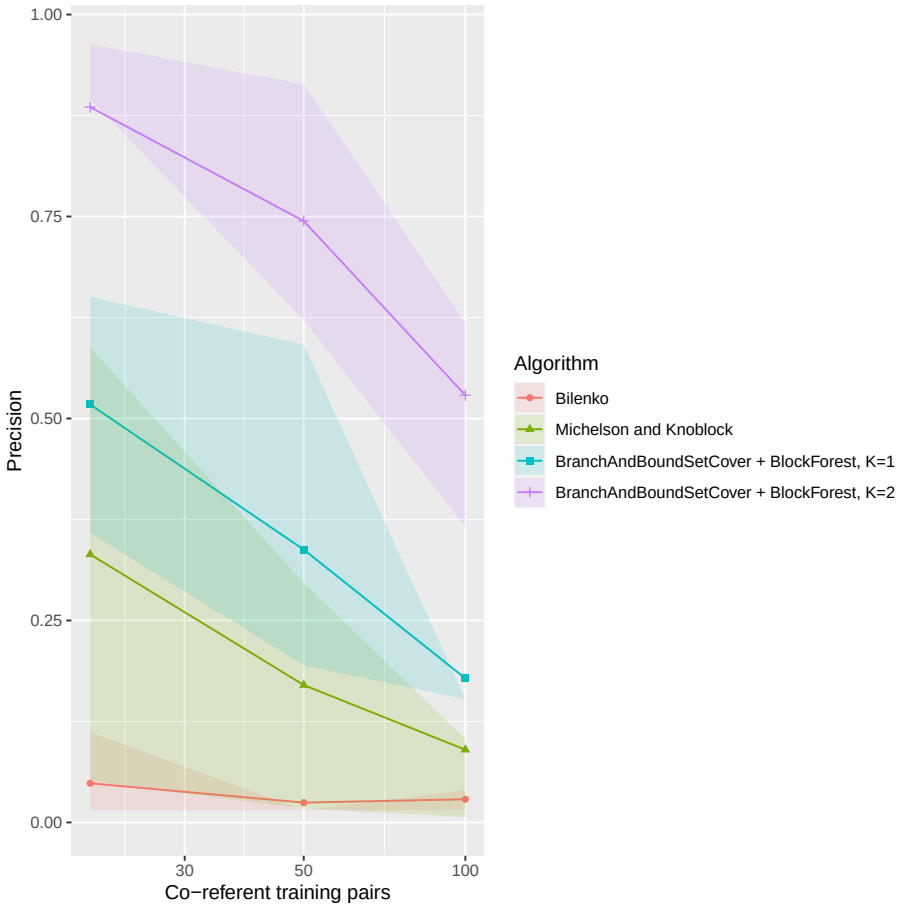

Figure 1: Mean Precision with Interquartile Range, `restaurant` dataset

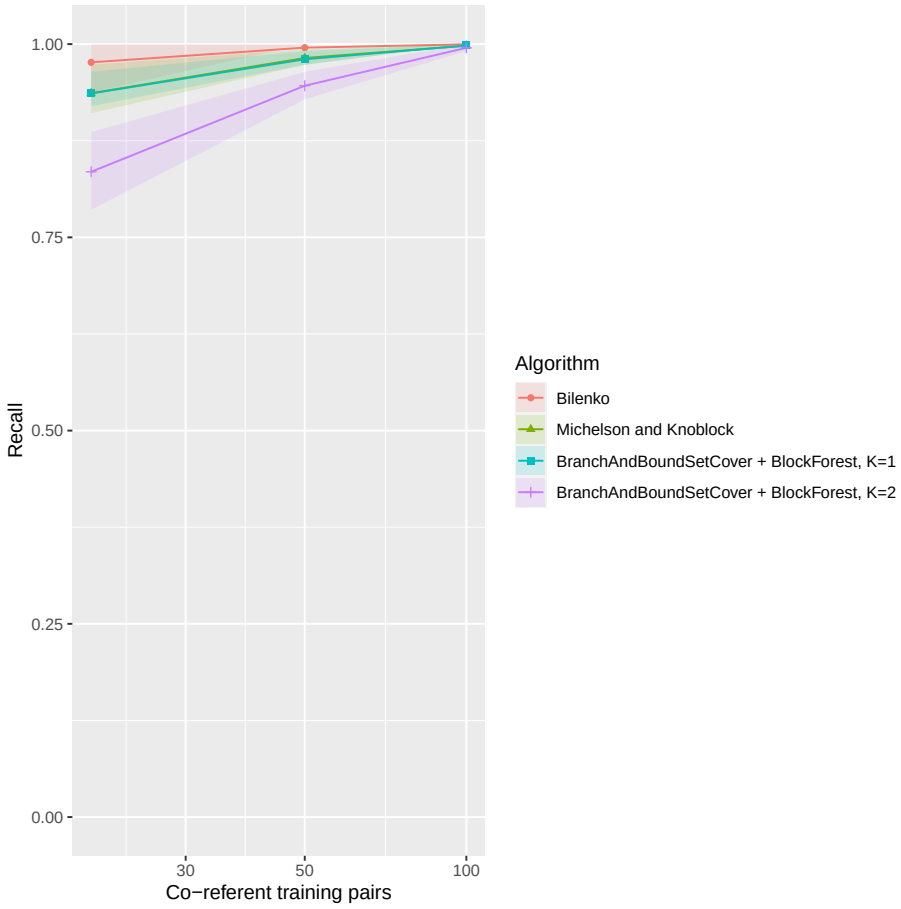

Figure 2: Mean Recall with Interquartile Range, `restaurant` dataset

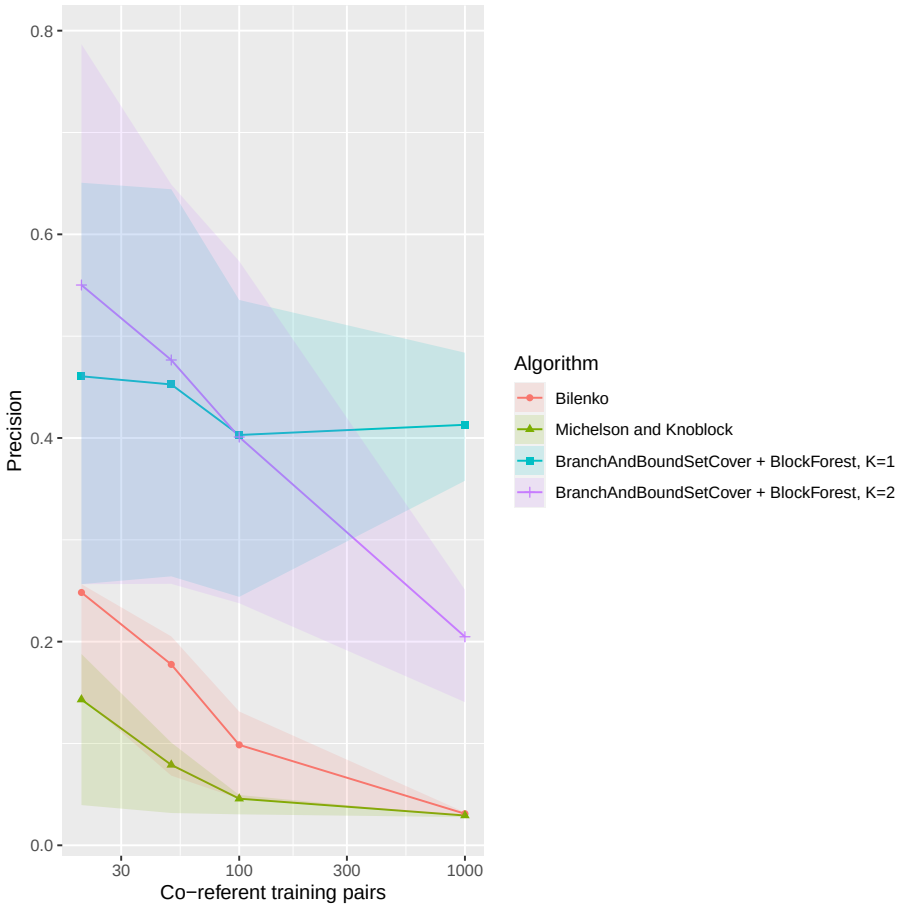

Figure 3: Mean Precision with Interquartile Range, `cora` dataset

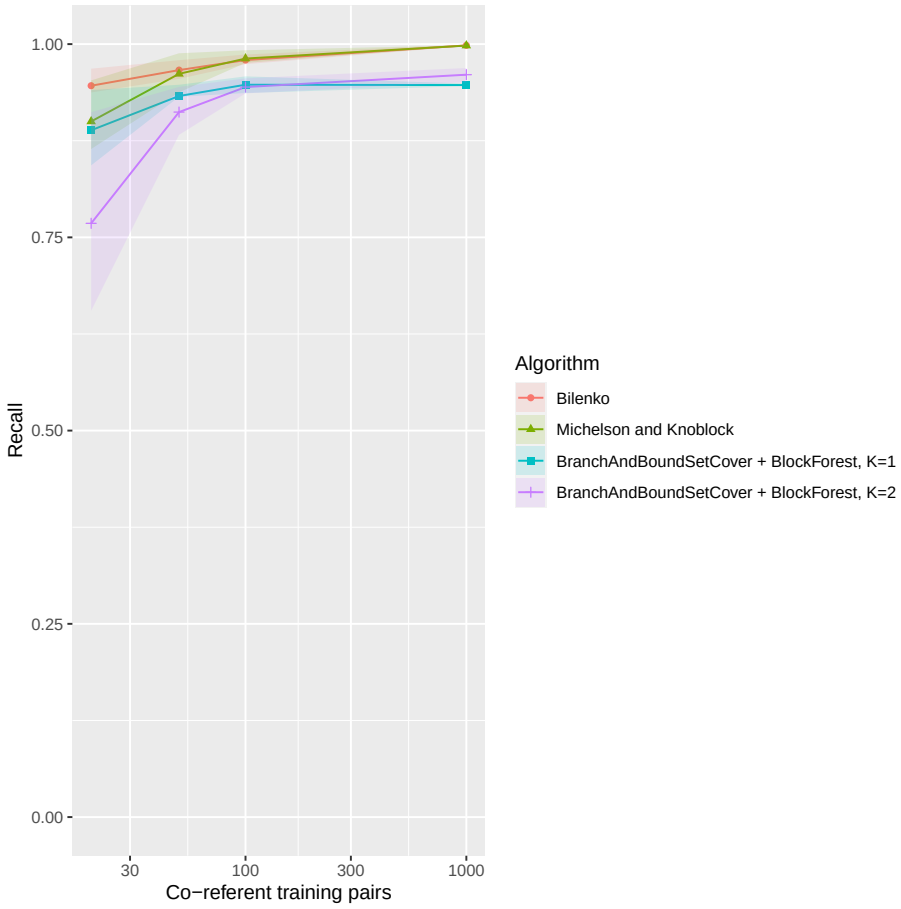

Figure 4: Mean Recall with Interquartile Range, `cora` dataset

improving recall, particularly when more training data is supplied would be desirable and should be possible.

References

Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the red-blue set cover problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, page 345–353, USA, 2000. Society for Industrial and Applied Mathematics. ISBN 0898714532.

Forest Gregg, Derek Eder, and Contributors. Dedupe (version 2.0.8), 2021. URL `https://github.com/dedupeio/dedupe`.

Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the Sixth IEEE International Conference on Data Mining*, pages 87–96, December 2006. URL `https://www.cs.utexas.edu/~ml/papers/blocking-icdm-06.pdf`.

David Peleg. Approximation algorithms for the label-cover max and red-blue set cover problems. In *Scandinavian Workshop on Algorithm Theory*, pages 220–231. Springer, 2000. URL `ftp://nozdr.ru/biblio/kolxoz/Cs/CsLn/AlgorithmTheory-SWAT2000,7conf.(LNCS1851,Springer,2000)(ISBN3540676902)(577s).pdf#page=231`.

Yash Govind, Erik Paulson, Palaniappan Nagarajan, Paul Suganthan G. C., AnHai Doan, Youngchoon Park, Glenn M. Fung, Devin Conathan, Marshall Carter, and Mingju Sun. Cloudmatcher: A hands-off cloud/crowd service for entity matching. *Proc. VLDB Endow.*, 11(12):2042–2045, August 2018. ISSN 2150-8097. doi: 10.14778/3229863.3236255. URL `https://doi.org/10.14778/3229863.3236255`.

Matthew Michelson and Craig A Knoblock. Learning blocking schemes for record linkage. In *AAAI*, volume 6, pages 440–445, 2006. URL `https://www.aaai.org/Papers/AAAI/2006/AAAI06-070.pdf`.

Anna Primpeli and Christian Bizer. Restaurants (Fodors-Zagats), Augmented Version, Fixed Splits, 2020.

Andrew McCallum. Cora Dataset, 2017. URL `https://doi.org/10.18738/T8/HUIG48`.

Wikipedia contributors. Birthday problem — Wikipedia, the free encyclopedia, 2021. URL `https://en.wikipedia.org/w/index.php?title=Birthday_problem&oldid=1027170129`. [Online; accessed 29-June-2021].

Jacopo Grilli and Stefano Allesina. Last name analysis of mobility, gender imbalance, and nepotism across academic systems. *Proceedings of the National Academy of Sciences*, 114(29):7600–7605, 2017. doi: 10.1073/pnas.1703513114. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1703513114`.

A Equivalence of Red-Blue Set Cover and Total Cover Problem

The Red-Blue cover problem is equivalent to the problem of minimizing total comparisons (Equations 3-7), and Peleg's original analysis of the bounds can be translated to minimizing total cover as the analysis does not depend on $\mathcal{R}$ and $\mathcal{B}$ being disjoint.[Peleg, 2000]

$$\underset{\mathcal{P}^*:\bigcup_{p\in\mathcal{P}^*}\mathrm{C}(p,\mathcal{B})=\mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p\in\mathcal{P}^*} \mathrm{C}(p,\mathcal{R}) \right| \tag{3}$$

$$\underset{\mathcal{P}^*:\bigcup_{p\in\mathcal{P}^*}\mathrm{C}(p,\mathcal{B})=\mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p\in\mathcal{P}^*} (\mathrm{C}(p,\mathcal{T}) - \mathrm{C}(p,\mathcal{B})) \right| \tag{4}$$

$$\underset{\mathcal{P}^*:\bigcup_{p\in\mathcal{P}^*}\mathrm{C}(p,\mathcal{B})=\mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p\in\mathcal{P}^*} \mathrm{C}(p,\mathcal{T}) - \bigcup_{p\in\mathcal{P}^*} \mathrm{C}(p,\mathcal{B}) \right| \tag{5}$$

$$\underset{\mathcal{P}^*:\bigcup_{p\in\mathcal{P}^*}\mathrm{C}(p,\mathcal{B})=\mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p\in\mathcal{P}^*} \mathrm{C}(p,\mathcal{T}) \right| - |\mathcal{B}| \tag{6}$$

which has the same solution as

$$\underset{\mathcal{P}^*:\bigcup_{p\in\mathcal{P}^*}\mathrm{C}(p,\mathcal{B})=\mathcal{B}}{\operatorname{argmin}} \left| \bigcup_{p\in\mathcal{P}^*} \mathrm{C}(p,\mathcal{T}) \right| \tag{7}$$

$$\tag{8}$$

B Sampling

In practice, it will often not be feasible to work directly with the set of all possible comparisons $\mathcal{T}$. In this appendix, we show that we can instead profitably work with a random sample of record pairs, $\mathcal{S}_\mathcal{T}$ from $\mathcal{T}$.

In order for an algorithm to find an approximate solution to minimizing total cover, the number of sampled comparisons covered by a predicate needs to be approximately proportional to the total comparisons covered by the predicate.

$$K\,|\mathrm{C}(p,\mathcal{S}_\mathcal{T})| \approx |\mathrm{C}(p,\mathcal{T})| \text{ for all } p \in \mathcal{P} \tag{9}$$

If $\mathcal{S}'_\mathcal{T}$ is a simple random sample, without replacement, from $\mathcal{T}$, then

$$\frac{|\mathcal{T}|}{|\mathcal{S}'_\mathcal{T}|}\,|\mathrm{C}(p,\mathcal{S}'_\mathcal{T})| \approx |\mathrm{C}(p,\mathcal{T})| \text{ for all } p \in \mathcal{P} \tag{10}$$

with a standard error of

$$\sqrt{\frac{|\mathrm{C}(p,\mathcal{T})|\left(|\mathcal{T}|-|\mathrm{C}(p,\mathcal{T})|\right)}{|\mathcal{S}'_{\mathcal{T}}|}\left(1-\frac{|\mathcal{S}'_{\mathcal{T}}|-1}{|\mathcal{T}|-1}\right)} \tag{11}$$

Generally, better predicates for efficient blocking will be the predicates that cover a very small proportion of $\mathcal{T}$. For these predicates, that rarity means that $|\mathcal{S}'_{\mathcal{T}}|$ has to be very large to get a precise enough estimate to use $\mathcal{S}'_{\mathcal{T}}$ in place of $\mathcal{T}$. Further, in order to calculate $\mathrm{C}(p,\mathcal{S}'_{\mathcal{T}})$, we have apply the predicate p to up to $2\,|\mathcal{S}'_{\mathcal{T}}|$ records.[3] Thus, simple random sampling is generally not computationally convenient.

Instead, we use a form of grid sampling to form $\mathcal{S}_{\mathcal{T}}$. Let $\mathcal{D}$ be the set of all records. Let $\mathcal{S}_{\mathcal{D}}$ be a simple random sample, without replacement, from $\mathcal{D}$. Let $\mathcal{S}_{\mathcal{T}}$ be the set of all unique pairs of records in $\mathcal{S}_{\mathcal{D}}$. The resultant estimator also converges.[Grilli and Allesina, 2017]

$$\frac{|\mathcal{T}|}{|\mathcal{S}_{\mathcal{T}}|}\,|\mathrm{C}(p,\mathcal{S}_{\mathcal{T}})| \approx |\mathrm{C}(p,\mathcal{T})| \text{ for all } p \in \mathcal{P} \tag{13}$$

This approach dramatically reduces the number of records that need to be blocked to calculate $\mathrm{C}(p,\mathcal{S}_{\mathcal{T}})$ compare to $\mathrm{C}(p,\mathcal{S}'_{\mathcal{T}})$. For example, let's say our dataset $\mathcal{D}$ has 20,000 records. If we block 100 records, we will form a sample $\mathcal{S}_{\mathcal{T}}$ of 4,950 pairs. If we randomly sample 4,950 pairs from a $\mathcal{T}$, we will expect to have to block 7,808 records.

Unfortunately, we do not have a formula for the standard error, but simulation studies show good convergence properties (Figure 5).

[3] How many records will really need to be blocked is a variation of the Birthday Problem[Wikipedia contributors, 2021]. Let $\mathcal{D}$ be the set of unique records, then the expected number of records covered by $\mathcal{S}'_{\mathcal{T}}$ is bounded below by

$$|\mathcal{D}| - |\mathcal{D}|\left(\frac{|\mathcal{D}|-2}{|\mathcal{D}|}\right)^{|\mathcal{S}'_{\mathcal{T}}|} \tag{12}$$

This is the formula for sampling with replacement. As $\mathcal{S}'_{\mathcal{T}}$ is sampled without replacement, the number of covered records will be a bit higher.

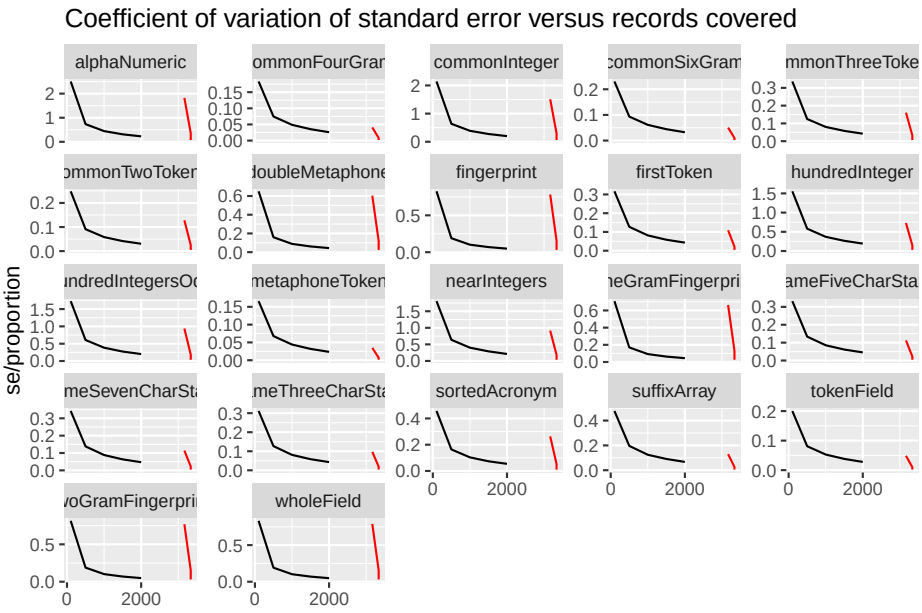

Figure 5: Coefficient of Variation of Standard error and Sample Size for Various Predicates. Dataset of 3,337 records. Black line: Grid Sample (10,000 runs for each sample size); Red line: Simple Random Sample