A General Approach to Supervised Meta-Blocking

(Discussion Paper)

Luca Gagliardelli[1,*], George Papadakis[2], Giovanni Simonini[1], Sonia Bergamaschi[1] and Themis Palpanas[3]

[1]*Università degli Studi di Modena e Reggio Emilia, Italy*
[2]*University of Athens, Greece*
[3]*Université Paris Cité & IUF, France*

Abstract
Entity Resolution is a core data integration task that relies on Blocking to scale to large datasets. Schema-agnostic blocking achieves very high recall, requires no domain knowledge, and works on data with any structure. The main drawback of this approach is the number of generated superfluous comparisons (i.e. non-matching), which can be reduced by Meta-blocking techniques that aim to discard most of them. Unsupervised Meta-blocking performs this process by scoring each comparison with a single metric and then applying a pruning algorithm, so choosing the right metric among the existing ones is fundamental to achieving good results. Supervised Meta-blocking improves this approach by combining multiple scores per comparison into a feature vector that is fed to a binary classifier used to decide if a comparison is a match or not. In this work, we generalize the Supervised Meta-blocking approach by using a probabilistic classifier that pairs each comparison with a score that represents its likelihood to be a match, allowing to use of any pruning algorithm.

Keywords
Entity Resolution, Data Integration, Data Cleaning, Big Data

1. Introduction

Entity Resolution (ER) is the task of identifying records (profiles) that pertain to the same real-world object (entity) among different data sources [1, 2, 3, 4, 5]. ER is a core data integration task since allows removing duplicates from dirty data sets that could compromise the downstream analysis, and joining multiple data sets when no explicit joining keys are available.

ER is a challenging task due to its quadratic time complexity: in the worst case, every profile has to be compared with all others, obtaining poor performance in terms of scalability. To mitigate this high complexity, *Blocking* is employed [6, 7, 8, 9, 10] to restrict ER to *blocks* of profiles that have similar signatures. When dealing with noisy web data that does not have a fixed structure, *schema-agnostic* signatures can be used effectively to perform blocking without requiring domain or schema knowledge [11, 6], with this approach parts of any attribute value

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy
*Corresponding author.
luca.gagliardelli@unimore.it (L. Gagliardelli); gpapadis@di.uoa.gr (G. Papadakis); simonini@unimore.it (G. Simonini); sonia.bergamaschi@unimore.it (S. Bergamaschi); themis@mi.parisdescartes.fr (T. Palpanas)
0000-0001-5977-1078 (L. Gagliardelli); 0000-0002-7298-9431 (G. Papadakis); 0000-0002-3466-509X (G. Simonini); 0000-0001-8087-6587 (S. Bergamaschi); 0000-0002-8031-0265 (T. Palpanas)

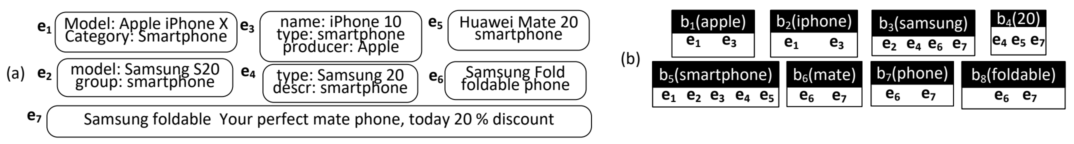

(a)
e1 Model: Apple iPhone X Category: Smartphone
e3 name: iPhone 10 type: smartphone producer: Apple
e5 Huawei Mate 20 smartphone
e2 model: Samsung S20 group: smartphone
e4 type: Samsung 20 descr: smartphone
e6 Samsung Fold foldable phone
e7 Samsung foldable Your perfect mate phone, today 20 % discount
(b)
b1(apple) e1 e3
b2(iphone) e1 e3
b3(samsung) e2 e4 e6 e7
b4(20) e4 e5 e7
b5(smartphone) e1 e2 e3 e4 e5
b6(mate) e6 e7
b7(phone) e6 e7
b8(foldable) e6 e7

Figure 1: (a) The input entities (smartphone models), and (b) the redundancy-positive blocks produced by Token Blocking.

in each profile can be used as signatures. An example of schema-agnostic blocking is reported in Figure 1. The data set in Figure 1a contains three duplicate pairs, $\langle e_1, e_3\rangle$, $\langle e_2, e_4\rangle$ and $\langle e_6, e_7\rangle$, that are clustered using Token Blocking (a block is created for every token appearing in at least 2 profiles). The resulting blocks appear in Figure 1b. ER is performed within each block, detecting all duplicates.

The main drawback of this approach is that the resulting blocks involve high levels of redundancy: every profile is associated with multiple blocks, yielding many *redundant* (comparisons repeated across different blocks) and *superfluous* (involving non-matching profiles) comparisons. For example, the pair $\langle e_1, e_3\rangle$ is redundant in b_2, as it is already examined in b_1, while the pair $\langle e_2, e_6\rangle \in b_3$ is superfluous, as the two entities are not duplicates. These comparisons can be removed from the block collection, reducing the computational cost of ER, while keeping the same result in terms of recall.

Meta-blocking [12] was introduced to reduce the portion of superfluous comparisons and remove all the redundant ones. To perform this task, it relies on two components: (*i*) a *weighting scheme*, which is a function that given a pair of profiles and their associated blocks, returns a score proportional to their matching likelihood; (*ii*) a *pruning algorithm*, which receives as input all weighted pairs and retains the ones that are more likely to be matching. Meta-blocking can be unsupervised or supervised.

Unsupervised Meta-blocking. Starting from a block collection (Figure 1b) it builds a blocking graph (Figure 2a) in which the nodes represent the profiles, and two nodes are connected by an edge if the corresponding profiles co-occur in at least one block. The edges are weighted by using a weighting scheme; in our example, the number of blocks shared by the connected profiles. Then, the blocking graph is pruned using a pruning algorithm; in our example, for each node, we discard the edges with a weight lower than the average of its edges.

Figure 2b shows the pruned blocking graph, the dashed lines represent the superfluous

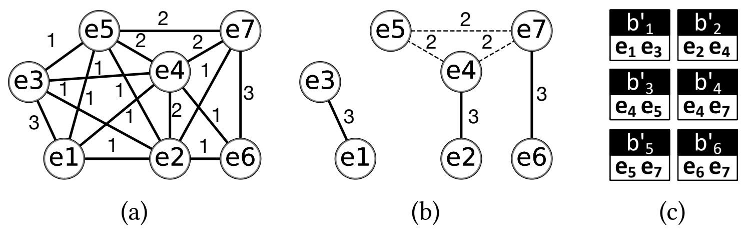


Figure 2: Unsupervised Meta-blocking example: (a) The blocking graph of the blocks in Figure 1b, using the number of common blocks as edge weights, (b) a possible pruned blocking graph, and (c) the new blocks.

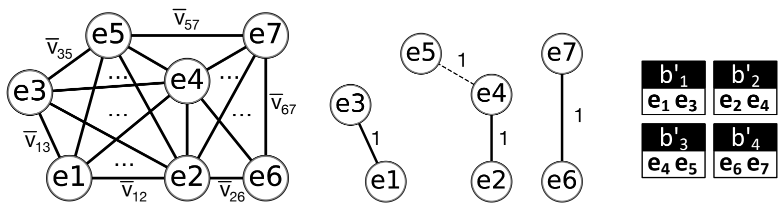


Figure 3: Supervised Meta-blocking example: (a) a graph where each edge is associated with a feature vector, (b) the graph pruned by a binary classifier, and (c) the output, which contains a new block per retained edge.

comparisons. Finally, a new block is created for every retained edge (Figure 2c), the new block collection involves significantly fewer pairs without missing the matching ones.

Supervised Meta-blocking [13]. It trains a binary classifier to learn to classify every comparison as positive (i.e. likely to be matching) or negative (i.e. unlikely to be matching). Then, the classifier is used to perform the pruning of the blocking graph, retaining only positive classified comparisons. Each pair is associated with a feature vector comprising the most distinctive weighting schemes that are used by unsupervised meta-blocking.

An example of Supervised Meta-Blocking is shown in Figure 3. A blocking graph is generated as for unsupervised meta-blocking (Figure 3a), pairing each edge with a feature vector. In this example, each pair of profiles $\langle e_i, e_j \rangle$ is represented by a feature vector $v_{i,j} = \{CB(e_i, e_j), JS(e_i, e_j)\}$, where $CB(e_i, e_j)$ is the number of their common blocks and $JS(e_i, e_j)$ is the Jaccard coefficient of blocks associated with e_i and e_j. Then, a binary classifier is trained with a sample of labeled vectors and is used to predict whether a pair $\langle e_i, e_j \rangle$ is a match ($l_{i,j}$=1) or not ($l_{i,j}$=0). The pairs classified as positive are retained, as shown in Figure 3b (the dashed line indicates the superfluous pair $\langle e_4, e5 \rangle$). The final result, which includes a new block per retained pair, appears in Figure 3c.

Supervised Meta-blocking requires to generate labeled data for training, but by representing each edge with multiple features it is able to produce more accurate results, obtaining better precision and recall than the unsupervised approach [13]. However, the binary classifier learns a *global* threshold to perform the pruning of the edges since it is applied on the whole blocking graph. Defining a *local* threshold for each node, as for unsupervised meta-blocking, would allow better control of the pruning. This is the intuition behind the Generalized Supervised

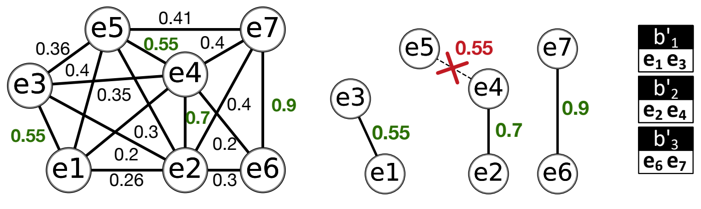


Figure 4: Generalized Supervised Meta-blocking example: (a) a graph weighted with a probabilistic classifier, (b) the pruned graph, and (c) the new blocks.

Meta-blocking approach.

Generalized Supervised Meta-blocking. Our new approach, first, builds a graph as Supervised Meta-blocking does (Figure 3a), then trains a probabilistic classifier, which assigns the matching probability to each edge (Figure 4a). Now, it is possible to apply several weight- and cardinality-based pruning algorithms on the resulting graph. In our example, we employed the Supervised WNP: for each node, first discards all the edges with a weight lower than 0.5, then retains only those with a weight greater than the average of the remaining ones. Figure 4b shows the result of this step: two edges may be assigned the same weight by the probabilistic classifier, e.g., $\langle e_1, e3\rangle$ and $\langle e_4, e5\rangle$, but they may be kept (e.g., the matching pair $\langle e_1, e3\rangle$) or discarded (e.g., the non-matching pair $\langle e_4, e5\rangle$) depending on their context, i.e., the weights in their neighborhood. Note that $\langle e4, e5\rangle$ is not discarded by Supervised Meta-blocking in Figure 3b, which thus underperforms Generalized Supervised Meta-blocking in terms of precision (for the same recall).

Our Contributions. This work is a resume of our previous article [14] published in PVLDB 2022 that aims to illustrate our major findings. In particular, we improved Supervised Meta-blocking by generalizing it from a binary classification task to a binary probabilistic classification process, and then using the resulting probabilities as comparison weights in order to use them with pruning algorithms that are incompatible with the original approach [13]. We introduced three new weighting schemes, and finally, we performed an extensive experimental study that involves 9 real-world datasets. The results demonstrate that the new pruning algorithms significantly outperform the existing ones. They also identify the top-performing algorithms and feature vectors, showing that 50 labeled instances (25 per class) suffice for high performance.

The rest of the paper is organized as follows: Section 2 provides background knowledge on the task of Supervised Meta-blocking and the problem we tackle. The experimental analysis is presented in Section 3, the main works in the field are discussed in Section 4, and the paper concludes in Section 5 along with directions for future work.

2. Preliminaries and problem definition

An entity profile p_i is defined as a set of name-value pairs, in which both the attribute names and the attribute values are textual. Two profiles p_i, p_j that pertain to the same real-world object are called *duplicates* or *matches*, we denote them as $p_i \equiv p_j$.

As in other state-of-the-art ER frameworks [6, 3, 15], we employ Blocking to reduce the quadratic time complexity of the ER. In particular, we employ Token Blocking, a redundant blocking technique in which a profile p_i can appear in multiple blocks. The performance of blocking can be assessed through *Recall* and *Precision*. The former measures the number of retrieved matches over the existing ones in the data set, while the latter the number of retrieved matches over the number of retrieved entity profiles pairs.

Unsupervised Meta-blocking [16] operates in top of Blocking to restructure a collection of blocks B, generated by a redundant blocking technique, relying on the intuition that the more blocks two profiles share, the more likely they match. Starting from a block collection B, the goal of meta-blocking is to produce a new block collection B' in which $Recall(B') \approx Recall(B)$ and $Precision(B') >> Precision(B)$. To achieve this goal, Meta-blocking builds a graph

$G_B = \{V_B, E_B, W_B\}$ in which: V_B is the set of nodes representing all profiles p_i, E_B is the set of edges; an edge $e_{i,j}$ between two entity profiles p_i, p_j exists if they co-occur in at least one block; W_B is the set of weights $w_{i,j}$ associated to the edges. Several weighting schemes can be used to weight the edges and capture the matching likelihood of the entity profiles that they connect. For example, one of the simplest weighting scheme is $CBS(e_{i,j}) = |B_i \cap B_j|$ that counts the number of shared blocks among two entity profiles p_i, p_j. A complete description of all the weighting schemes can be found in [14]. Finally, the graph is pruned using a pruning algorithm that removes less promising edges. A pruning algorithms can be *weight-based* or *cardinality-based*, the former retains edges that are weighted above a certain threshold, while the latter the top-k weighted ones. An algorithm can be applied *locally* (i.e. for each node of the graph) or *globally* (i.e. on all the edges of the graph). The combination of these strategies produces the following pruning algorithms: (i) Weighted Edge Pruning (WEP), prunes all the edges with a weight lower than a given threshold; (ii) Cardinality Edge Pruning (CEP), sorts the edges in descending order with respect to their weights, and then keeps only the first k; (iii) Weighted Node Pruning (WNP), considers in turn each node e_i and its connected edges, and prunes the edges with a weight lower than a calculated threshold; (iv) Cardinality Node Pruning (CNP), for each node p_i retains the top-k edges by using a cardinality threshold k_i. The node based algorithms can be also *reciprocal*, which means that an edge $e_{i,j}$ is retained only if it is retained by both connected profiles p_i, p_j. Thus, other two algorithms can be derived: *RCNP* and *RWNP*. There are also variant of these standard algorithms, like *BLAST* that instead of using the average weight per node to define a threshold, relies on the maximum weight per node [4]. After the pruning, each retained edge can be used to build a new block collection B'.

Supervised Meta-blocking [13] models every weight $w_{i,j}$ as a feature vector $f_{i,j} = [s_1(e_{i,j}), s_2(e_{i,j}), ..., s_n(e_{i,j})]$, where each s_i is a weighting scheme. Then, 5% of the feature vectors are labeled and used to train a *binary classifier* that is then used to labels the others as `matches` or `non-matches`. Finally, to prune the edges it employs a pruning algorithm called *BCl* that discards all the edges labelled as `non-matches`. In [13] Supervised Meta-blocking was adapted to work also with CEP and CNP.

The run-time RT of Supervised Meta-blocking is composed by the time of: generating the feature vectors for E_B, training the classification model M, applying M to E_B.

Problem definition. Generalized Supervised Meta-blocking differs from Supervised Meta-blocking in two ways: (i) instead of a *binary* classifier that assigns class labels, it trains a *probabilistic* classifier that assigns a weight $w_{i,j} \in [0, 1]$ to every edge $e_{i,j}$. This weight expresses how likely it is to belong to the positive class. (ii) The candidate pairs with a probability lower than 0.5 are discarded, but the rest, called *valid pairs*, are further processed by a pruning algorithm as for unsupervised meta-blocking. The ones retained after pruning produce the new block collection B'.

The run-time of Generalized Supervised Meta-blocking, RT, adds to that of Supervised Meta-blocking the time required to process the assigned probabilities by a pruning algorithm.

Table 1
Characteristics of the datasets used in the experimental study. $|E_x|$ number of entities, $|D|$ number of duplicate pairs, and $|C|$ number of distinct candidate pairs. The rightmost part reports the performance of the original blocks that are given as input to the supervised meta-blocking methods.

Dataset	$\|\mathbf{E_1}\|$	$\|\mathbf{E_2}\|$	$\|\mathbf{D}\|$	$\|\mathbf{C}\|$	Recall	Precision	F1
AbtBuy	1.1k	1.1k	1.1k	36.7k	0.948	$2.78{\cdot}10^{-2}$	$5.40{\cdot}10^{-2}$
DblpAcm	2.6k	2.3k	2.2k	46.2k	0.999	$4.81{\cdot}10^{-2}$	$9.18{\cdot}10^{-2}$
ScholarDblp	2.5k	61.3k	2.3k	832.7k	0.998	$2.80{\cdot}10^{-3}$	$5.58{\cdot}10^{-3}$
AmazonGP	1.4k	3.3k	1.3k	84.4k	0.840	$1.29{\cdot}10^{-2}$	$2.54{\cdot}10^{-2}$
ImdbTmdb	5.1k	6.0k	1.9k	109.4k	0.988	$1.78{\cdot}10^{-2}$	$3.50{\cdot}10^{-2}$
ImdbTvdb	5.1k	7.8k	1.1k	119.1k	0.985	$8.90{\cdot}10^{-3}$	$1.76{\cdot}10^{-2}$
TmdbTvdb	6.0k	7.8k	1.1k	198.6k	0.989	$5.50{\cdot}10^{-3}$	$1.09{\cdot}10^{-2}$
Movies	27.6k	23.1k	22.8k	26.0M	0.976	$8.59{\cdot}10^{-4}$	$1.72{\cdot}10^{-3}$
WalmartAmazon	2.5k	22.1k	1.1k	27.4M	1.000	$4.22{\cdot}10^{-5}$	$8.44{\cdot}10^{-5}$

3. Experimental evaluation

3.1. Experimental setup

Hardware and Software—All the experiments were performed on a machine equipped with four Intel Xeon E5-2697 2.40 GHz (72 cores), 216 GB of RAM, running Ubuntu 18.04. We integrated Generalized Supervised Meta-blocking in the *SparkER* library, code and usage examples are available on the GitHub page of the project[1]. Unless stated otherwise, we perform machine learning analysis using Python 3.7 and the Support Vector Classification (SVC) model of scikit-learn [17]. We used the default configuration parameters, enabling the generation of probabilities and fixing the random state so as to reproduce the probabilities over several runs. We performed all experiments with logistic regression, too, obtaining almost identical results, but we omit them for brevity.

Datasets—Table 1 lists the 9 real-world datasets employed in our experiments. They have different characteristics and cover a variety of domains. Each dataset involves two different, but overlapping data sources, where the ground truth of the real matches is known. `AbtBuy` matches products extracted from Abt.com and Buy.com [18]. `DblpAcm` matches scientific articles extracted from dblp.org and dl.acm.org [18]. `ScholarDblp` matches scientific articles extracted from scholar.google.com and dblp.org [18]. `ImdbTmdb`, `ImdbTvdb` and `TmdbTvdb` match movies and TV series extracted from IMDB, TheMovieDB and TheTVDB [19], as suggested by their names. `Movies` matches information about films that are extracted from imdb.com and dbpedia.org [11]. `WMAmazon` matches products from Walmart.com and Amazon.com [20].

Blocking—The initial block collection is extracted through Token Blocking [7]. The original blocks are then processed by Block Purging [11], which discards all the blocks that correspond to highly frequent tokens (e.g., stop-words). Finally, we apply Block Filtering [16], removing each profile p_i from the largest 20% blocks in which it appears. The performance of the resulting block collections is reported in the rightmost part of Table 1. To apply Generalized Supervised Meta-blocking to these block collections, we performed 10 runs and averaged the values of precision, recall, and F1. In each run, a different seed is used to sample the pairs that compose the training set.

[1]https://github.com/Gaglia88/sparker

3.2. Comparison with Supervised Meta-blocking

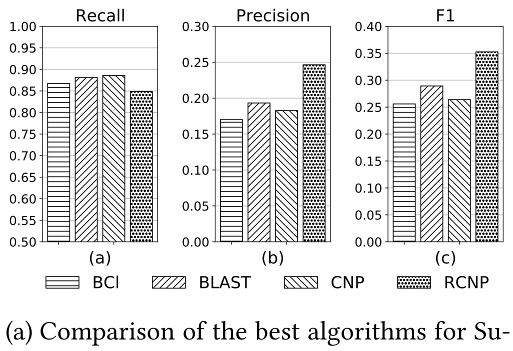
The whole experimental evaluation is available in our paper [14], we report here for sake of space only the comparison with Supervised Meta-blocking. To compare Generalized Supervised Meta-blocking with Supervised Meta-blocking[13] we follow these steps by using a balanced training set composed of 500 samples: first, we selected the best weight and cardinality based pruning algorithms for Generalized Supervised Meta-blocking by using the feature vector proposed in [13], obtaining as result BLAST and RCNP respectively. Then, for both pruning algorithms we selected the best feature combination by using a brute force approach, so trying all possible weighting scheme combinations. Finally, we compared BLAST and RCNP in combination with the best features selected with BCl and CNP (the original Supervised Meta-blocking algorithms), which use the feature set proposed in [13].

The average performance is presented in Figure 5a. Between the weight-based algorithms, we observe that BLAST outperforms BCl with respect to all measures: its recall, precision and F1 are higher by 1.6%, 13.6% and 13%, respectively, on average. Thus, *BLAST is much more accurate in the classification of the candidate pairs and more suitable than BCl for recall-intensive applications.* While, among the cardinality-based algorithms, RCNP trades slightly lower recall than CNP for significantly higher precision and F1: on average, across all datasets, its recall is lower by -4.1%, while its precision and F1 are higher by 34.9% and by 33.6%, respectively. As a result, *RCNP is more suitable than CNP for precision-intensive applications.*

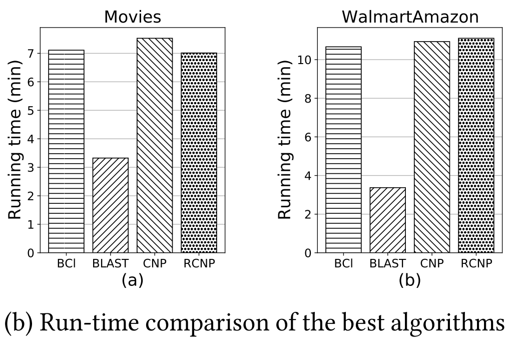
Regarding time efficiency, Figure 5b reports the running times of these algorithms on the largest datasets, i.e., `Movies` and `WalmartAmazon`. We observe that BCl, CNP and RCNP exhibit similar RT in both cases, since they all employ more complex feature sets. BLAST is substantially faster than these algorithms, reducing RT by more than 50%. In particular, comparing it with its weight-based competitor, we observe that BLAST is faster than BCl by 2.1 times over `Movies` and by 3.2 times over `WalmartAmazon`.

We also demonstrated that Generalized Supervised Meta-blocking can obtain better results than Supervised Meta-blocking by using a balanced training set composed only of 50 labeled samples as shown in Figure 6. We observe that BLAST with a balanced training set composed of 50 pairs outperforms BCl with respect to all measures: its recall, precision and F1 are higher by 7%, 5% and 9.9%, respectively, on average. The same behaviour is shown by RCNP over CNP, the recall is improved by 9.2%, the precision by 16.4% and the F1 by 18.3%.

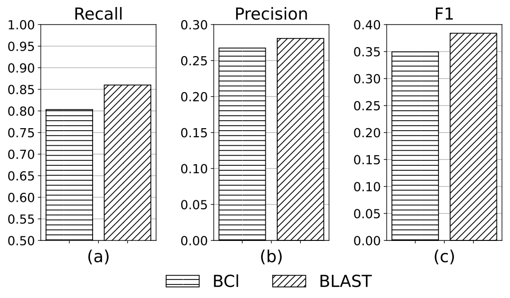
We can conclude that Generalized Supervised Meta-blocking bring significant improvements



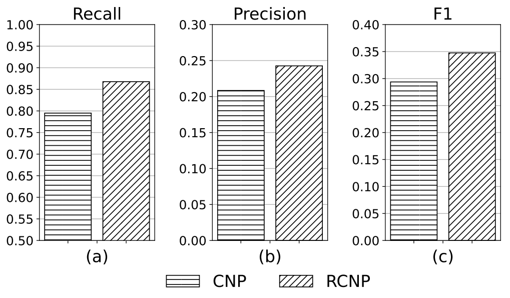
(a) Comparison of the best algorithms for Supervised (BCl, CNP) and Generalized Supervised Meta-blocking (BLAST, RCNP).



(b) Run-time comparison of the best algorithms for Supervised (BCl, CNP) and Generalized Supervised Meta-blocking (BLAST, RCNP).



(a) Weight-based



(b) Cardinality-based

Figure 6: Comparison of Supervised meta-blocking (BCl, CNP)[13] with the original features and 5% of comparisons as training set and Generalized Supervised Meta-blocking (BLAST, RCNP) with new features and 50 labeled samples as training.

with respect to Supervised Meta-blocking.

4. Related work

The unsupervised pruning algorithms WEP, WNP, CEP, and CNP were introduced in [12]. WNP and CNP were improved in [16] to avoid the generation of redundant comparisons. Unsupervised RWNP and RCNP were defined in [16], while unsupervised BLAST was proposed in [21].

The most similar work to this one is BLOSS [22] which introduces an active learning method that reduces the size of the labeled comparisons needed by Supervised Meta-blocking. BLOSS divides the unlabelled candidate pairs into groups based on CF-IBF weighting schema. Then, it applies a rule-based active sampling inside each group to select the pairs to label with the lowest commonalities with the already labeled ones to maximize the captured information. In the final step, BLOSS cleans the labeled sample from non-matching outliers with high Jaccard weight.

5. Conclusions

We have presented Generalized Supervised Meta-blocking, which casts Meta-blocking as a probabilistic binary classification task and weights all candidate pairs in a block collection with the probabilities produced by the trained classifier. These weights are processed by a pruning algorithm that can be weight-based (promoting recall) or cardinality-based (promoting precision). Through a thorough experimental study over 9 established, real-world datasets, we verified that BLAST and RCNP constitute the best weight- and cardinality-based pruning algorithms, respectively. We determined the best feature set for these algorithms and we demonstrated they are able to obtain good results with a balanced training set composed of 50 samples.

In the future, we plan to leverage Generalized Supervised Meta-blocking as a means for optimizing the performance of Progressive Entity Resolution [23].

Acknowledgments

The publication of this work was possible thanks to the FARD-2022.

References

[1] X. L. Dong, D. Srivastava, Big Data Integration, Morgan & Claypool Publishers, 2015.

[2] P. Christen, Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, Data-Centric Systems and Applications, Springer, 2012.

[3] G. Papadakis, E. Ioannou, E. Thanos, T. Palpanas, The Four Generations of Entity Resolution, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2021. URL: https://doi.org/10.2200/S01067ED1V01Y202012DTM064. doi:10.2200/S01067ED1V01Y202012DTM064.

[4] G. Simonini, L. Gagliardelli, S. Bergamaschi, H. V. Jagadish, Scaling entity resolution: A loosely schema-aware approach, Inf. Syst. 83 (2019) 145–165. URL: https://doi.org/10.1016/j.is.2019.03.006. doi:10.1016/j.is.2019.03.006.

[5] G. Simonini, L. Zecchini, S. Bergamaschi, F. Naumann, et al., Entity resolution on-demand, Proceedings of the VLDB Endowment 15 (2022) 1506–1518.

[6] P. Christen, A survey of indexing techniques for scalable record linkage and deduplication, TKDE 24 (2012) 1537–1555.

[7] G. Papadakis, J. Svirsky, A. Gal, T. Palpanas, Comparative analysis of approximate blocking techniques for entity resolution, PVLDB 9 (2016) 684–695.

[8] G. Papadakis, D. Skoutas, E. Thanos, T. Palpanas, Blocking and filtering techniques for entity resolution: A survey, ACM Comput. Surv. 53 (2020) 31:1–31:42. URL: https://doi.org/10.1145/3377455. doi:10.1145/3377455.

[9] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, K. Stefanidis, An overview of end-to-end entity resolution for big data, ACM Comput. Surv. 53 (2021) 127:1–127:42. URL: https://doi.org/10.1145/3418896. doi:10.1145/3418896.

[10] G. Papadakis, G. M. Mandilaras, L. Gagliardelli, G. Simonini, E. Thanos, G. Giannakopoulos, S. Bergamaschi, T. Palpanas, M. Koubarakis, Three-dimensional entity resolution with jedai, Inf. Syst. 93 (2020) 101565. URL: https://doi.org/10.1016/j.is.2020.101565. doi:10.1016/j.is.2020.101565.

[11] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederée, W. Nejdl, A blocking framework for entity resolution in highly heterogeneous information spaces, TKDE 25 (2012) 2665–2682.

[12] G. Papadakis, G. Koutrika, T. Palpanas, W. Nejdl, Meta-blocking: Taking entity resolution to the next level, TKDE 26 (2014) 1946–1960.

[13] G. Papadakis, G. Papastefanatos, G. Koutrika, Supervised meta-blocking, PVLDB 7 (2014) 1929–1940.

[14] L. Gagliardelli, G. Papadakis, G. Simonini, S. Bergamaschi, T. Palpanas, Generalized supervised meta-blocking, Proc. VLDB Endow. 15 (2022) 1902–1910. URL: https://www.vldb.org/pvldb/vol15/p1902-gagliardelli.pdf.

[15] L. Gagliardelli, G. Simonini, D. Beneventano, S. Bergamaschi, Sparker: Scaling entity resolution in spark, in: EDBT, 2019, pp. 602–605.

[16] G. Papadakis, G. Papastefanatos, T. Palpanas, M. Koubarakis, Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking., in: EDBT, 2016, pp. 221–232.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[18] H. Köpcke, A. Thor, E. Rahm, Evaluation of entity resolution approaches on real-world match problems, PVLDB 3 (2010) 484–493.
[19] D. Obraczka, J. Schuchart, E. Rahm, Eager: Embedding-assisted entity resolution for knowledge graphs, arXiv preprint arXiv:2101.06126 (2021).
[20] S. Das, A. Doan, P. S. G. C., C. Gokhale, P. Konda, Y. Govind, D. Paulsen, The magellan data repository, https://sites.google.com/site/anhaidgroup/projects/data, 2023.
[21] G. Simonini, S. Bergamaschi, H. V. Jagadish, BLAST: a loosely schema-aware meta-blocking approach for entity resolution, PVLDB 9 (2016) 1173–1184.
[22] G. D. Bianco, M. A. Gonçalves, D. Duarte, BLOSS: effective meta-blocking with almost no effort, Inf. Syst. 75 (2018) 75–89.
[23] G. Simonini, G. Papadakis, T. Palpanas, S. Bergamaschi, Schema-agnostic progressive entity resolution, IEEE Trans. Knowl. Data Eng. 31 (2019) 1208–1221. URL: https://doi.org/10.1109/TKDE.2018.2852763. doi:10.1109/TKDE.2018.2852763.