Big Data Research 6 (2016) 43–63

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Boosting the Efficiency of Large-Scale Entity Resolution with Enhanced Meta-Blocking

George Papadakis [a,*], George Papastefanatos [b], Themis Palpanas [c], Manolis Koubarakis [a]

[a] University of Athens, Greece
[b] Athena Research Center, Greece
[c] Paris Descartes University, France

ARTICLE INFO

Article history:
Received 15 March 2016
Accepted 29 August 2016
Available online 5 October 2016

Keywords:
Entity Resolution
Redundancy-positive blocking
Meta-blocking

ABSTRACT

Entity Resolution constitutes a quadratic task that typically scales to large entity collections through blocking. The resulting blocks can be restructured by Meta-blocking to raise precision at a limited cost in recall. At the core of this procedure lies the blocking graph, where the nodes correspond to entities and the edges connect the comparable pairs. There are several configurations for Meta-blocking, but no hints on best practices. In general, the node-centric approaches are more robust and suitable for a series of applications, but suffer from low precision, due to the large number of unnecessary comparisons they retain.

In this work, we present three novel methods for node-centric Meta-blocking that significantly improve precision. We also introduce a pre-processing method that restricts the size of the blocking graph by removing a large number of noisy edges. As a result, it reduces the overhead time of Meta-blocking by 2 to 5 times, while increasing precision by up to an order of magnitude for a minor cost in recall. The same technique can be applied as graph-free Meta-blocking, enabling for the first time Entity Resolution over very large datasets even on commodity hardware. We evaluate our approaches through an extensive experimental study over 19 voluminous, established datasets. The outcomes indicate best practices for the configuration of Meta-blocking and verify that our techniques reduce the resolution time of state-of-the-art methods by up to an order of magnitude.

1. Introduction

A common task in the context of Web Data is *Entity Resolution* (ER), i.e., the identification of different entity profiles that pertain to the same real-world object. Exhaustive solutions to this task suffer from low efficiency, due to their inherently quadratic complexity: every entity profile has to be compared with all others. This problem is accentuated by the continuously larger size of datasets that are now available on the Web. For example, the *LOD-Stats*[1] Web application recorded around a billion triples for Linked Open Data in December, 2011, which had grown to more than 100 billion triples by March, 2016. As a result, ER typically scales to large data collections through approximate techniques, which sacrifice recall to a controllable extent in order to enhance precision and time efficiency.

The most popular among these techniques is *blocking* [1–3]. It groups similar entities into clusters (called *blocks*) so that comparisons are executed only between the entities within each block [4,5]. Typically, blocking methods for Big Data have to overcome high levels of noise not only in attribute values, but also in attribute names, due to the unprecedented schema heterogeneity. For instance, Google Base[2] alone encompasses 100,000 distinct schemata that correspond to 10,000 entity types [6]. Most blocking methods deal with these high levels of noise through *redundancy* [1,7]: they place every entity profile into multiple blocks so as to reduce the likelihood of missed matches.

The simplest method of this type is Token Blocking [9,2]. It disregards schema information and semantics, creating a separate block for every token that appears in the attribute values of at least two entities. To illustrate its functionality, consider the entity profiles in Fig. 1(a), where p_1 and p_2 match with p_3 and p_4, respectively; Token Blocking clusters them in the blocks of Fig. 1(b), which place both pairs of duplicates in at least one common block

* Corresponding author.
E-mail addresses: gpapadis@di.uoa.gr (G. Papadakis), gpapas@imis.athena-innovation.gr (G. Papastefanatos), themis@mi.parisdescartes.fr (T. Palpanas), koubarak@di.uoa.gr (M. Koubarakis).

[1] http://stats.lod2.eu.

[2] http://www.google.com/base.

http://dx.doi.org/10.1016/j.bdr.2016.08.002
2214-5796/

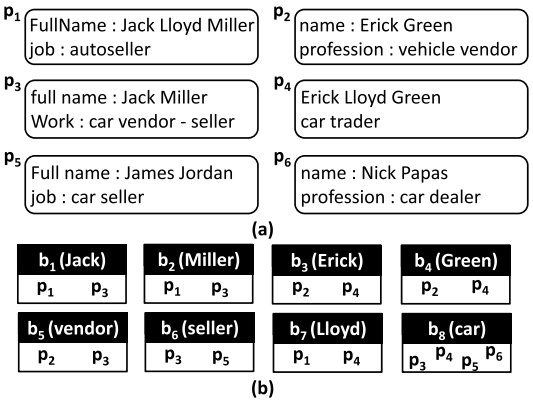

Fig. 1. (a) A set of entity profiles, and (b) the blocks of Token Blocking.

at the cost of 13 comparisons, in total. The resulting computational cost is high, given that the brute-force approach executes 15 comparisons.

This is a general trait of block collections that involve redundancy: in their effort to achieve high recall, they produce a large number of unnecessary comparisons. These come in two forms: the *redundant* ones repeatedly compare the same entity profiles across different blocks, while the *superfluous* ones compare non-matching entities. In our example, b_2 and b_4 contain one redundant comparison each, which are repeated in b_1 and b_3, respectively; all other blocks entail superfluous comparisons between non-matching entity profiles, except for the redundant comparison p_3–p_5 in b_8 (it is repeated in b_6). In total, the blocks of Fig. 1(b) involve 3 redundant and 8 superfluous out of the 13 comparisons.

Current state-of-the-art. To mitigate this phenomenon, methods such as Comparison Propagation [10] and Iterative Blocking [11] aim to process an existing block collection in the optimal way (see Section 2 for more details). Among these methods, Meta-blocking achieves the best balance between precision and recall, being one of the few techniques to scale well to millions of entities [7,8]. In essence, it restructures a block collection B into a new one B' that contains a significantly lower number of unnecessary comparisons, while detecting almost the same number of duplicates. This procedure operates in 2 steps.

First, it transforms B into the blocking graph G_B, which contains a node n_i for every entity p_i in B and an edge $e_{i,j}$ for every pair of *co-occurring entities* p_i and p_j (i.e., entities sharing at least one block). Fig. 2(a) depicts the graph for the blocks in Fig. 1(b). As no parallel edges are constructed, every pair of entities is compared at most once, thus eliminating all *redundant* comparisons.

Second, it annotates every edge with a weight analogous to the likelihood that the adjacent entities are matching, based on the blocks they have in common. For instance, the edges in Fig. 2(a) are weighted with the Jaccard similarity of the lists of blocks containing their adjacent entities. The edges with low weights correspond to *superfluous comparisons* and are pruned. A possible approach is to discard all edges with a weight lower than the overall mean one (1/4). This yields the pruned graph in Fig. 2(b).

Pruning algorithms of this type are called *edge-centric*, because they iterate over the edges of the blocking graph and retain the globally best ones. Higher recall is achieved by the *node-centric* pruning algorithms, which iterate over the nodes of the blocking graph and retain the locally best edges. These are the edges with the highest weights in each neighborhood and correspond to the most likely matches for each entity. In contrast, the edge-centric algorithms do not guarantee to include every entity in the restructured blocks. Their recall is lower than the node-centric algorithms by 20%, on average, when compared under the same settings [7].

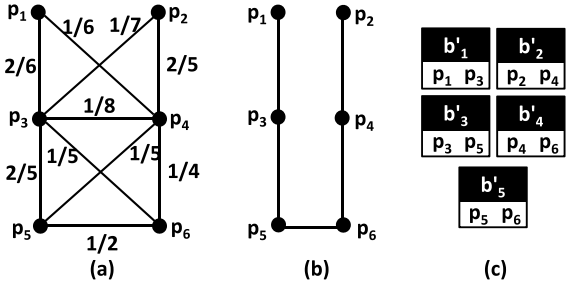

Fig. 2. (a) A blocking graph extracted from the blocks in Fig. 1(b), (b) one of the possible edge-centric pruned blocking graphs, and (c) the new blocks derived from it.

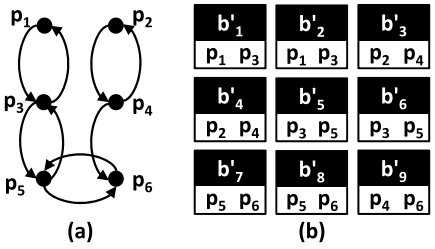

Fig. 3. (a) One of the possible node-centric pruned blocking graphs for the graph in Fig. 2(a). For clarity, the retained edges are directed and outgoing, since they might be preserved in the neighborhoods of both adjacent entities. (b) The new blocks derived from the pruned graph.

To illustrate the functionality of node-centric approaches, consider the pruned blocking graph in Fig. 3(a); for each node in Fig. 2(a), it has retained the adjacent edges that exceed the average weight of the neighborhood. Regardless of the type of the pruning algorithm, the restructured block collection B' is formed by creating a new block for every retained edge – as depicted in Figs. 2(c) and 3(b). In both cases, B' maintains the original recall, while reducing the number of executed comparisons to 5 and 9, respectively.

Open issues. Despite the significant enhancements in efficiency, Meta-blocking suffers from three drawbacks:

(i) Though more robust to recall, the node-centric pruning algorithms exhibit low efficiency, because they retain a considerable portion of redundant and superfluous comparisons. In most cases, their precision is lower than the edge-centric ones by 50% [7]. This is also illustrated in our example, where the restructured blocks of Fig. 3(b) contain 4 redundant comparisons in b'_2, b'_4, b'_6 and b'_8 and 3 superfluous in b'_5, b'_7 and b'_9; the edge-centric counterpart in Fig. 2(c) retains just 3 superfluous comparisons.

(ii) The processing of voluminous datasets involves a significant overhead. The corresponding blocking graphs comprise millions nodes that are strongly connected with billions edges. Inevitably, the pruning of such graphs is very time-consuming, leaving plenty of room for improving its efficiency (see Section 5.6).

(iii) Meta-blocking is difficult to configure. There are five different weighting schemes that can be combined with four pruning algorithms, thus yielding 20 *pruning schemes*, in total (see Section 3 for more details). As yet, there are no guidelines on how to choose the best configuration for the application at hand and the available resources.

Proposed solution. In this paper, we describe novel techniques for overcoming the weaknesses of Meta-blocking.

First, we propose three new node-centric pruning algorithms that achieve significantly higher precision than the existing ones. The most conservative approach, *Redundancy Pruning*, produces restructured blocks with no redundant comparisons and prunes up to 50% more comparisons. It achieves the same recall as the existing techniques, but its precision is almost the double. The other