To Compare or Not to Compare: Making Entity Resolution more Efficient

George Papadakis[$,♯], Ekaterini Ioannou[◇],
Claudia Niederée[♯], Themis Palpanas[‡], and Wolfgang Nejdl[♯]

[$] National Technical University of Athens, Greece gpapadis@mail.ntua.gr
[◇] Technical University of Crete, Greece ioannou@softnet.tuc.gr
[♯] L3S Research Center, Germany {surname}@L3S.de
[‡] University of Trento, Italy themis@disi.unitn.eu

ABSTRACT

Blocking methods are crucial for making the inherently quadratic task of Entity Resolution more efficient. The blocking methods proposed in the literature rely on the homogeneity of data and the availability of binding schema information; thus, they are inapplicable to the voluminous, noisy, and highly heterogeneous data of the Web 2.0 user-generated content. To deal with such data, attribute-agnostic blocking has been recently introduced, following a two-fold strategy: the first layer places entities into overlapping blocks in order to achieve high effectiveness, while the second layer reduces the number of unnecessary comparisons in order to enhance efficiency.

In this paper, we present a set of techniques that can be plugged into the second strategy layer of attribute-agnostic blocking to further improve its efficiency. We introduce a technique that eliminates redundant comparisons, and, based on this, we incorporate an approximate method for pruning comparisons that are highly likely to involve non-matching entities. We also introduce a novel measure for quantifying the redundancy a blocking method entails and explain how it can be used to a-priori tune the process of comparisons pruning. We apply our blocking techniques on two large, real-world data sets and report results that demonstrate a substantial increase in efficiency at a negligible (if any) cost in effectiveness.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithms, Experimentation, Performance

Keywords

Data Cleaning, Entity Resolution, Attribute-Agnostic Blocking

1. INTRODUCTION

Entity Resolution (ER) is the process of automatically identifying pairs of entities that correspond to the same real-world object. In principle, ER is a task with quadratic time complexity, since each entity has to be compared with all others. Approximate ER methods sacrifice *effectiveness* (i.e., the portion of detected entity matches) to a small and controlled extent in order to enhance *efficiency* (i.e., lower number of pair-wise comparisons). Among these methods, *data blocking* is the most prevalent one: it clusters the data into blocks, and handles the data inside each block separately, instead of operating on the entire collection [2]. Blocking techniques partition entities (or records) into blocks, such that potential duplicates are placed in the same block with a high probability, and the number of entities per block remains low.

Various blocking methods were proposed in the literature for homogeneous information spaces. They typically associate each record with a *Blocking Key Value* (BKV), and then operate exclusively on the BKVs [2]. Some prominent examples are the Sorted Neighborhood approach [4], the StringMap method [6], the q-grams blocking technique [3], canopy clustering [7] and Suffix Arrays approach [11]. An extension to this framework comprises the *iterative techniques*; these are based on the principle that more duplicates can be detected and more pair-wise comparisons can be saved through the iterative distribution of identified matches to subsequently (re-)processed blocks [5, 12]. The performance of these methods depends on the fine-tuning of a wealth of application- and data-specific parameters [11], a process that can be automated with the help of machine learning algorithms, as demonstrated in [8, 1]. A major requirement of these methods is that the schema describing the data at hand as well as the properties of its individual attributes are know a priori. Inevitably, though, this fundamental assumption is broken by the inherent characteristics of heterogeneous information spaces (i.e., loose schema binding, noise, missing or inconsistent values, as well as an unprecedented level of heterogeneity), turning them inapplicable.

To remedy this problem, more robust blocking methods with a higher level of redundancy are required to ensure effectiveness in the context of the voluminous, noisy, and highly heterogeneous data of the Web 2.0 user-generated content. An example of such a robust blocking approach is the recently introduced *attribute-agnostic blocking* method [9], which deals with an important subclass of ER in heterogeneous information spaces: the *Clean-Clean Entity Resolution*; that is, identifying the matching pairs of entities among two large, heterogeneous, individually clean, but overlapping collections of entities.

Attribute-agnostic blocking assumes no background knowledge of the data at hand. Instead, it deals with heterogeneous information spaces by disregarding their schemata and considering solely the values of their entity profiles. In essence, it consists of two layers; the first one focuses on achieving high effectiveness by building blocks of low-level granularity: each block corresponds to a

SWIM 2011, June 12, 2011, Athens, Greece.

specific token and contains all entities that have this token in one of their attribute values. Thus, duplicate entities are highly unlikely to have no block in common, as each entity is associated with multiple blocks resulting (i.e., *redundancy*). The second layer encompasses a set of methods that reduce the number of unnecessary and redundant comparisons (i.e., repeated comparisons among the same entities) of the resulting, overlapping blocks in order to enhance the efficiency of the ER process.

In this paper, we introduce an innovative method that enhances the second layer of attribute-agnostic blocking by eliminating the redundant and unnecessary comparisons it entails. The key observation in our method is that we can estimate how likely it is for a pair of entities to match, without having to explicitly compare them. Pairs whose matching likelihood lies below a predefined pruning threshold do not need to be compared, thus saving a large number of unnecessary comparisons. The value of the threshold depends on the level of redundancy that the underlying block-building method (i.e., the first layer) entails. We also introduce a metric that appropriately encapsulates redundancy levels and demonstrate how it can be used to automatically fine-tune the pruning threshold with a simple inspection of the data at hand. The main contributions of this paper can be summarized as follows:

1. As a basis for our new method, we introduce an extension of the attribute-agnostic method that allows the propagation of all executed comparisons, in order to avoid repeating those comparisons and eliminate redundancy (Section 2).

2. Building on comparisons propagation, we develop an approximation method for discarding comparisons between entities with a very low probability to match (Section 3). We also present a metric for rating the redundancy conveyed by each blocking method, and explain how it can be used to set the optimal threshold for comparisons pruning.

3. We report on the results of applying our modified blocking method on two large, real-world data sets. The outcomes of our experiments (Section 4) demonstrate a substantial increase in efficiency, at a negligible cost in effectiveness.

The rest of the paper is organized as follows: Section 2 defines the problem, and outlines the attribute-agnostic method we are extending. In Section 3, we present the additional steps we incorporate in the original method and introduce the modified attribute-agnostic method. Section 4 reports the results of our experimental evaluation, and, finally, Section 5 concludes the paper, briefly discussing our plans for future work.

2. PRELIMINARIES

In this section, we introduce and formally define the main notions that lie at the core of our work and present an outline of the blocking method we extend (i.e., attribute-agnostic blocking).

2.1 Blocking-based Entity Resolution

Central to our work are the notions of entities and of *entity profiles*, which are used to represent them. To formally describe them, together with the related notions of *entity collection*, *blocking schema*, *blocks* and *inner blocks*, we adopt the definitions that were introduced in [9]. For completeness, these definitions are summarized below:

Definition 1. *An* ***entity profile*** $\mathbf{p}$ *is a tuple* $\langle id, A_p \rangle$*, where* A_p *is a set of attributes {a} and id is the global identifier for the profile. Each* ***attribute*** $a_i \in A_p$ *is a tuple* $\langle n_i, v_i \rangle$*, consisting of an* ***attribute name*** $\mathbf{n_i}$ *and an* ***attribute value*** $\mathbf{v_i}$.

Note that this simple model supports not only structured data but also tag-style annotations (represented by values with empty attribute names) and even relationships between entities (represented by entity identifiers as attribute values).

Definition 2. *An* ***entity collection*** $\mathcal{E}$ *is a tuple* $\langle A_E, V_E, ID_E, P_E \rangle$*, where* A_E *is the set of attribute names appearing in it,* V_E *is the set of values used in it,* ID_E *is the set of global identifiers contained in it, and* $P_E \subseteq ID_E$ *is the set of entity profiles that it comprises.*

Definition 3. *A* ***blocking scheme*** $\mathbf{bs_{t,c}}$ *for an entity collection* $\mathcal{E}$ *is defined by a transformation function* $f_t : \mathcal{E} \mapsto T$ *and a set of constraint functions* $f_c^i : T \times T \mapsto \{true, false\}$*. The* ***transformation function*** $\mathbf{f_t}$ *derives the appropriate representation for blocking from the complete entity profiles of* $\mathcal{E}$ *or parts of them. The* ***constraint function*** $\mathbf{f_c^i}$ *is a transitive and symmetric function that encapsulates the condition that has to be satisfied by two entities, if they are to be placed in the same block* b_i.

Note that this definition is comprehensive enough to encapsulate any blocking method, both for homogeneous and for heterogeneous information spaces. For example, existing blocking methods use a transformation function that extracts a BKV from an entity profile and a set of constraint functions that define blocks on the equality - or similarity - of the BKVs. Applying a blocking scheme $bs_{t,c}$ on an entity collection $\mathcal{E}$ yields a set of blocks $\mathcal{B}^{\mathcal{E}}_{\mathbf{t,c}}$. Its elements are defined as follows:

Definition 4. *Given an entity collection* $\mathcal{E}$ *and a blocking scheme* $bs_{t,c}$*, a* ***block*** $\mathbf{b_i} \in \mathcal{B}^{\mathcal{E}}_{t,c}$ *is the maximal subset of* $\mathcal{E}$ *- with a minimum cardinality of 2 - that is defined by the transformation function* f_t *and one of the constraint functions* f_c^i *of* $bs_{t,c}$*:* $b_i \subseteq \mathcal{E} \wedge \forall p_1, p_2 \in \mathcal{E} : f_c^i(f_t(p_1), f_t(p_2)) = true \Rightarrow p_1, p_2 \in b_i$.

In the following, we focus on an important subclass of the Entity Resolution problem, namely the case of merging two duplicate-free entity collections $\mathcal{E}_1$ and $\mathcal{E}_2$ (*Clean-Clean ER*). Applying a blocking scheme $bs_{t,c}$ on the union of two entity collections $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$ yields a set of blocks $\mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{\mathbf{t,c}}$. In this context, a valid block has to contain entities from both input entity collections. Each block b_i can be, thus, divided into two parts, $b_{i,1}$ and $b_{i,2}$, which we call *inner blocks*:

Definition 5. *For a block* $b_i \in \mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c}$*, the* ***inner block*** $\mathbf{b_{i,1}}$ *(*$\mathbf{b_{i,2}}$*) is the subset of elements in* b_i *that originate from the entity collection* $\mathcal{E}_1$ *(*$\mathcal{E}_2$*):* $b_{i,k} = \{p \in b_i : p \in \mathcal{E}_k\}$ *for* $k = 1, 2$.

The blocking-based Clean-Clean ER process consists of iterating over the resulting set of blocks $\mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c}$ in order to compare the entities of their inner blocks between them. We use symbol $\mathbf{m_{i,j}}$ to denote the match between profiles p_i and p_j that were identified to describe the same real-world object. Thus, the output of a blocking method is a set of matches, which we denote with symbol $\mathbf{M}$.

To address the characteristics of heterogeneous information spaces, *redundancy-bearing* blocking methods have been recently introduced [11, 9, 12]: they associate each entity with multiple, overlapping blocks, thus minimizing the likelihood of missed matches and increasing effectiveness. Efficiency, on the other hand, is significantly downgraded, due to the redundant comparisons between pairs of entities that appear in many blocks. Apparently, the higher the redundancy conveyed by a blocking method, the lower the efficiency of the ER process.

The level of redundancy a blocking method entails is proportional to the number of blocks an entity is placed in, on average;

that is, the more blocks are associated with each entity, the higher the overall redundancy. To formally express this relation, we introduce the metric of *individual Blocking Cardinality*:

Definition 6. *Given an entity collection $\mathcal{E}$ and the set of blocks $\mathcal{B}^{\mathcal{E}}_{t,c}$, their* **individual Blocking Cardinality** *(*$\mathbf{iBC^{\mathcal{E}}_{t,c}}$*) is defined as the number of blocks $b_i \in \mathcal{B}^{\mathcal{E}}_{t,c}$ an entity $p \in \mathcal{E}$ is placed on average:*

$$iBC^{\mathcal{E}}_{t,c} = \frac{\sum_{p \in \mathcal{E}} |b_i \in \mathcal{B}^{\mathcal{E}}_{t,c} : p \in b_i|}{|\mathcal{E}|},$$

where $|\mathcal{E}|$ denotes the size (i.e., number of entities) of the entity collection $\mathcal{E}$.

In particular for the Clean-Clean ER, we employ an aggregate measure of blocking cardinality in order to be able to directly compare the redundancy of two different applications. We call this metric *overall Blocking Cardinality* and formally define it as follows:

Definition 7. *Given two clean, overlapping entity collections - $\mathcal{E}_1$ and $\mathcal{E}_2$ - along with the set of blocks $\mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c}$, their* **overall Blocking Cardinality** *(*$\mathbf{oBC^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c}}$*) is defined as the number of blocks $b_i \in \mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c}$ an entity $p \in (\mathcal{E}_1 \cup \mathcal{E}_2)$ is placed on average:*

$$oBC^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c} = \frac{\sum_{p \in (\mathcal{E}_1 \cup \mathcal{E}_2)} |b_i \in \mathcal{B}^{\mathcal{E}_1 \times \mathcal{E}_2}_{t,c} : p \in b_i|}{|\mathcal{E}_1| + |\mathcal{E}_2|}.$$

Blocking cardinality - both individual and overall - takes values in the interval $[0, \infty)$, with higher values denoting higher levels of redundancy. Values lower than 1 indicate blocking methods that fail to place each entity in at least one block. A value equal to 1 corresponds to a blocking scheme that partitions an entity collection in non-overlapping blocks. Note that blocking cardinality is defined with respect to a blocking method and an entity collection. The reason is that the same blocking scheme can produce different levels of redundancy when applied on different entity collections, because they convey different quantities of information (i.e., number of name-value pairs per entity). This is particularly important in the case of resolving two duplicate-free entity collections: the blocking cardinality of each collection is different, depending not only on their individual quantities of information, but also on the amount of information these two collections have in common.

We build on an existing blocking method that achieves high effectiveness, with the aim of enhancing its efficiency, regardless of the redundancy levels it conveys. Our focus is as follows:

Problem 1. *Given two heterogeneous, individually clean entity collections $\mathcal{E}_1$ and $\mathcal{E}_2$, detect the vast majority of the matches M they contain as efficiently as possible (i.e., with the minimum number of pair-wise entity comparisons).*

2.2 Attribute-agnostic Blocking

Attribute-agnostic blocking comprises five techniques, which are divided into two strategy layers: the *effectiveness tier* and *efficiency tier*. The former encompasses a technique for creating overlapping blocks, namely **Block Building**. It considers only the attribute values of an entity profile, tokenizing them on their special characters. Each token that appears in both input entity collections forms a block, encompassing all the entities that contain it in their profile. Thus, each entity is associated with multiple blocks, increasing the effectiveness and the robustness of the method to noise.

The *efficiency tier* consists of four techniques that aim at safely reducing the required number of pair-wise comparisons. They are described in the following, in the order they are executed:

Block Purging. This step aims at removing *oversized blocks*: these are blocks that contain a large number of entities, which typically share other blocks, as well. Consequently, the majority of the pair-wise comparisons these blocks entail are *redundant* ones, having a negligible contribution to effectiveness, but a disproportionately negative impact on efficiency. A conservative way of determining such blocks is to examine whether they satisfy the following condition: $\rho \cdot min(|\mathcal{E}_1|, |\mathcal{E}_2|) \leq min(|b_{i,1}|, |b_{i,2}|)$, where ρ was experimentally set to 0.005.

Duplicate Propagation. Since the given collections are individually clean, each entity matches with at most one entity of the other collection. As a result, entities that have already been identified as duplicates do not need to be compared with any other entity. We can significantly increase the efficiency, by sparing those comparisons that involve any of the matching entities.

Block Scheduling. The earlier a pair of duplicates is identified, the more comparisons are saved by duplicate propagation. Thus, we can boost efficiency by examining those blocks first that exhibit the best trade-off between cost (i.e., number of comparisons) and gain (i.e., expected number of matches). This relation is encapsulated by a metric called *block utility* ($\mathbf{u_i}$), which was estimated through a probabilistic analysis to be equal to: $u_i = 1/max(|b_{i,1}|, |b_{i,2}|)$ [9]. Based on this definition, blocks are scheduled and processed in descending order of their utility.

Block Pruning. Block Scheduling ensures that the lower a block is placed in the processing queue, the less likely it is to contain *non-redundant matches* (i.e., pairs of duplicates that share no other block). This enables pruning a part of the "tail" of the processing list without any considerable impact on effectiveness. Block Pruning is based on the metric *duplicate overhead*, which keeps track of the cost of finding new matches, and terminates the ER process as soon as this cost exceeds a threshold.

The thorough experimental evaluation of [9] verified that this blocking framework is scalable, requiring an average of around 100 comparisons per entity. Though a quite satisfactory performance, in this work we aim at further improving it.

3. REDUCING COMPARISONS

In this section, we present a series of techniques that enhance the efficiency of attribute-agnostic blocking, at a controllable (if any) cost in effectiveness. The development of the methods is inspired by two observations. The first is that although duplicate propagation is quite effective in improving the efficiency of the blocking approach, it cannot prevent the repetition of comparisons that entail non-matching entities. The second observation is that Block pruning (i.e. discarding unpromising blocks) is quite coarse- grained, since it stays on the block level. A more fine-grained assessment and approximation holds the promise to lead to better pruning decisions and higher flexibility.

We, therefore, developed two additional methods for further enhancing the efficiency of attribute-agnostic blocking. Both are based on a common data structure, a type of "inverted index", which points from entities to the blocks they are contained in. The first method constitutes an indirect form of *Comparisons Propagation*, which supersedes duplicate propagation by avoiding the repetition of all types of comparisons, instead of just the comparisons of identified matches; the second method - *Comparison Pruning* - replaces Block Pruning by looking into individual comparisons instead of entire blocks, when deciding where to prune. In addition, a third method - *Block Enumeration* - has been incorporated into the approach, serving as a preparatory step that facilitates the functionality of the other two.

Our work exclusively focuses on the second strategy layer, treating the first one as a black box. The modified efficiency tier consists of the following steps (they are presented in the order they are executed):

1. Block Purging*,
2. Block Scheduling*,
3. Block Enumeration,
4. Duplicate Propagation*,
5. Comparisons Propagation, and
6. Comparisons Pruning.

The steps that are marked with $*$ remain unchanged, and are implemented and executed as described in [9]. In the following paragraphs we elaborate on the functionality of the new steps.

3.1 Block Enumeration

This is a preparatory, but core step that forms the basis for the operation of Comparisons Propagation and Pruning. It serves the need of uniquely and efficiently identifying each block, so as to facilitate the association of each entity with all the blocks that contain it. Thus, Block Enumeration merely consists of assigning a unique index to each block. For reasons explained below, this index is assigned based on the position of the block in the processing list, as it is determined by Block Scheduling (this explains why this step is placed right after scheduling). This procedure has a linear complexity (i.e., $O(|\mathcal{B}_{t,c}^{\mathcal{E}_1 \times \mathcal{E}_2}|)$), both with respect to time and space.

Note that, from now on, symbol b_i will denote that the corresponding block is placed in the i-th position of the processing list.

3.2 Comparisons Propagation

The aim of this step is to completely eliminate redundant comparisons, without any impact on effectiveness. In other words, the goal is to identify whether a pair of entities in block b_i has already been compared in another block b_j, where $j < i$; in this case, we do not need to compare it again in b_i, thus saving an unnecessary comparison.

A solution for this task was proposed in [10] in the form of a generic method for eliminating redundancy, that is able to be adapted and incorporated into any blocking scheme. However, the authors did not investigate how it works in the context of the attribute-agnostic method, i.e., whether it can significantly reduce the required number of comparisons when combined with Block Pruning. In this paper, we experimentally examine its functionality inside the original attribute-agnostic method in order to decide whether it alone suffices for covering the need for higher efficiency. Most importantly, though, Comparisons Propagation serves in our case as a basis for the next efficiency-enhancing technique.

The main idea behind Comparisons Propagation is to propagate executed comparisons indirectly, without explicitly storing information about them. To achieve this goal, we employ the shared data structure that is depicted in Figure 1. It constitutes a *hash table*, with the *keys* of its entries comprising the IDs of the entities of the input collections ($\mathcal{E}_1$ and $\mathcal{E}_2$). Each *value* consists of the list of the indices of the blocks that contain the corresponding entity. The employed block indices are those defined by Block Enumeration. For efficiency reasons, the blocks ids are sorted in ascending index order within individual lists. In this way, the overlap between the block list of two entities can be identified in linear time (see below). The sorting of the blocks has to be done only once (at construction time), while the checking procedure is repeatedly executed, justifying the investment in block list sorting.

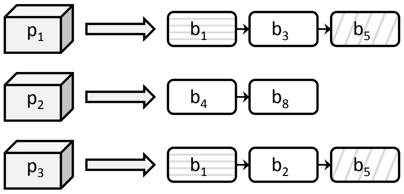


Figure 1: The data structure in Comparisons Propagation.

Based on this data structure, a pair of entities is compared only if the **Least Common Block Index Condition** holds, i.e., the lowest common block index of these two entities is equal with the index of the current block. This condition ensures that the current block is the first one in the processing list that contains both entities. Otherwise (if the former index is lower than the latter), the entities have already been compared in another block, and the comparison would be redundant. Consider, for example, the entities p_1 and p_3 in Figure 1. They have two blocks in common, b_1 and b_5. Their least common block index is b_1, which means that the condition is satisfied in their first common block, but not in the second. Thus, we save the redundant comparison in block b_5.

Checking this condition is linear in the number of blocks associated with each entity. In the average case, though, it is equal to Blocking Cardinality. The reason is that retrieving the value of a specific key from a hash table is an operation of constant time complexity, while the ordering of block indices enables the identification of the least common index by iterating over the two lists in parallel. The time complexity for building this data structure is linear with respect to the block set size (i.e., $O(|\mathcal{B}_{t,c}^{\mathcal{E}}|)$); its space complexity, on the other hand, is linear with respect to the size of the input entity collections, depending, of course, on the overall level of redundancy, i.e., $O(oBC_{t,c}^{\mathcal{E}_1 \times \mathcal{E}_2} \cdot (|\mathcal{E}_1| + |\mathcal{E}_2|))$, in average.

3.3 Comparisons Pruning

The goal of this technique is to replace the last step of the original attribute-agnostic method, namely Block Pruning. Similar to it, Comparisons Pruning is an approximation technique that discards selected comparisons in order to significantly improve efficiency, at a controllable cost in effectiveness. Its main difference from Block Pruning is that it operates on a lower level of granularity, thus being able to take more accurate pruning decisions: Block Pruning treats the set of comparisons within a block collectively, as it operates on block level, while our new technique operates on the level of individual comparisons, adjusting its pruning criterion to a particular pair of entities.

In more detail, Comparisons Pruning decides, before comparing a specific pair of entities, whether they are likely to match or not; in case matching is not likely, the comparison is avoided (i.e., it is pruned). The evidence for this decision is drawn from the overlap between the corresponding lists of block indices. As explained in Section 3.2, this information is already available through the data structure maintained for Comparisons Propagation and is efficiently accessed through the cheap search functionality of the hash table.

To estimate the overlap between two lists of blocks, we employ the Jaccard similarity. In more detail, we call *Entity Similarity* - $\mathbf{ES_{i,j}}$ the portion of common block ids between two entities p_i and

p_j, and define it as:

$$ES_{i,j} = \frac{|p_i.indices() \cap p_j.indices()|}{|p_i.indices() \cup p_j.indices()|} = \frac{|p_i.indices() \cap p_j.indices()|}{|p_i.indices()| + |p_j.indices()| - |p_i.indices() \cap p_j.indices()|},$$

where $p_k.indices()$ denotes the block indices associated with the entity profile p_k. The last part of this equation denotes that we basically need to detect the number of common indices between two entities, in order to estimate their Entity Similarity. The time complexity of this operation is linear with respect to the number of indices associated with them: due to their ordering, it suffices to iterate over the two lists of indices just once (and in parallel).

To decide whether a pair of entities justifies its detailed comparison, we compare their Entity Similarity with a threshold, ES_{min}, that denotes the minimum allowed similarity value; that is, the comparison is executed only if this threshold is exceeded. The actual value of ES_{min} is set in a generic way that considers the average case, and is adapted to the redundancy inherent in the given blocking scheme and entity collection(s). More specifically, the formula for deriving the value of ES_{min} is the following:

$$ES_{min} = \frac{a \cdot min(iBC_{t,c}^{\mathcal{E}_1}, iBC_{t,c}^{\mathcal{E}_2})}{iBC_{t,c}^{\mathcal{E}_1} + iBC_{t,c}^{\mathcal{E}_2} - a \cdot min(iBC_{t,c}^{\mathcal{E}_1}, iBC_{t,c}^{\mathcal{E}_2})} \quad (1)$$

where a takes values in the interval $(0, 1]$ (i.e., it should be always higher than 0, otherwise no comparison is pruned).

The rationale behind this threshold is the following: given a blocking scheme $bs_{t,c}$, each profile of the entity collection $\mathcal{E}_k$ is associated with $iBC_{t,c}^{\mathcal{E}_k}$ blocks, on average; for two entities of $\mathcal{E}_1$ and $\mathcal{E}_2$ to be considered similar enough, we demand that they share $a\%$ of the minimum individual Blocking Cardinality. Apparently, the higher the value of a, the higher the corresponding threshold and, thus, the stricter the definition of similarity. For our purposes, we consider a value of $a = 0.25$. As explained in the next section, this is a conservative threshold, ensuring that comparisons are not harshly pruned, and that effectiveness is not affected to a large extent. Note, also, that this threshold is easily computed through a simple inspection of the input blocks, adjusting the functionality of Comparisons Pruning to the data at hand with a minimum cost.

4. EXPERIMENTAL EVALUATION

In this section, we present a thorough experimental evaluation of our suggested techniques on two real-world, large and heterogeneous information spaces. All methods were fully implemented in Java 1.6, employing the Lucene search engine library for the blocking functionality[1]. All experiments were performed on a server with Intel Xeon 3.0GHz, running Linux with kernel version 2.6.18.

4.1 Experimental Setup

Data Sets. In the course of our thorough experimental study, we employed the two data sets that were used in [9] in order to directly compare the original method with its enhanced version. Their technical characteristics are summarized in Table 1.

	D_{movies}		$D_{infoboxes}$	
	DBPedia	**IMDB**	**DBPedia$_1$**	**DBPedia$_2$**
Entities	27,615	23,182	1,190,734	2,164,058
Name-Value Pairs	186,013	816,012	17,453,516	36,653,387
Avg. Profile Size	6.74	35.20	14.66	16.94
iBC	8.27	39.81	14.78	15.71
Duplicates	22,405		892,586	
Blocks	40,430		1,210,262	
oBC	22.52		15.38	

Table 1: Overview of the evaluation data sets.

The first data set (D_{movies}) consists of a collection of movies shared among IMDB and DBPedia. It contains around 50,000 movies, split almost equally between the two entity collections. Note, however, that the quantity of information each collection conveys is quite different: the DBPedia movies are described by 7 name-value pairs, on average, while the IMDB ones consist of 35 pairs, on average. This results in proportionally different levels of redundancy, with the former collection exhibiting 5 times less individual blocking cardinality that the latter one.

The second data set ($D_{infoboxes}$) consists of two different versions of the DBPedia Infobox Data Set[2]; they contain all name-value pairs of the infoboxes in the articles of Wikipedia's english version, extracted at specific points in time. The older collection, $DBPedia_1$, is a snapshot from October 2007, whereas $DBPedia_2$ dates from October 2009. The large time period that intervenes between the two collections renders their resolution challenging, as only 25% of all name-value pairs is shared among them [9]. As expressed by their average profile sizes, however, their individual quantities of information are similar, resulting in a small difference in their individual blocking cardinalities.

It is worth noting here, that, although $D_{infoboxes}$ is much larger than D_{movies} (2.35 million entities described by 54 million name-value pairs in comparison to 50 thousand entities and 1 million name-value pairs), the latter entails a considerably higher level of redundancy. As denoted by their overall blocking cardinality, each entity of D_{movies} is placed in 22.5 blocks, on average, whereas each entity of $D_{infoboxes}$ is associated with just 15.4 blocks (on average). Below, we explain how this affects the performance of our method on the individual data sets.

Metrics. We follow [1, 11, 8, 9] and use two established metrics for blocking techniques. The first one is **Pair Completeness** (*PC*), which denotes the portion of matches existing in the data set that are identified as duplicates by the blocking method. It is computed as follows: $PC = dm/gm$, where dm stands for the number of detected matches, and gm represents the number of ground-truth matches. It takes values in the interval $[0, 1]$, with higher values indicate higher *effectiveness* of the blocking method. The second metric is **Reduction Ratio** (*RR*), which expresses the reduction a blocking method achieves in the number of pair-wise comparisons with respect to those required by the baseline method. It is defined as follows: $RR = 1 - mc/bc$, where mc stands for the number of comparisons the considered method entails, and bc expresses the number of comparisons of the baseline method. RR takes values in the interval $[0, 1]$ (for $mc \leq bc$), with higher values denoting higher *efficiency*. The baseline method for estimating RR is the original attribute-agnostic method, presented in [9].

4.2 Performance Comparison

This section contains a series of experiments we conducted in order to investigate whether introducing Comparisons Propagation in the original attribute-agnostic method suffices for highly efficient results. To this end, we created a modified attribute-agnostic version merely by adding Comparisons Propagation between Duplicate Propagation and Block Pruning, which in the remaining text we will call *Modified Version*. We compare this method with the original one and the method presented in Section 3, which comprises both Comparisons Propagation and Pruning with the ES_{min}

[1] http://lucene.apache.org

[2] http://wiki.dbpedia.org/Datasets

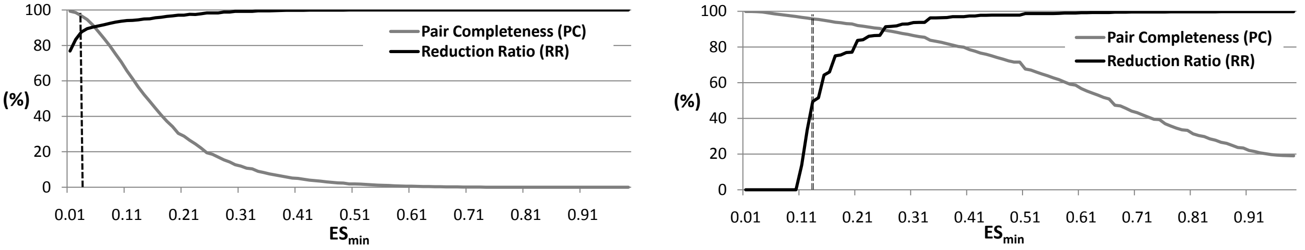
Method	Compar.	Duplicates	PC	RR
Attribute-agnostic	1.03×10^6	22,268	99.39%	-
Modified Version	2.39×10^5	22,268	99.39%	76.79%
Comp. Pruning	1.09×10^5	21,194	94.59%	89.59%

(a) D_{movies} data set.

Method	Compar.	Duplicates	PC	RR
Attribute-agnostic	1.12×10^8	838,760	93.97%	-
Modified Version	9.80×10^7	853,720	95.65%	12.53%
Comp. Pruning	5.42×10^7	851,108	95.35%	51.56%

(b) $D_{infoboxes}$ data set.

Table 2: Comparison of the performance with respect to PC and RR between the original attribute-agnostic method, its Modified Version that encompasses Comparisons Propagation and the method of Section 3 (marked as Comp. Pruning).



(a) D_{movies} data set.

(b) $D_{infoboxes}$ data set.

Figure 2: Sensitivity analysis of the ES_{min} threshold with respect to both metrics, PC and RR. The vertical, dotted lines correspond to the actual value of ES_{min} for a = 0.25 in Formula 1.

threshold corresponding to $a = 0.25$ (this method is noted as *Comparisons Pruning* in the following). The results of our experimental study are depicted in Tables 2(a) and (b) for the data sets D_{movies} and $D_{infoboxes}$, respectively.

We can easily notice that both new methods significantly outperform the original one with respect to efficiency, while incurring no negative impact on effectiveness - in most of the cases. In fact, only Comparisons Pruning over D_{movies} exhibits a significantly - but moderately - lower value for *PC* by 4.8%. This is probably related to the very low individual Blocking Cardinality of DBPedia movies: they are placed in just 8 blocks, on average, and the requirement of sharing at least 2 blocks (i.e., $a = 0.25$) with a potential match of IMDB movies turns out to be high for many of them. On the other hand, the higher levels of individual Blocking Cardinality in $D_{infoboxes}$ allow for a slight, but considerable improvement (around 1.5%) in its *PC* for both methods.

The difference in the performance of the two novel methods is worth further analysis. First of all, we can see that the Modified Version has quite different behavior in the two datasets we are considering: in D_{movies}, it has a considerably high RR, that is close to that of Comparisons Pruning, while the RR in $D_{infoboxes}$ remains at low levels. Apparently, this can be explained by the higher levels of redundancy in the former data set, as explained above. Phenomenally, though, the Modified Version outperforms Comparisons Pruning with respect to effectiveness, while their difference with respect to efficiency is not as high as one would expect: the *RR* of the former method is lower by 13% and 39% for D_{movies} and $D_{infoboxes}$, respectively. In practice, however, Comparisons Pruning involves in both cases approximately half the comparisons of the Modified Version: if we computed the *RR* of Comparisons Pruning with respect to the latter method, its values would be around 50% for both data sets. Thus, we believe that its slightly lower *PC* in both data sets is worth the effort. Besides, the final value of *PC* remains around 95% in both cases.

Last but not least, note that dividing the actual number of comparisons required by Comparisons Pruning by the total size of the input collections yields for both data sets 10 comparisons, on average, per entity. This is an order of magnitude lower than the 100 comparisons that were required by the original method [9].

4.3 Threshold Sensitivity Analysis

We also analyzed the performance of Comparisons Pruning for the following range of valid values for the parameter ES_{min}: $0.01 \leq ES_{min} \leq 1$[3], for both data sets. The goal is to verify that the method of threshold setting we presented above provides a conservative trade-off between effectiveness and efficiency. In addition, we intend to identify interesting patterns in the variation of the method's performance. The results of this evaluation are depicted in Figures 2 (a) and (b) for D_{movies} and $D_{infoboxes}$, respectively. We can clearly notice the trade-off between efficiency and effectiveness, as the higher the value of ES_{min} (x-axis), the lower the *PC* and the higher the *RR*.

The vertical, dotted lines in both figures correspond to the similarity threshold for $a = 0.25$; for D_{movies} this results in $ES_{min} = 0.04$, while for $D_{infoboxes}$ it corresponds to $ES_{min} = 0.14$. As noted above, it is interesting that both lines intersect the line of *PC* at 95%. There is no pattern with respect to *RR*; instead, there is a discrepancy in efficiency gains between the two diagrams for low values of the threshold (i.e., in the beginning of the x-axis). More specifically, we notice that in the case of the D_{movies} very small decreases in *PC* result in large gains in *RR* (around 80%). In the case of $D_{infoboxes}$, however, the value of *RR* remains 0 for small decrease in *PC*, and only raises to 50% (approximately) for larger values of ES_{min}. This can be explained by the higher levels of redundancy that D_{movies} entails, in comparison to $D_{infoboxes}$: as demonstrated in Table 1, the former has an overall Blocking Cardinality of 22, while the latter of just 15. Thus, the step of Comparisons Propagation conveys much higher comparison savings for the former data set than for the latter. This is a strong indication that the performance of our method with respect to efficiency depends on the redundancy of the involved data sets.

Another interesting pattern that can be drawn from these experiments is the intersection point of the *PC* and the *RR* lines: in both cases it is around 90% of the y-axis. This means that by reducing *PC* to 90% we spare 90% of the original comparisons. However, this performance is achieved for different values of a ($a = 0.28$ for D_{movies} and $a = 0.43$ for $D_{infoboxes}$). This is another indication that the method might require some parameter adaptation for different applications.

Last but not least, we can notice that the *PC* and the *RR* lines of D_{movies} change in a smoother way than those of $D_{infoboxes}$. In particular, the *RR* line of the latter consists of parts with negligible increase, followed by sharp rises. This should be attributed to the different values of the individual Blocking Cardinalities of

[3] $ES_{min} = 0$ is meaningless in our case - it does not discard any comparison. For similar reasons also values below 0.01 were discarded.

the involved entity collections: the minimum *iBC* of $D_{infoboxes}$ is much lower than that of D_{movies}; thus, small changes in ES_{min} have a higher impact on Comparisons Propagation for D_{movies}, whereas for $D_{infoboxes}$ large variations are required to exhibit the same level of changes.

5. CONCLUSIONS

In this paper, we extended the attribute-agnostic method, coined in [9], in order to achieve significant reductions in the number of comparisons required per entity. First, we incorporated Comparisons Propagation into the original framework and proved that it considerably enhances the overall performance, leaving though enough space for further improvement. To this end, we replaced the last step of the initial method (i.e., Block Pruning) with Comparisons Pruning: an approximation method that works at a lower level of granularity, and results in better pruning decisions. The optimal, conservative threshold for this procedure is determined in relation to Blocking Cardinality, a novel measure we introduced for estimating the redundancy of a blocking method, not only with respect to an individual entity collection, but also with respect to a pair of collections that are to be resolved. The results of our thorough experimental evaluation verified a significant increase in efficiency (by an order of magnitude), while Recall is maintained at high levels (over 95%).

In the future, we plan to further enhance the efficiency of our method, through the incorporation of parallelization techniques. More specifically, we plan to adapt it to the high-performance framework of MapReduce. We also intend to extend our method in order to be able to handle the Dirty-Dirty case of Entity Resolution (i.e., the identification of matching entities among two overlapping entity collections, which contain duplicates in themselves, as well).

References

[1] M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, pages 87–96, 2006.

[2] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *TKDE*, 19(1):1–16, 2007.

[3] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[4] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *SIGMOD Conference*, pages 127–138, 1995.

[5] H. Kim and D. Lee. HARRA: fast iterative hashed record linkage for large-scale data collections. In *EDBT*, pages 525–536, 2010.

[6] C. Li, L. Jin, and S. Mehrotra. Supporting efficient record linkage for large data sets using mapping techniques. *WWW Journal*, 9(4):557–584, 2006.

[7] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, pages 169–178, 2000.

[8] M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, 2006.

[9] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *WSDM*, pages 535–544, 2011.

[10] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, 2011 (to appear). Available at: http://l3s.de/~papadakis/papers/redundancyElimination.pdf.

[11] T. Vries, H. Ke, S. Chawla, and P. Christen. Robust record linkage blocking using suffix arrays. In *CIKM*, pages 305–314, 2009.

[12] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD Conference*, pages 219–232, 2009.