Beyond 100 Million Entities: Large-scale Blocking-based Resolution for Heterogeneous Data

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

A prerequisite for leveraging the vast amount of data available on the Web is Entity Resolution, i.e., the process of identifying and linking data that describe the same real-world objects. To make this inherently quadratic process applicable to large data sets, blocking is typically employed: entities (records) are grouped into clusters - the *blocks* - of matching candidates and only entities of the same block are compared. However, novel blocking techniques are required for dealing with the noisy, heterogeneous, semi-structured, user-generated data in the Web, as traditional blocking techniques are inapplicable due to their reliance on schema information. The introduction of redundancy, improves the robustness of blocking methods but comes at the price of additional computational cost.

In this paper, we present methods for enhancing the efficiency of redundancy-bearing blocking methods, such as our attribute-agnostic blocking approach. We introduce novel blocking schemes that build blocks based on a variety of evidences, including entity identifiers and relationships between entities; they significantly reduce the required number of comparisons, while maintaining blocking effectiveness at very high levels. We also introduce two theoretical measures that provide a reliable estimation of the performance of a blocking method, without requiring the analytical processing of its blocks. Based on these measures, we develop two techniques for improving the performance of blocking: combining individual, complementary blocking schemes, and purging blocks until given criteria are satisfied. We test our methods through an extensive experimental evaluation, using a voluminous data set with 182 million heterogeneous entities. The outcomes of our study show the applicability and the high performance of our approach.

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

The Linked Data principles[1] are of central importance for the effective re-use of the (semi-)structured, heterogeneous data that are increasingly becoming available in the Web. They enable exposing, sharing, and connecting these data, thus making it possible to leverage the investment in creating and collecting them [2]. However, the currently available Web data are not as "linked" as this paradigm envisions; content providers tend to use their own entity (resource) identifiers and schemata rather than re-using existing ones. As a result, the Web of Data ends up consisting of heterogeneous, overlapping islands of knowledge that pose significant challenges in the effort to combine entity information from different sources [9].

To remedy this problem, *Entity Resolution* (ER) is typically used; it is the process of automatically identifying sets of entities that correspond to the same real-world object. In principle, ER is a task of quadratic complexity, as it requires comparing each entity with all others. To scale it to large data collections, approximate ER methods drastically reduce the required number of comparisons, sacrificing some of their effectiveness (i.e., number of detected duplicates) in order to enhance their efficiency. Among these methods, *data blocking* is the most prevalent one: it clusters data into blocks, such that potential duplicates are placed in the same block with a high probability. Thus, instead of operating on the entire collection, it suffices to handle the data inside each block separately [6].

However, generating and processing blocks for large heterogeneous data sets imposes serious, new challenges; their inherent characteristics - i.e., loose schema binding, noise, missing or inconsistent values, as well as unprecedented levels of heterogeneity - break the fundamental assumptions of existing blocking techniques and turn them inapplicable [19]. To deal with these settings, blocking methods typically rely on *redundancy* (i.e., the common practice of placing each entity in multiple blocks); a representative example is the *attribute-agnostic blocking* approach [19], which achieves higher performance by splitting its functionality in two layers: (i) the *effectiveness layer*, which minimizes the likelihood of missed matches with the help of redundancy, and (ii) the *efficiency layer*, which employs a series of techniques to scale up the ER process by saving a large portion of the unnecessary comparisons (i.e., repeated ones or comparisons between non-matching entities) [20, 21]. Through these layers, we are able to handle large volumes of noisy and heterogeneous data, without relying on any assumptions about their schemata.

In more detail, we introduce blocking mechanisms for heterogeneous data that keep redundancy under control, without diminishing the blocking effectiveness of our attribute blocking approach.

[1]http://en.wikipedia.org/wiki/Linked_data#Principles

We actually aim at enhancing the blocking efficiency at no impact on the number of duplicates that share at least one block. In particular, we present novel, schema-independent blocking schemes that operate at a higher level of granularity, distinguishing three sources of evidence in entity profiles: (a) strings used for entity identification (typically URIs), (b) entity identifiers used for linking entities with each other (relationships), and (c) literal values used in entity descriptions. The use of these evidences in blocking significantly reduces the resulting number of required comparisons, thus offering substantially higher efficiency. However, they are less robust in isolation, since they make more assumptions on the entity profiles than attribute-agnostic blocking. Therefore, we experiment with *composite blocking schemes*, which combine the functionality of two or more - preferably complementary - individual blocking techniques in order to ensure increased robustness.

We also introduce two theoretical measures for a priori estimating the performance of a blocking method, without requiring any analytical processing of its blocks. We verify the high correlation of these metrics with the blocking effectiveness and efficiency, and demonstrate how they can be employed in the context of *block purging* [19], i.e., the process of discarding oversized blocks in order to save superfluous comparisons at a limited and controllable cost in effectiveness.

The main contributions of this paper are as follows:

1. We introduce two theoretical metrics that quantitatively capture the trade-off between blocking effectiveness and efficiency. Their values provide a reliable estimation of the performance of a blocking method, without requiring any analytical block examination. Therefore, they are suitable for a-priori estimating the best performing among a set of blocking methods.
2. We present a set of atomic blocking schemes that are crafted for large-scale ER within heterogeneous collections of data. They rely on different aspects of entity descriptions and, thus, can be combined into composite schemes of higher robustness and enhanced performance.
3. We introduce an intelligent technique for block purging that enhances the efficiency of a blocking method at a negligible and controllable cost in its effectiveness. It relies on our theoretical metrics in order to discard the excessively large blocks and save the superfluous comparisons they entail.
4. We apply our techniques on a real-world data set that comprises 182 million entities, which are described by 1.15 billion statements - the largest entity collection ever employed in the context of Entity Resolution. The overall evaluation demonstrates the utility of the proposed theoretical metrics and verifies the higher efficiency of our blocking techniques in comparison with attribute-agnostic blocking.

The rest of the paper is organized as follows: Section 2 discusses the related work, while Section 3 defines the problem. In Section 4 we propose three new blocking methods, we explain the benefits of merging them into composite ones, and we present a new algorithm for block purging. Section 5 reports the results of our experimental evaluation, while Section 6 provides conclusions and discusses future work.

2. RELATED WORK

Being a traditional problem of Computer Science, a variety of methods for ER have already been proposed in the literature. They employ a rich diversity of techniques that - among others - include string similarity metrics [4], similarity methods that rely on transformations [23], as well as relationships among entities [5]. A comprehensive overview of the most important works in this domain can be found in [6].

The most important ER blocking methods are summarized in [3]. They typically associate each record with a *Blocking Key Value* (BKV), which summarizes the values of selected attributes, and operate exclusively on it [6]. For instance, the Sorted Neighborhood approach [10] orders records according to their BKV and then slides a window of fixed size over them, comparing the records it contains. The StringMap method [14] maps the BKV of each record to a multi-dimensional, Euclidean space and employs suitable data structures to efficiently identify pairs of similar records. The *q*-grams blocking technique [7] builds overlapping clusters of records that share at least one q-gram (i.e., sub-string of length *q*) of their BKV. Canopy clustering [16] employs a cheap string similarity metric for building high-dimensional overlapping blocks, whereas the Suffix Arrays approach [24] considers the suffixes of the BKV instead. Whang et al. [25] explores another aspect of these blocking approaches, arguing 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. The same principle lies at the core of HARRA [13], a suite of LSH-based, iterative blocking algorithms that scale well in the context of homogeneous information spaces. [22] presents a principled framework for parallelizing any sound ER method at a minimum message passing cost. Note, though, that all these methods are not suitable for handling the heterogeneous entity collections of the Web of Data, due to their core assumption that entities are described by the same schema(ta), and that the quality of the individual attributes is known a priori.

In addition, the vast majority of the above methods depends on fine-tuning several application- and data-specific parameters in order to achieve their optimal performance [24]. To automate the parameter setting procedure, existing methods typically model it as a machine learning problem. For instance, [17] defines it as learning disjunctive sets of conjunctions that consist of an attribute (used for blocking) and a method (used for comparing the corresponding values). Similarly, [1] considers disjunctions of *blocking predicates* (i.e., conjunctions of attributes and methods) along with predicates combined in disjunctive normal form (DNF). These approaches, however, are only applicable to data sets involving a restricted number of distinct attributes. Otherwise, they have to consider an excessively high number of attribute combinations. They do not scale, therefore, in the context of heterogeneous information spaces.

In sharp contrast to existing approaches, attribute-agnostic blocking requires no fine-tuning, while assuming no background knowledge of the data at hand [19]. Instead, it completely disregards the schemata of the heterogeneous information spaces and solely considers the values of their entity profiles. It ensures high robustness and blocking effectiveness (i.e., the vast majority of the matching entities have at least one block in common), by creating one block for each token that appears in at least two entity profiles. Although this approach scales well to middle-sized entity collections (with few million entities), its fine granularity conveys extremely high levels of redundancy when applied to large entity collections, like the BTC09.

In this work, we argue that Heterogeneous Information Spaces allow for blocking schemes of higher granularity that involve substantially lower levels of redundancy. They are, thus, more efficient, while maintaining high levels of blocking effectiveness. In addition, they can be arbitrarily combined, forming more robust and more effective blocking methods, which are more efficient than the original attribute-agnostic blocking. To the best of our knowledge, there is no prior work on blocking techniques that can handle hundreds of millions of entities described by arbitrary schemata.

3. PROBLEM DEFINITION

To describe the problem succinctly, we follow the unifying data model that was introduced in [19]. Entity profiles are basically composed of a globally unique identifier (e.g. URIs) coupled with a set of attributes names and the values that accompany them. We assume the existence of an infinite set of attribute names $\mathcal{AN}$, along with an infinite set of possible values $\mathcal{V}$ and an infinite set of identifiers $\mathcal{ID}$.

Definition 1. *An **entity profile** $\mathbf{p_{id}}$ is a tuple $\langle id, A_{p_{id}} \rangle$, where $id \in \mathcal{ID}$ is a globally unique identifier of the profile, and $A_{p_{id}}$ is a set of attributes a_i. Each **attribute** $a_i \in A_{p_{id}}$ is a tuple $\langle n_i, v_i \rangle$, consisting of an **attribute name** $n_i \in \mathcal{AN}$ and an **attribute value** $v_i \in (\mathcal{V} \cup \mathcal{ID})$.*

The simplicity of this model enables it to accommodate entities in more complex formats, such as RDF and XML, and to represent data both in Web [26] and dataspace applications [8, 11]. For example, multiple values can be assigned to the same attribute name, while the attribute name can be a mere empty string, thus allowing for *tag-style attributes* (i.e., no associated attribute name). In addition, attribute values can contain entity identifiers, enabling the representation of relationships between entities.

Definition 2. *An **entity collection** $\mathcal{E}$ is a tuple $\langle AN_{\mathcal{E}}, V_{\mathcal{E}}, ID_{\mathcal{E}}, P_{\mathcal{E}} \rangle$, where $AN_{\mathcal{E}} \subseteq \mathcal{AN}$ is the set of attribute names appearing in it, $V_{\mathcal{E}} \subseteq (\mathcal{V} \cup \mathcal{ID})$ is the set of values used in it, $ID_{\mathcal{E}} \subseteq \mathcal{ID}$ is the set of global identifiers contained in it, and $P_{\mathcal{E}} \subseteq ID_{\mathcal{E}} \times \wp(AN_{\mathcal{E}} \times V_{\mathcal{E}})$ is the set of entity profiles that it comprises.*

3.1 Resolution through Blocking Techniques

We expect that data aggregated from diverse sources contain duplicate entity profiles, which describe the same real-world objects. Two such profiles p_i and p_j are said to *match*, a situation that is denoted as $\mathbf{p_i \equiv p_j}$. The general problem of entity resolution can, thus, be defined as follows:

Problem Statement 1 (Entity Resolution). *Given an entity collection $\mathcal{E}$ that contains duplicates, detect all matches as effectively (i.e., with high recall) and efficiently (i.e., with few entity comparisons) as possible.*

In this work, we focus on two particular cases of Entity Resolution: (i) the *Dirty-Clean ER*, where the goal is to identify the matching entities among a duplicate-free (i.e., clean) entity collection and another collection containing duplicates (i.e., dirty), and (ii) the *Dirty-Dirty ER*, which deals with the resolution of two dirty collections. Both cases are equivalent to detecting the duplicates within a single dirty entity collection that is derived from the union of the individual input data sets (i.e., *Dirty ER*). Therefore, in the following, we assume that the input to our blocking methods comprises a single set of - possibly matching - entities.

ER apparently constitutes a quadratic problem, whose naive solution (i.e., comparing each entity with all others) does not scale to large entity collections. To reduce the number of comparisons, we follow the paradigm of data blocking: our goal is to define methods that group the given set of profiles into blocks according to a blocking scheme. For our purposes, we model a blocking scheme as a combination of two functions:

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 $f_c^i : T \times T \mapsto \{true, false\}$, where T stands for the space of all possible blocking representations for entity profiles. The **transformation function** $\mathbf{f_t}$ derives the appropriate representation for blocking from the complete entity profile (or parts of it). The **constraint function** $\mathbf{f_c^i}$ is a transitive and symmetric function, encapsulating the condition that has to be satisfied by two entities, if they are to be placed in the same block b_i.*

To elucidate this definition, consider a homogeneous data collection of demographic data; a possible blocking scheme could comprise a transformation function that represents each person through her/his zip code, and a set of constraint functions that define blocks on the equality of the zip code (each constraint function corresponds to a single zip code, and, thus, to a single block). However, for heterogeneous data sets, other, less attribute dependent transformation functions are required.

Applying a blocking scheme $bs_{t,c}$ to an entity collection $\mathcal{E}$ yields a set of blocks $\mathbf{B^{\mathcal{E}}_{t,c}}$, whose instances are defined as follows:

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

The challenge in developing an efficacious blocking method - including a blocking scheme and a method for block processing - is to successfully balance the trade-off between the following two competing quality targets [1, 17, 19, 24]:

Pair Completeness: Duplicates should share at least one common block, otherwise they cannot be detected. A blocking method should, therefore, minimize the matching entities that have no block in common. This requirement is expressed through the metric of *Pair Completeness* (**PC**), which is formally defined as $PC = dm/rm$, where dm denotes the detected matches (i.e., the number of matches that share at least one block), and rm represents the number of real matches. It takes values in the interval $[0, 1]$, with higher values indicating higher *effectiveness* of the blocking method. Note that PC is comparable to the Recall metric of Information Retrieval (i.e., Blocking Recall), but is different from the Recall of the entire ER process, though having a direct impact on it.

Reduction Ratio: Each block should contain as few irrelevant entities as possible, in order to reduce the number of unnecessary pair-wise comparisons. This requirement is typically measured with respect to a baseline blocking method through the metric of *Reduction Ratio* (**RR**); this measure is defined as $RR = 1 - mc/bc$, where mc stands for the method's number of comparisons and bc for the number of baseline comparisons. RR takes values in the interval $[0, 1]$ (for $mc \leq bc$), with higher values denoting higher *efficiency* of the blocking scheme.

In this context, the formal definition of the problem we are tackling in this work is the following:

Problem Statement 2 (Blocking for Entity Resolution). *Given an entity collection $\mathcal{E}$ that contains duplicates and a baseline blocking method, cluster its entities into blocks, such that Pair Completeness and Reduction Ratio are maximized.*

Maximizing RR means that the process of Entity Resolution can be efficiently applied to large data sets, while maximizing PC satisfies the application requirements (i.e., an acceptable level of effectiveness for heterogeneous data sets).

3.2 Metric Space for Blocking Techniques

To meet the goal of high PC in the context of noisy or heterogeneous information spaces, blocking methods typically rely on *redundancy*: entities are associated with multiple, overlapping blocks, thus minimizing the likelihood of missed matches [3, 19, 24, 25]. In this way, effectiveness is significantly enhanced, though at the cost of the second target: redundancy has a substantial impact on efficiency, due to the increased number of comparisons between entities. There is, therefore, a clear trade-off between these two targets: higher blocking redundancy increases PC, but lowers RR, and vice versa.

To concretely express this general observation, we define two measures that a priori assess the performance of blocking methods. The first one - called *Blocking Cardinality* (**BC**) - captures the proportional relationship between redundancy and the number of blocks associated with each entity, on average:

Definition 5. *Given an entity collection $\mathcal{E}$ along with a blocking scheme $bs_{t,c}$, the* ***Blocking Cardinality*** *($\mathbf{BC^{\mathcal{E}}_{t,c}}$) of the resulting set of blocks $\mathcal{B}^{\mathcal{E}}_{t,c}$ is defined as the average number of blocks $b_i \in \mathcal{B}^{\mathcal{E}}_{t,c}$ an entity profile $p_{id} \in \mathcal{E}$ is placed in:*

$$BC^{\mathcal{E}}_{t,c} = \frac{\sum_{p_{id} \in \mathcal{E}} |b_i \in \mathcal{B}^{\mathcal{E}}_{t,c} : p_{id} \in b_i|}{|\mathcal{E}|} = \frac{\sum_{b_i \in \mathcal{B}^{\mathcal{E}}_{t,c}} |b_i|}{|\mathcal{E}|},$$

where $|\mathcal{E}|$ denotes the size of the given entity collection $\mathcal{E}$, and $|b_i|$ the size of block b_i.

BC takes values in the interval $[0, |\mathcal{E}| - 1]$, with higher values denoting higher levels of redundancy. A value of 1 denotes a technique that is close to a *partitioning blocking method* (i.e., one that partitions the given entity collection into non-overlapping blocks). Values lower than 1 indicate blocking methods that fail to place each entity in at least one block; this is possible, for instance, with blocking techniques that exclusively rely on a single attribute and ignore entity profiles that do not possess it. The maximum value of BC is set to $|\mathcal{E}| - 1$, since there is no point in associating every entity with all others more than once.

Another crucial factor for the performance of a blocking method is the distribution of block sizes: a large set of individually small blocks is substantially more efficient than a set of few, but extremely large blocks, even if these sets share the same BC value. To quantify this notion, we introduce the metric of *Comparisons Cardinality* (**CC**):

Definition 6. *Given an entity collection $\mathcal{E}$ along with a blocking scheme $bs_{t,c}$, the* ***Comparisons Cardinality*** *($\mathbf{CC^{\mathcal{E}}_{t,c}}$) of the resulting set of blocks $\mathcal{B}^{\mathcal{E}}_{t,c}$ is defined as the ratio between the sum of block sizes and the total number of comparisons entailed in $\mathcal{B}^{\mathcal{E}}_{t,c}$:*

$$CC^{\mathcal{E}}_{t,c} = \frac{\sum_{b_i \in \mathcal{B}^{\mathcal{E}}_{t,c}} |b_i|}{\sum_{b_i \in \mathcal{B}^{\mathcal{E}}_{t,c}} |b_i| \cdot (|b_i| - 1)/2},$$

where $|b_i|$ denotes the size of block b_i.

In essence, CC denotes the number of *block assignments* (i.e., associations between a block and an entity) that account for a single comparison. It takes values in the interval $[0, 2]$, with higher values corresponding to block size distributions that are dominated by small blocks, and, thus, are more efficient. In fact, its maximum value $CC_{max} = 2$ corresponds to the ideal case of a blocking method that associates each entity of the input collection $\mathcal{E}$ with a single block that contains just another entity: $CC = \frac{|\mathcal{E}|}{|\mathcal{E}|/2} = 2$. In contrast, for a blocking method that places all given entity profiles in a single block, we have $CC_{max} = \frac{|\mathcal{E}|}{|\mathcal{E}| \cdot (|\mathcal{E}|-1)/2} = \frac{2}{|\mathcal{E}|-1} \ll CC_{max}$. On the whole, the closer the CC is to CC_{max}, the more efficient the corresponding blocking method is.

Note that both the BC and the CC depend on the input entity collection as well as the blocking method at hand: the same blocking scheme can yield different levels of redundancy and different block size distributions, when applied to different entity collections.

The combination of these two metrics - BC and CC - captures in a comprehensive way the trade-off between effectiveness and efficiency, that is inherent in any blocking method. This trade-off can be actually visualized by mapping a blocking method onto a

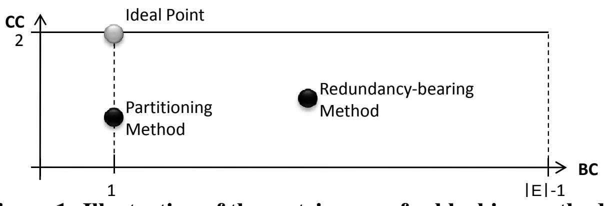


Figure 1: Illustration of the metric space for blocking methods.

Cartesian space, with its X-axis corresponding to BC values, and its Y-axis corresponding to CC values. We call it the **BC-CC space** (see Figure 1). A partitioning method has a BC value of 1, while its CC value depends on the distribution of block sizes; thus, it is mapped to some point on the $x = 1$ line. The *ideal point* (i.e., point (1,2)) corresponds to the ideal case of a partitioning method that completely disregards non-matching entities and builds a block of minimum size for each pair of duplicates. A typical redundancy-bearing blocking method is mapped to a point lying somewhere between the lines $x = 1$ and $x = |\mathcal{E}| - 1$.

In the following, we demonstrate that the conjunction of the BC and CC metrics is appropriate not only for describing blocking methods, but also for providing a reliable estimation of their actual performance. There is indeed a high correlation between the BC value of a blocking method and its effectiveness (i.e., Pair Completeness). The same holds true for CC and its relation to the efficiency of a method. The metric space of BC-CC can be used, therefore, for comparing a priori a set of blocking methods: the closer a blocking method is to the ideal point (1,2), the better its performance. The main advantage of this methodology is its negligible cost: both metrics are computed in linear time, $O(|\mathcal{B}^{\mathcal{E}}_{t,c}|)$, simply by passing over the given set of blocks in order to record their size[2]. Below we also explain how the BC-CC mapping of a blocking method can be used as a guide for further enhancements in its performance, such as in block purging.

4. BLOCKING TECHNIQUES

Our approach to blocking for large-scale heterogeneous data sets is inspired by three observations: (i) The data sets we are considering entail loose schema binding and unprecedented levels of heterogeneity. The latter pertains to the schemata describing the same entity types as well as to different profiles describing the same entity. Google Base[3], for instance, encompasses $100,000$ distinct schemata that correspond to $10,000$ entity types [15]. Consequently, these conditions strongly oppose to a crucial role of attributes in blocking and duplicate detection. (ii) Many identifiers in the Web of Data, (i.e., URIs) contain semantics, thus constituting a strong basis on which entities can be matched. This was experimentally verified in [18], where a large-scale analysis of matching entities according to the semantics of their identifiers had a precision well over 90% and a recall exceeding 70%. (iii) Relationships between entities provide important evidences for matching them and can be exploited in numerous ways. For example, after detecting a pair of duplicates, positive and negative evidences can be propagated across related entities in order to increase the precision and recall of ER [5].

Based on the above observations, we developed a set of blocking schemes that are robust enough to tolerate high schema heterogeneity, as well as inconsistencies and noise (i.e., missing or incorrect values in the entity profiles). Observation (i) inspired us

[2]This is the reason why CC considers the entire amount of comparisons, including the redundant ones: its value is derived from the plain inspection of the blocks at hand, without looking into the entities they contain.

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

	Prefix	Infix	Suffix
(a)	http://dblp.l3s.de/d2r/resource/publications/books/sp/wooldridgeV99	/ThalmannN99	
	http://bibsonomy.org/uri/bibtexkey/books/sp/wooldridgeV99	/ThalmannN99	/dblp

	Prefix	Infix	Suffix
(b)	http://liris.cnrs.fr	/olivier.aubert	/foaf.rdf#me
	http://bat710.univ-lyon1.fr	/~oaubert	/foaf.rdf#me

Figure 2: Examples of URI pairs in the PI(S) form that refer to the same real-world entity.

to develop techniques that completely ignore the schema information and exclusively rely on the values of entity profiles [19]; all blocking schemes that are presented in the following adhere to this principle. Observation (ii) advocates the creation of blocks on the basis of evidence drawn from the semantics of entity identifiers (Blocking Scheme A.1). In view of observation (iii), we additionally consider blocking schemes that rely on the relationships between entities, exploiting the semantics contained in the identifiers of affiliated resources (Blocking Scheme A.2).

4.1 Preliminaries: Entity Identifier Analysis

Even though the *W3C* explicitly discourages users from incorporating semantics into URIs [12], the latter often contain substrings that provide clues about the corresponding entity. These substrings usually stem from well-established, often hierarchical, human labeling schemes (e.g., names, titles, or addresses), or explicit standardization efforts (e.g., industry-wide product codes or digital object identifiers - DOI). Hence, they constitute a natural source of evidence appropriate for matching duplicate entities. Indeed, [18] verified experimentally that approximately 66% of the 182 million URIs of the BTC09 data set[4] follow a common pattern: the *Prefix-Infix(-Suffix) scheme* - PI(S) for short. Each component of this form plays a special role: the *Prefix* part contains information about the source (i.e., domain) of the URI, the *Infix* part is a sort of local identifier, and the optional *Suffix* part contains either details about the format (e.g., `.rdf` and `.n3`), or a named anchor.

Our approach builds upon this pattern in order to extract matching evidences from entity identifiers. We actually expect the Infixes of URIs, which are more *source-independent* than the Prefixes and the Suffixes, to contain the most discriminative information for our purpose within a URI [18]. As an example, consider the duplicate pairs shown in Figure 2. Pair (a) shows that despite the high heterogeneity in the Prefixes of the URIs, the Infix remains the same. Moreover, the Suffix (e.g., `dblp`) is optional and can be ignored when matching URIs. Pair (b) illustrates the limitations of the approach: non-identical Infixes such as (`olivier.aubert` and `~oaubert`) can only be matched with the help of appropriate similarity methods. In the following, we explain how this situation can be resolved through blocking methods that combine evidence from different aspects of entity profiles.

4.2 Atomic Blocking Schemes

Given an entity profile p_{id}, our blocking schemes disregard all schema information (i.e., attribute names), and rely exclusively on the remaining description items that are illustrated in Figure 3: the identifier *id* of the profile (its Infix, in particular), the URIs contained in p_{id} as values for denoting entity relationships, and the *literal values*[5] that are used to represent textual values of attributes. Blocking methods that rely solely on one category of the aforementioned information are **atomic blocking schemes**, in contrast to **composite blocking schemes**, which consider evidence from two or more of the above information sources.

To apply our blocking schemes to a collection of RDF data, we generate the entity profiles from its statements in line with Definition 1: all triples with a particular subject s, form the basis for an entity profile p_s that has s as its identifier, while the respective predicates and objects correspond to the set A_{p_s} of the name-value pairs of p_s.

In the following, we formally define our blocking schemes A.1 to A.3. For this purpose, we use two functions: $Infix(id)$[6], which extracts the Infix from the given entity identifier *id*, and *tokenize*(v), which tokenizes a string value v on all special characters (i.e., characters that are neither letters nor digits).

A.1. Infix Blocking. This blocking scheme conveys a *transformation function* that extracts the Infix from the *id* of each entity profile: $f_t(p_{id}) = Infix(id)$. Its *constraint functions* define blocks on the equivalence of Infixes: every block is associated with an Infix and contains all entities that share it. More formally, $f_c^i(p_{id_1}, p_{id_2}) = ((i = Infix(id_1)) \wedge (i = Infix(id_2)))$, where i is an Infix.

Apparently, this blocking scheme places each entity in just one block, thus resulting in non-overlapping blocks, i.e., $BC \leq 1$. For example, the entity of Figure 3, is placed solely in the block that corresponds to the Infix "Barack_Obama", together with all other entities sharing the same Infix. Values lower than 1 denote that the given entity collection contains profiles, whose URIs lack an Infix. The absence of redundancy together with the discriminative information of Infixes ensure blocks of small size and high CC values (i.e., high Reduction Ratio). However, its blocking effectiveness as well as its robustness is expected to be limited, as a side-effect of its low BC value. For instance, it does not apply to entities with a blank node[7] or a random URI as their *id*; the reason is that, in these cases, the corresponding identifier has merely a local scope, thus not providing a meaningful Infix for entity matching.

A.2. Infix Profile Blocking. The *Infix Profile* of an entity profile p_{id} - denoted by $IP_{p_{id}}$ - is the set of the Infixes of all URIs contained in p_{id} as attribute values, with the exception of its own identifier (i.e., *id*): $IP_{p_{id}} = \{Infix(id_i) : \exists n_i :< n_i, id_i > \in A_{p_{id}} \wedge id_i \neq id\}$.

Based on this concept, we can define a blocking scheme with a *transformation function* that represents each entity profile p_{id} by its $IP_{p_{id}}$ (i.e., $f_t(p_{id}) = IP_{p_{id}}$), and a set of *constraint functions* that form blocks on Infix equality: $f_c^i(p_{id_1}, p_{id_2}) = ((i \in IP_{p_{id_1}}) \wedge (i \in IP_{p_{id_2}}))$, where i is an Infix.

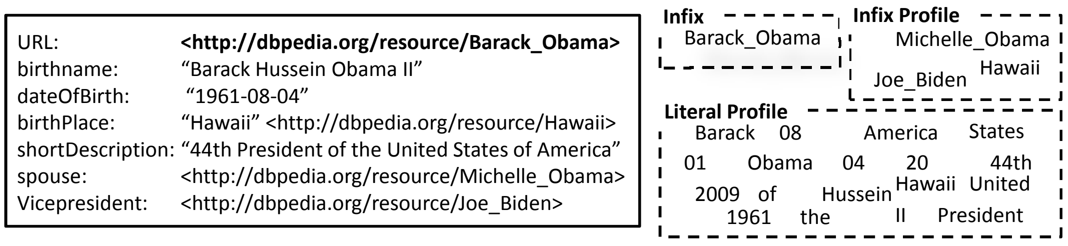
The cardinality of an $IP_{p_{id}}$ can be larger than one, since an entity is typically associated with multiple entities. Hence, every entity with a non-empty Infix Profile can be placed in multiple blocks, which are now overlapping (i.e., $BC \geq 1$ if all entities have a non-empty Infix Profile). For example, the entity of Figure 3, is contained in three blocks: those corresponding to the Infixes "Michelle_Obama", "Hawaii", and "Joe_Biden". This redundancy leads to larger blocks - on average - and inevitably to a lower CC value than Infix Blocking. RR is, therefore, lower to the benefit of higher PC and robustness. This blocking scheme has actually the potential to cover duplicates, where pure Infix Blocking is inappli-

[4]http://vmlion25.deri.ie.

[5]In RDF statements, these are objects that are neither URIs nor blank nodes.

[6]See [18] for an efficient approach to splitting a collection of URIs into the PI(S) form in order to extract the Infix from the individual URIs.

[7]Blank nodes constitute anonymous nodes in an RDF graph that are typically used whenever there is no information available about the corresponding resource. Consequently, their identifiers do not carry any semantics.



URL:	**<http://dbpedia.org/resource/Barack_Obama>**
birthname:	“Barack Hussein Obama II”
dateOfBirth:	“1961-08-04”
birthPlace:	“Hawaii” <http://dbpedia.org/resource/Hawaii>
shortDescription:	“44th President of the United States of America”
spouse:	<http://dbpedia.org/resource/Michelle_Obama>
Vicepresident:	<http://dbpedia.org/resource/Joe_Biden>

Figure 3: Illustration of the description items of an entity profile that are used by our blocking schemes.

cable: even though blank nodes and numerical URIs have no Infix, their Infix Profile can be non-empty. The same applies to matching entities with non-identical Infixes: their Infix Profile is likely to share at least one Infix of a related entity. Regarding its mapping to the BC-CC space, this schema is placed to the right of A.1 on the X-axis and lower than it on the Y-axis (assuming that both schemes are applied to the same entity collection).

However, the coverage of this strategy is also limited: entities with numerical/arbitrary URIs tend to be related to other numerical/arbitrary URIs of the same domain, thus lacking an Infix Profile, unless they are used in multiple domains. Moreover, it cannot cover the cases of entity profiles that lack an Infix Profile (e.g., profiles that solely contain literal values).

A.3. Literal Profile Blocking. The Literal Profile of an entity profile p_{id} - denoted by $LP_{p_{id}}$ - comprises the set of all tokens of the literal values contained in it: $LP_{p_{id}} = \{t_i : \exists n_i, v_i :< n_i, v_i >\in A_{p_{id}} \wedge t_i \in tokenize(v_i) \wedge v_i \notin \mathcal{ID}\}$.

This textual information can be employed in the context of a blocking method that has the following *transformation function*: $f_t(p_{id}) = LP_{p_{id}}$. Blocks are based on the equality of tokens, with each block corresponding to a single token and each entity associated with multiple blocks. For instance, in Figure 3 we can see that the depicted entity will be placed in 18 blocks, one for each token in its Literal Profile. More formally, the *constraint functions* are defined as follows: $f_c^{t_i}(p_{id_1}, p_{id_2}) = ((t_i \in LP_{p_{id_1}}) \wedge (t_i \in LP_{p_{id_2}}))$, where t_i is a token.

Note that this blocking scheme is similar to that presented in [19] with the difference that it does not tokenize the URIs contained in entity profiles as values. It is expected, therefore, to be inapplicable to cases where a profile has no literal values. Still, PC is potentially higher than that of A.1 and A.2, as the only prerequisite for an eligible entity profile is to contain at least one literal value that is shared with other entities. However, its efficiency (i.e., RR) is expected to be lower, due to the higher degree of redundancy: each block corresponds to a single token, with the number of entities sharing a token being typically higher than those sharing an Infix (Infixes normally consist of several tokens concatenated together, and are less common than individual tokens). This results in blocks that are larger, on average, thus reducing the value of CC. Its mapping to the BC-CC space is placed to the right of A.1 on the X-axis and lower than it on the Y-axis (assuming that both schemes are applied to the same entity collection).

4.3 Enhancing Effectiveness: Composite Blocking Schemes

In isolation, the individual atomic blocking schemes are of limited robustness, with their effectiveness depending upon the characteristics of the entity collections at hand. To remedy this situation, we propose the combination of atomic blocking schemes into composite ones. Given that the individual blocking schemes rely on different aspects of entity profiles, they are *complementary*. Thus, the composite blocking schemes derived from their union are expected to exhibit significantly higher robustness, leading to significantly higher PC. This is at the cost of efficiency, however, since the new scheme entails more blocks that are larger in size, due to the common constraint functions (i.e., those stemming from different atomic blocking schemes, but corresponding to the same token or Infix).

In Figure 4, we illustrate the effect of merging atomic blocking schemes into composite ones on the BC-CC space: combining $Method_1$ with $Method_2$ leads to $Method_3$ that has increased BC value (i.e., higher redundancy) and lower CC value (i.e., restricted efficiency), due to the larger - on average - blocks.

In the following paragraphs, we present all four possible combinations of the above atomic schemes and explain the rationale behind them. Note that all of them produce overlapping blocks.

B.1. Complete Infix Blocking. This blocking scheme is derived from the combination of the schemes A.1 and A.2. Thus, the *transformation function* extracts from an entity profile the union of its Infix and its Infix Profile: $f_t(p_{id}) = Infix(id) \cup IP_{p_{id}}$. Blocks are again built on the equivalence of Infixes, with each block corresponding to a single Infix and each entity potentially placed in multiple blocks: $f_c^i(p_{id_1}, p_{id_2}) = ((i = Infix(id_1) \vee i \in IP_{p_{id_1}}) \wedge (i = Infix(id_2) \vee i \in IP_{p_{id_2}})$, where i is an Infix.

Compared to Infix Blocking, it covers profiles with a synthetic identifier (i.e., a blank node or a random URI) and is able to match entities with non-identical Infixes. Compared to Infix Profile Blocking, it applies to entities with empty Infix Profiles which, nevertheless, have an Infix. The only case that this scheme is not applicable is for entities that lack any meaningful URI in their profile; these are entities that have arbitrary, synthetic URIs as ids and are solely associated with identifiers of the same kind as well as with literal values. Most importantly, though, this combination takes advantage of Infixes that originally did not result in blocks, due to their scarcity in the individual blocking schemes (i.e., they appeared in just one profile). As verified in Section 5, this results in significantly higher robustness that brings about higher PC, as well.

B.2. Infix-Literal Profile Blocking. This scheme results from the combination of A.1 with A.3: its *transformation function* represents each entity by the union of its Infix and its Literal Profile: $f_t(p_{id}) = Infix(id) \cup LP_{p_{id}}$, while its *constraint functions* build blocks on the equality of Infixes or tokens: $f_c^i(p_{id_1}, p_{id_2}) = ((i = Infix(id_1) \vee i \in LP_{p_{id_1}}) \wedge (i = Infix(id_2) \vee i \in LP_{p_{id_2}}))$, where i is an Infix and/or a token.

It is worth noting that the merge of these two atomic schemes is more than just the union of the blocks of the individual schemes: some Infixes merely consist of a single token, leading to some constraint functions that are common among both schemes (e.g., the Infix “Hawaii” of the entity in Figure 3 appears as a token of a literal value, as well). Thus, this composite scheme involves blocks common among the atomic ones; these blocks have a larger size than before, entailing more comparisons and resulting in lower efficiency and lower CC values. However, robustness and effectiveness are substantially enhanced: the only profiles that are not covered by this composite scheme are those having a blank node or a random URI as id, while containing no literal values in their set of attribute values. Apparently, this case is highly unlikely.

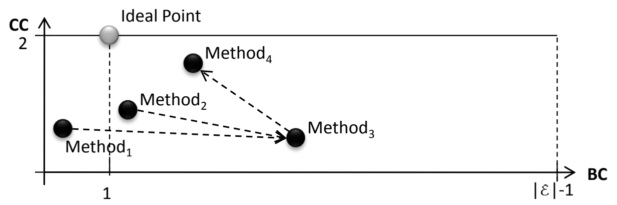


Figure 4: Illustration of the effect on the BC-CC space of two optimization techniques: (i) merging individual blocking schemes ($Method_1$ and $Method_2$) into composite ones ($Method_3$), and (ii) purging the oversized blocks of a blocking technique ($Method_3$) to produce a more efficient one ($Method_4$).

B.3. Infix Profile-Literal Profile Blocking. This scheme is derived from the combination of the schemes A.2 and A.3. Similar to B.2, the *transformation function* extracts from an entity profile the union of its Infix and its Literal profile (i.e., $f_t(p_{id}) = IP_{p_{id}} \cup LP_{p_{id}}$), while the *constraint functions* define blocks on the equality of Infixes or tokens: $f_c^i(p_{id_1}, p_{id_2}) = ((i \in IP_{p_{id_1}} \vee i \in LP_{p_{id_1}}) \wedge (i \in IP_{p_{id_2}} \vee i \in LP_{p_{id_2}}))$, where i is an Infix and/or a token.

For the reasons explained in B.2, it exhibits higher redundancy (i.e., higher BC value) than the individual blocking schemes it comprises, thus involving higher robustness coupled with higher effectiveness. The main difference between B.2 and B.3 lies in the cases that B.3 does not cover: this scheme does not apply to profiles whose set of attribute values contains solely blank nodes and random URIs.

B.4. Total Description Blocking. This blocking scheme is formed by the combination of all atomic ones. Thus, it represents entities by the Infixes of all URIs contained in their profiles and the tokens of all their literal values: $f_t(p_{id}) = Infix(id) \cup IP_{p_{id}} \cup LP_{p_{id}}$. Blocks are then defined on the equality of tokens or Infixes: $f_c^i(p_{id_1}, p_{id_2}) = ((i = Infix(id_1) \vee i \in IP_{p_{id_1}} \vee i \in LP_{p_{id_1}}) \wedge (i = Infix(id_2) \vee i \in IP_{p_{id_2}} \vee i \in LP_{p_{id_2}}))$, where i is an Infix and/or a token.

The interplay of all atomic blocking methods leads to the highest robustness among all schemes presented in this paper: this combination fails only in the highly unlikely cases where an entity profile has a non-meaningful identifier as its id and exclusively contains identifiers of this kind in its set of attribute values. This results not only in the highest effectiveness among all methods, but also in the highest BC value: it entails the highest number of blocks, which are also expected to be larger - on average - than the blocks of the other schemes. They entail, therefore, a considerably higher number of comparisons that leads to lower efficiency and lower CC values.

4.4 Enhancing Efficiency: Block Purging

BC and CC are useful not only for assessing the performance of blocking techniques, but also for improving it. In fact, they can be employed in the context of *block purging*, a method introduced in [19] as a way of increasing the efficiency of redundancy-bearing blocking methods. In essence, it is based on the observation that very large blocks entail a high cost (in terms of comparisons), although they have a negligible contribution to Pair Completeness (i.e. it is very unlikely that they contain duplicates that have no other block in common). Based on this observation, Block Purging specifies an upper limit on the size of the blocks that are processed, and discards all blocks that exceed it.

The effect of this approach on the BC-CC space is illustrated in Figure 4 by $Method_3$ and $Method_4$. With the removal of any block, the overall number of block assignments is reduced together with Blocking Cardinality (as its denominator remains stable); thus, the BC-CC mapping of a blocking method moves to the left on the X-axis. On the other hand, CC increases, as the removal of large blocks leads to a more even distribution of block assignments across the remaining collection of blocks. In the general case, therefore, block purging moves the original method ($Method_3$) to the direction of $Method_4$ (i.e., closer to the ideal point (1,2)).

Algorithm 1: Tuning for Block Purging.

```
   Input:  B a set of blocks
   Output: optimalBlockSize the optimal, maximum block size for block purging
1  $B' \leftarrow$ orderBySize($B$);
2  $blockAssignments \leftarrow 0$;
3  $comparisons \leftarrow 0$;
4  $lastBlockSize \leftarrow 2$;
5  $index \leftarrow 0$;
6  $statistics[] \leftarrow \{\}$;
7  foreach $b_i \in B'$ do
8      if $lastBlockSize < b_i.size()$ then
9          $statistics[index]$.size = $lastBlockSize$;
10         $statistics[index]$.cc = $\frac{blockAssignments}{comparisons}$;
11         $index$++;
12         $lastBlockSize \leftarrow b_i$.size();
13     $blockAssignments \leftarrow blockAssignments + b_i$.size();
14     $comparisons \leftarrow comparisons + \frac{b_i.size() \cdot (b_i.size()-1)}{2}$;
15 $statistics[index]$.size = $lastBlockSize$;
16 $statistics[index]$.cc = $\frac{blockAssignments}{comparisons}$;
17 $optimalBlockSize \leftarrow lastBlockSize$;
18 for $i \leftarrow statistics.size()-1$ to $1$ do
19     if $statistics[i].cc \approx statistics[i-1].cc$ then
20         $optimalBlockSize \leftarrow statistics[i]$.size;
21         break;
22 return $optimalBlockSize$;
```

The key for successfully applying this technique is the selection of the upper limit on block sizes. A simple method for this task was introduced in [19], but it is crafted for the Clean-Clean case of ER and is not applicable to the Dirty ER we are considering. To replace it, we introduce a novel method for selecting the optimal threshold on the maximum block size that relies on the CC metric. It is based on the following observation: starting with the exclusion of the largest block and going down towards the smaller ones, the value of Comparisons Cardinality slowly increases. This is because the removal of large blocks (i.e., blocks with $|b_i| \gg 2$) decreases its denominator (i.e., number of comparisons - $\sum_{b_i \in B} |b_i| \cdot (|b_i|-1)/2$) faster than its numerator (i.e., number of block assignments - $\sum_{b_i \in B} |b_i|$). Block purging should stop, therefore, as soon as there is no more increase in the value of CC to be expected; discarding additional blocks would just reduce BC and, consequently, PC, while having a negligible effect - if any - on RR. For this reason, we use the following termination criterion: block purging stops as soon as two consecutive block sizes have the same value for CC.

The outline of our approach to Block Purging is presented in Algorithm 22. Line 1 orders the given collection of blocks B in ascending order of block sizes, and, thus, makes it possible to calculate the Comparisons Cardinality value for each block size with a single pass (Lines 2-19). Lines 15-16 ensure that the last block is also taken into account in the computation of the statistics. Starting from the largest block size, the CC values of consecutive sizes are then compared in Lines 17-22. The procedure is terminated as soon as the CC value remains stable. Apparently, the complexity of our algorithm is dominated by the initial sorting and is equivalent to $O(|B| \log |B|)$, where $|B|$ stands for the size of the given collection of blocks B.

5. EVALUATION

5.1 Setup

Baseline. Previous studies have shown that — in the context of heterogeneous information spaces — existing blocking methods, which rely on schema information, exhibit high efficiency (i.e., very few entities per block), but they suffer from remarkably poor recall (i.e., PC): they place less than half the matching entities into (at least) one common block [19]. In this paper, we do not repeat the comparison experiments to such blocking methods. Instead we use as baseline for our experiments the original attribute-agnostic blocking method of [19], which outperforms previous techniques. It is denoted by **Total Tokenizer** and essentially works as follows: given an entity profile, this blocking scheme tokenizes all the values it contains on their special values and defines blocks on the equality of tokens. The only values that are excluded from this procedure are the URIs that correspond to blank nodes. As stated above, the goal is to prove that the blocking schemes presented in this paper provide a better balance between PC and RR: they maintain blocking effectiveness at equally high levels, while requiring a significantly lower amount of pair-wise entity comparisons.

Performance Metrics. To assess our blocking techniques, we employ the two metrics that have been established in the blocking literature and were introduced in Section 3.1: Pair Completeness and Reduction Ratio. The former estimates the ratio between the true matches that share at least one block and the true matches in the entire data set; the latter measures the reduction in the number of pair-wise comparisons the examined blocking method achieves with respect to the baseline one. Note that we follow the literature [1, 17, 19, 24], and focus on the performance of the blocking methods, thus ignoring the precision of entity matching techniques. The latter is actually out of the scope of this paper, as the proposed schemes can be integrated with any entity comparison method.

Technical Metrics. The second group of metrics we consider elucidates important technical characteristics of blocking methods, providing the grounds for their measured performance. The considered measures are the following: **(i)** *Method Coverage* denotes the portion of the given entities that qualify for the respective blocking method, **(ii)** *Blocks Coverage* expresses the percentage of entities that are placed in at least one block. In conjunction with Method Coverage, it estimates those entities that qualify for the blocking scheme, but share no description item(s) with any other entity[8], **(iii)** *Disk Space* occupied on the hard drive, **(iv)** *Average Block Size* denotes the average entities per block, **(v)** *Number of Blocks* generated by the technique.

The first two measures are indicative of the robustness of a blocking scheme, with higher values corresponding to more robust methods. The remaining ones are related to the efficiency aspects of a method (including storage efficiency), with higher values corresponding to lower efficiency; for example, the larger the blocks of a blocking method are on average, the more comparisons they entail and the lower its efficiency is.

Large-scale Data Set. We evaluated our approaches on the real-world data set of BTC09, the largest one ever used in the context of Entity Resolution. It comprises more than 182 million distinct entities that are described by 1.15 billion RDF statements. They have been crawled from several thousand sources of the Semantic Web, each having its unique characteristics for the format and the quality of the contained information. As a result, BTC09 constitutes a sizeable and representative heterogeneous information space, which is suitable for deriving safe conclusions about the generality of our blocking techniques.

As ground-truth we employed two collections of matching entities: the first one - denoted by **SameAs** - was derived from the explicit `owl:sameAs` statements and encompasses 5.99 million matches of the form $p_i \equiv p_j$ that, in total, involve 8.67 million distinct entities. The second ground-truth set - symbolized as **IFP** - was inferred from the implicit equivalence relationships of the InverseFunctionalProperties[9] and contains $11,553$ matches among $17,990$ distinct entities.

The reason for considering two sources of ground-truth is the bias that is perhaps lurking in the explicit equivalence relationships: it is possible that some of them consist of machine-generated same-as statements, which typically follow specific URI patterns (i.e., they change solely the Prefix and the Suffix of a URI, leaving the Infix intact). This is the case, for instance, with the transformation of Wikipedia URIs to DBPedia ones. To have a better understanding of the general performance and the robustness of our algorithm, we need, therefore, an additional data set that involves a higher variety of equivalence relationships, derived from a rich diversity of sources. The ground-truth set of implicit equivalence relationships (i.e., IFP) serves this need perfectly.

5.2 Blocking Schemes Performance

Technical Metrics. The technical metrics of our thorough experimental comparisons are presented in the left side of Table 1. Regarding Method Coverage, we can see that the atomic blocking schemes individually cover less than 2/3 of all entities, while their combinations have substantially higher coverage - well over 90% in most of the cases. Nevertheless, the coverage of Infix and Infix Profile blocking alone is larger than one would expect. To explain this phenomenon, we investigated the extent to which blank nodes are used as ids, not only for uniquely identifying entities, but also for expressing the associations between entities. We found out that, among all data sources of BTC09, less than a third of their entities (32.61%) have a blank node as an id, and a mere 4.99% of their statements have a blank node as their object. Consequently, blank nodes are completely outweighed by real URIs, and have a restricted impact on the applicability of our method.

The values of Blocks Coverage follow the same pattern as those of Method Coverage: they are slightly lower than 66% for atomic blocking schemes, but significantly higher for the composite ones. It is worth noting that - in almost all the cases - the value of Blocks Coverage is around 2% lower than that of Method Coverage. This means that our blocking schemes place almost all entities that contain the required description item(s) in at least one block. The only exception to this is Infix Blocking: there is a discrepancy of 36% between its Blocks and its Method Coverage. This is caused not only by the arbitrary entity identifiers (i.e., random or numerical URIs), but also by the scarcity of Infixes (i.e., a considerable part of Infixes appears in just one entity identifier, thus not forming a block).

It is interesting to examine why this does not apply to Infix Profile Blocking, as well. A possible reason is the connectivity of the resources contained in BTC09; in fact, it was reported in [18] that 12.55% of the resources appear only as subjects, 30.49% appear solely as objects, while the rest (56.96%) appear both as subjects and objects. This evidence suggests that there are strong connections between the nodes of the RDF graph and, consequently, many relations between the entities of different domains. Therefore, sources with synthetic URIs are connected to other domains that do not necessarily follow the same methodology for generating identifiers. Thus, they pose no serious threat to the robustness of Infix Profile blocking and the blocking schemes built on it.

The results on the efficiency characteristics are quite intuitive:

[8]Note that each block has to contain at least 2 entities.

[9]InverseFunctionalProperties (IFPs) provide a reliable means of discovering *implicit* equivalence relationships in the Semantic Web: *any two resources that share the same value for an IFP are, actually, identical.*

	Technical Metrics					Performance Metrics				
Blocking Scheme	**Method Coverage**	**Blocks Coverage**	**Disk Space** (GB)	**Blocks** ($\times 10^6$)	**Av. Block Size**	**RR**	**PC_{IFP}** (%)	**PC_{SameAs}** (%)	**BC**	**CC** ($\times 10^{-7}$)
Infix	67.20%	31.32%	13	15.34	3.83	99.98	59.31	49.60	0.32	94.20
Infix Profile	66.17%	64.84%	26	23.59	17.80	95.86	84.19	43.78	2.30	3.92
Literal Profile	55.63%	54.95%	59	17.83	111.39	89.68	94.49	24.51	10.90	7.43
Complete Infix	98.91%	97.25%	33	68.87	8.13	95.77	85.16	87.14	3.07	5.10
Infix-Literal Pr.	91.88%	74.18%	65	27.90	74.07	87.46	96.43	65.67	11.34	6.36
Infix Pr.-Literal Pr.	69.29%	68.13%	75	38.44	62.14	85.30	96.64	52.09	13.11	6.61
Total Description	99.66%	98.90%	83	81.08	31.05	84.96	97.98	91.13	13.82	6.13
Total Tokenizer	100.00%	100.00%	114	31.16	112.45	-	99.32	92.26	19.23	1.35

Table 1: General statistics and performance of the proposed blocking schemes in comparison with the baseline method.

composite blocking schemes have more blocks, which are larger in size (on average) and occupy more disk space. Their efficiency is, therefore, lower than that of the atomic schemes. The baseline method is the least efficient, as it requires more than 25% additional disk space and has the largest average block size. This is a side-effect of its low-granularity, which results in rather frequent blocking units (e.g., most of the URIs contain the token "http"). This also explains why the average size of its blocks is only comparable to that of Literal Profile: they both rely on the same blocking units. However, the latter involves a lower number of tokens and can be combined with the other atomic blocking schemes to produce blocks of smaller - on average - size, thus increasing its efficiency.

Performance Metrics. The actual performance of our blocking schemes is presented on the right side of Table 1. We notice that RR takes values over 85% in all cases (there is no RR value for Total Tokenizer, since it is the baseline against which RR is computed). This means, the Total Tokenizer performs an order of magnitude more comparisons than those required by the other methods. In absolute values, the former involves 2.59×10^{16} comparisons, while the comparisons for our methods range from 6.24×10^{12} (Infix Blocking) to 3.90×10^{15} (Total Description); that is, the required pair-wise comparisons per entity drop from 10^7 to a value between 10^4 and 10^6 (we further reduce these numbers to a significant extent in Section 5.4). The very low efficiency of Total Tokenizer is depicted in its CC value as well: it is the lowest by far, while Infix Blocking has the highest one - almost an order of magnitude higher.

Regarding effectiveness, we can see that - independently of the ground-truth set - PC is quite low for the atomic blocking schemes, but substantially higher for the composite ones. This pattern actually constitutes a strong evidence in favor of merging complementary blocking schemes: their PC is always considerably larger than the maximum value of the individual schemes comprising them. For this reason, the highest PC is achieved by the method that combines all three atomic schemes (i.e., *Total Description*), being lower than Total Tokenizer by just 1%, for both benchmark sets. Note also that there is a clear association between the values of PC and BC, with high BC values conveying high PC ones. This interesting association is analytically discussed in the following section.

Overall, we note that the atomic blocking schemes exhibit the highest levels of efficiency, but suffer from deficient robustness and effectiveness. The composite blocking methods improve on both of these weaknesses, without a significant sacrifice of their efficiency. In fact, the method that encompasses all atomic schemes exhibits similar levels of robustness and effectiveness with Total Tokenizer, while saving almost 85% of all comparisons. Thus, it achieves our goal by having two crucial differences from the baseline: first, it considers the Infix of URIs that appear as attribute values, instead of extracting all tokens from them, and second, it exploits the URIs that are used as entity identifiers, extracting their Infixes, as well.

5.3 BC-CC mapping vs Real Performance

In this section, we experimentally verify that the combination of BC and CC metrics provides a highly accurate estimation of a blocking method's real performance. The values of BC actually exhibit a high correlation with those of PC, while the same applies for CC and RR. Our analysis relies on the *Pearson correlation coefficient* ($\rho_{X,Y}$), a well established measure for estimating the linear dependency between two variables X and Y. It takes values in the interval $[-1, 1]$, and the higher its absolute value is, the stronger the correlation of the given variables is. A value of $|\rho_{X,Y}| = 1$ indicates a completely linear relationship of the form $X = \alpha \cdot Y + \beta$, where $\alpha, \beta \in \mathcal{R}$ and $0 < \alpha$ if $\rho_{X,Y} = 1$, while $\alpha < 0$ if $\rho_{X,Y} = -1$.

Our analysis consists of two parts: first, we measure the correlation between the above metrics with respect to different blocking schemes. We also investigate to which extent the BC and CC metrics can identify the best performing among a set of blocking methods; to this end, we examine the relation between a blocking method's distance from the ideal point of the BC-CC space (i.e., (1,2)) and its deviation from the optimal real performance (i.e., PC=100% and RR=100%). Second, we analyze the same correlations in the context of block purging, i.e., with respect to the same blocking scheme, for various thresholds on the maximum size of the blocks. Our goal is to demonstrate the usefulness of BC and CC for determining the optimal cutting point for this process.

For the first case, Pearson's ρ can be derived directly from the values in the right part of Table 1. We have: $\rho_{PC_{IFP},BC} = 0.84$, $\rho_{PC_{SameAs},BC} = 0.37$, and $\rho_{RR,CC} = 0.63$. We can see that BC accurately predicts the value of PC with respect to IFP, but this applies to a minor extent for the SameAs ground-truth set. In both cases, however, there is a positive correlation between these two metrics, with an average value of $\rho_{\bar{PC},BC} = 0.61$. This is sufficiently high for a good and valuable estimation, given that BC can be easily derived from a simple inspection of the blocks at hand. The same applies to the correlation between CC and RR. We can, thus, deduce that BC and CC can discern the best technique among a set of blocking methods with a relatively high accuracy.

This conclusion is also advocated by the positive correlation between a blocking method's deviation from its optimal performance and its distance from the ideal point of the BC-CC space. To quantify this correlation, we consider the **PC-RR space**: a two-dimensional space defined by PC on the X-axis and RR on the Y-axis. Apparently, the optimal point in this space is (100,100), which corresponds to the perfect blocking efficiency and effectiveness. Based on the numbers of Table 1, the correlation between the distances from the ideal points of the PC-RR and the BC-CC space was found to be $\rho_{IFP} = 0.49$ and $\rho_{SameAs} = 0.45$ with respect to the IFP and the Same-As ground-truth sets, respectively. There is, therefore, an indisputably positive correlation between these distances, which is high enough (0.47 on average) for discerning between the low and the high performing blocking schemes.

To evaluate the functionality of the BC-CC mapping in the context of block purging, we systematically applied various upper limits on the block size of all blocking schemes and estimated the resulting PC and BC values (we ignored the efficiency metrics CC

	$\rho_{\text{PC}_{\text{IFP}},\text{BC}}$	$\rho_{\text{PC}_{\text{SameAs}},\text{BC}}$
Infix	0.99	0.76
Infix Profile	0.97	0.82
Literal Profile	0.67	0.89
Complete Infix	0.97	0.73
Infix-Literal Pr.	0.71	0.67
Infix Pr.-Literal Pr.	0.75	0.66
Total Description	0.77	0.56
Total Tokenizer	0.68	0.93

Table 2: Pearson Correlation between PC and BC.

and RR, since RR is at nearly 100% for all block sizes due to the high reduction). To ensure a large variety of block sizes, we determined the maximum block size according to the following formula: $|b_{i_{max}}| = 10^{\log|\mathcal{E}|/d_i}$, where $|b_{i_{max}}|$ is the maximum block size, $|\mathcal{E}|$ is the size of the input entity collection, and d_i is an integer that takes all values in the interval $[1, 10]$. The outcomes of our analysis are presented in Table 2. The average correlation of BC and PC is $\rho_{\bar{PC_{IFP}},BC} = 0.83$ and $\rho_{\bar{PC_{SameAs}},BC} = 0.73$, with respect to the IFP and to the SameAs ground-truth sets, respectively. On the average case (across both ground-truth sets), it is equal to $\rho_{\bar{PC},BC} = 0.78$. These high values lead to the safe conclusion that the BC metric accurately determines whether a cutting point for block purging has a significantly negative impact on Pair Completeness. In this case, a higher value for the block size threshold is required.

5.4 Block Purging Performance

To check the performance of Algorithm 22, we applied it to all the aforementioned blocking schemes, and the outcomes are presented in Table 3. Note that, instead of RR values, we present the absolute number of pair-wise comparisons, because RR had the maximum value (i.e., 100%) for all blocking schemes. The required number of comparisons actually dropped by four orders of magnitude in all cases (e.g., from 10^{12} to 10^8 for Infix blocking). At the core of this substantial improvement lies the restriction imposed by block purging on the maximum block sizes: from several millions entities they were reduced to several thousands of entities.

Most importantly, though, we can see that block purging achieves a better balance between Pair Completeness and Reduction Ratio: although the latter is greatly enhanced, there is a negligible decrease in the former; the values of PC are lower than in Table 1 by just 1% or 2%, in most of the cases. The only exceptions to this rule are the Infix Profile and the Complete Infix with respect to the IFP, and the Total Tokenizer with respect to the SameAs golden standard. This is a strong indication of limited robustness for the corresponding methods, as their initially high effectiveness extensively depends on their oversized blocks. In complete contrast, Total Description is the only one that retains its PC (i.e., around 90%) for both ground-truth sets, thus constituting the most robust of the proposed blocking schemes. Note also that our experimental results verify the behavior of block purging on the BC-CC space, as demonstrated in Figure 4: the BC values move to the left of the X-axis, while the CC values move higher on the Y-axis.

6. CONCLUSIONS

We introduced novel blocking schemes tailored for heterogeneous data. Individually, they achieve high efficiency, but suffer from low robustness. To enhance it, we combine them into composite, more effective blocking methods and discard the largest of their blocks, thus boosting their efficiency as well. The trade-off between Pair Completeness and Reduction Ratio led to the idea of introducing two blocking scheme assessment metrics (BC and CC), which are easily derived from the external characteristics of a set of blocks. Their values are highly correlated with the actual performance of a blocking method and provide a quite accurate, a priori estimation for it. Our techniques were thoroughly tested on the largest, real-world data collection ever used for this task, and the outcomes verified the usefulness of our metrics The main conclusion is that combining Total Description with Block Purging yields a blocking method that excels in all aspects of performance (i.e., robustness, efficiency and effectiveness). In the future, we intend to investigate ways of parallelizing the required pair-wise comparisons, based on the MapReduce paradigm.

	Comp.	**PC$_{\text{IFP}}$**	**PC$_{\text{SameAs}}$**	**BC**	**CC**
	($\times10^{11}$)	(%)	(%)		($\times10^{-4}$)
Infix	0.004	58.85	49.54	0.28	1144.70
Infix Profile	1.62	76.53	41.36	1.42	15.97
Literal Profile	9.37	94.34	18.29	2.90	5.63
Complete Infix	1.69	72.64	86.60	2.14	23.21
Infix-Literal Pr.	8.79	94.48	63.75	3.07	6.38
Infix Pr.-Literal Pr.	12.00	93.57	50.03	4.29	6.52
Total Description	11.64	95.37	89.35	4.96	7.77
Total Tokenizer	15.49	96.79	60.52	4.31	5.07

Table 3: Performance of our Block Purging algorithm for all blocking schemes.

Acknowledgments. This work is partly funded by the European Commission under the FP7 EU Project GLOCAL (Contract No. 248984).

References

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

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3), 2009.

[3] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[4] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, 2003.

[5] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

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

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

[8] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, 2006.

[9] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. A framework for semantic link discovery over relational data. In *CIKM*, 2009.

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

[11] E. Ioannou, W. Nejdl, C. Niederée, and Y. Velegrakis. On-the-fly entity-aware query processing in the presence of linkage. *PVLDB*, 3(1), 2010.

[12] I. Jacobs and N. Walsh. Architecture of the world wide web, volume one. *W3C Recommendation*, December 2004.

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

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

[15] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, 2007.

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

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

[18] G. Papadakis, G. Demartini, P. Kärger, and P. Fankhauser. The missing links: Discovering hidden same-as links among a billion of triples. In *iiWAS*, 2010.

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

[20] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, pages 85–94, 2011.

[21] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. To compare or not to compare: making entity resolution more efficient. In *SWIM Workshop, co-located with SIGMOD*, 2011.

[22] V. Rastogi, N. N. Dalvi, and M. N. Garofalakis. Large-scale collective entity matching. *PVLDB*, 4(4), 2011.

[23] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *KDD*, 2002.

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

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

[26] M. Zhong, M. Liu, and Q. Chen. Modeling heterogeneous data in dataspace. In *IRI*, 2008.