Big Data Entity Resolution: From Highly to Somehow Similar Entity Descriptions in the Web

Vasilis Efthymiou[1,2], Kostas Stefanidis[2], Vassilis Christophides[1,3]
[1] University of Crete, Greece {vefthym,christop}@csd.uoc.gr
[2] *ICS-FORTH, Greece* *kstef@ics.forth.gr* [3] *INRIA Paris-Rocquencourt, France*

***Abstract*—In the Web of data, entities are described by interlinked data rather than documents on the Web. In this work, we focus on entity resolution in the Web of data, i.e., identifying descriptions that refer to the same real-world entity. To reduce the required number of pairwise comparisons, methods for entity resolution perform blocking as a pre-processing step. A blocking technique places similar entity descriptions into blocks and executes comparisons only between descriptions within the same block. We experimentally evaluate blocking techniques proposed for the Web of data and present dataset characteristics that determine the effectiveness and efficiency of such methods. Furthermore, we analyze the characteristics of the missed matching entity descriptions and examine different types of links that blocking techniques can potentially identify.**

I. Introduction

Nowadays, *knowledge bases* (KBs) offer comprehensive, machine-readable descriptions of a large variety of real-world entities (e.g., persons, places) published on the Web as *Linked Data* (LD). Although KBs (e.g., DBpedia, Freebase) may be derived from the same data source (e.g., Wikipedia), they may provide multiple descriptions of the same entities. This is mainly due to the different information extraction tools and curation policies [3] employed by KBs, resulting to complementary and sometimes conflicting descriptions. *Entity resolution* (ER) aims to identify descriptions that refer to the same entity within or across KBs [2], [4]. Compared to data warehouses, the new ER challenges stem from the *openness* of the Web of data in describing entities by an unbounded number of KBs, the *semantic and structural diversity* of the descriptions provided across domains even for the same entities, and the *autonomy* of KBs in terms of adopted processes for creating and curating descriptions.

In general, the way two descriptions can be effectively compared to efficiently decide if they refer to the same entity is challenged by the scale, diversity and graph structuring of the descriptions in the Web. This requires an understanding of the relationships among somehow similar descriptions that goes beyond duplicate detection. Also, the *huge volume* of entity collections that we need to resolve in the Web is prohibitive when examining pairwise all descriptions.

In this context of big Web data, *blocking* is typically used as a pre-processing step for ER to reduce the number of required comparisons. After blocking, each description can be compared only to others placed within the same block. The desiderata of blocking are to place (i) similar descriptions in the same block (*effectiveness*), and (ii) dissimilar descriptions in different blocks (*efficiency*). However, efficiency dictates skipping many comparisons, possibly leading to many missing matches, which in turn implies low effectiveness. Thus, the main objective of blocking is to achieve a trade-off between the number of comparisons suggested and the number of missed matches.

Most blocking algorithms (e.g., [8], [10]) assume both the availability and knowledge of a schema. Blocking approaches based on Locality-Sensitive Hashing (e.g., [9]) assume an a-priori knowledge of a minimum similarity threshold between description pairs, above which, they are considered candidate matches. Likewise, similarity-join algorithms are effective when the similarity threshold is extremely high [12]. However, in peripheral collections, matching descriptions are commonly complementary, i.e., somehow similar, and not duplicates, i.e., highly similar.

To support a Web-scale resolution of heterogeneous and loosely structured entities across domains, recent blocking algorithms (e.g., [14], [15]) disregard strong assumptions about knowledge of the schema of data and rely on a minimal number of assumptions about how entities match (e.g., when they feature a common token in their description or URI) within or across sources. However, these algorithms have not yet been experimentally evaluated with Linked Open Data (LOD) datasets exhibiting different characteristics in terms of the underlying number of entity types and size of entity descriptions (in terms of property-value pairs), as well as their structural (i.e., property vocabularies) and semantic (i.e., common property values and URLs) overlap. Existing works in ER benchmarks [7] and evaluation frameworks [11] focus on the similarity of descriptions and how these similarities affect the matching decision of ER; not on blocking, explicitly. Their data variations (focusing on highly similar descriptions) are not adequate to evaluate blocking algorithms suitable for the Web of data.

In summary, this paper makes the following contributions:

- We design a large scale evaluation on a cluster of 15 machines using real data. To capture the differences in the heterogeneity and overlap of entity descriptions, we distinguish between data originating from sources in the *center* (i.e., heavily interlinked) and the *periphery* (i.e., sparsely interlinked) of the LOD cloud.
- We empirically study the behavior of existing blocking

algorithms for datasets exhibiting different semantic and structural characteristics. We are interested in quantifying the factors that make blocking algorithms take different decisions on whether two descriptions from real LOD sources potentially match or not.
- We investigate typical cases of missed matches of existing blocking algorithms and examine alternative ways for them to be retrieved. We finally present the results of blocking, when other kinds of links, different to *owl:sameAs*, are used as a ground truth.

The rest of the paper is organized as follows. Section II overviews existing blocking algorithms, while Section III presents our implementation of those algorithms in MapReduce. Section IV describes the setup of our experiments, and Section V evaluates the blocking algorithms and analyzes the results. Finally, Section VI summarizes the paper.

II. Blocking Algorithms

We consider that an *entity description* is expressed as a set of attribute-value pairs. Then, *entity resolution* is the problem of identifying descriptions of the same entity (called *matches*). Given as input of ER the descriptions of Figure 1, $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, a possible output $P = \{\{e_1, e_4, e_6\}, \{e_2, e_5\}, \{e_3\}, \{e_7\}\}$ indicates that e_1, e_4 and e_6 refer to the same object, the Eiffel Tower, e_2 and e_5 both represent another object, the Statue of Liberty, and e_3 and e_7 represent by themselves the entities Auguste Bartholdi and Bartholdi Fountain. Such a collection is called *dirty*, since it contains duplicates, and the corresponding task is called *dirty ER*, while *clean-clean ER* is a special case of (dirty) ER [9], [15]; $\mathcal{E}$ consists of two clean, i.e., duplicate-free, but possibly overlapping collections, and ER targets at identifying their common descriptions[1]. In this work, we focus on both cases. Given $\mathcal{E}$, we define a blocking collection as a set of blocks containing the descriptions in $\mathcal{E}$.

Definition 1 (Blocking collection)**.** *Let $\mathcal{E}$ be a set of entity descriptions. A blocking collection is a set of blocks $B = \{b_1, \ldots, b_m\}$, such that, $\bigcup_{b_i \in B} b_i = \mathcal{E}$.*

To reduce the number of comparisons required by ER, *token blocking* [15] relies on the minimal assumption that matching descriptions should at least share a common token. Each distinct token t in the set of values of a description, defines a new block b_t. This essentially builds an inverted index of entity descriptions. Two descriptions are placed in the same block, if they share a token in their values.

Token blocking offers a brute-force method for comparing descriptions, even if they are considerably heterogeneous. Next, we present two extensions: attribute clustering blocking, in which candidate matches should at least share a common token for similar attributes known globally and prefix-infix(-suffix) blocking, in which candidate matches should additionally share a common URI infix.

[1] *Dirty-dirty* and *clean-dirty ER* can be seen as equivalent to *dirty ER*. Other names include *record-linkage* (for linking clean KBs) and *deduplication* (for merging duplicates in a dirty KB).

e_1={(about, Eiffel Tower), (architect, Sauvestre), (year, 1889), (located, Paris)}
e_2={(about, Statue of Liberty), (architect, Bartholdi Eiffel), (year, 1886)}
e_3={(about, Auguste Bartholdi), (born, 1834), (work, Paris)}
e_4={(about, Tour Eiffel), (built, 1887)}
e_5={(work, Lady Liberty), (artist, Bartholdi), (location, NY)}
e_6={(work, Eiffel Tower), (year-constr., 1889), (location, Paris)}
e_7={(work, Bartholdi Fountain), (year-constr., 1876), (location, Washington)}

Figure 1. A set of entity descriptions.

Attribute clustering blocking [15] exploits schematic information of the descriptions to minimize the number of false matches. To achieve this, prior to token blocking, it clusters attributes based on the similarities of their values over the entire collection of descriptions. For this step, similarity computations are exhaustively performed between all the attributes of two clean entity collections. Each attribute from one collection is connected to its most similar attribute in the other collection and connected attributes, taken by transitive closure, form non-overlapping clusters. Then, each token t in the values of an attribute, belonging to a cluster c, defines a block $b_{c.t}$. Hence, comparisons between descriptions without a common token in a similar attribute, are discarded. Like token blocking, attribute clustering generates overlapping blocks. Compared to the blocks created by token blocking, it produces a larger number of smaller blocks.

Prefix-infix(-suffix) blocking [14] exploits the prefix-infix(-suffix) pattern in the descriptions' URIs. The prefix describes the domain of the URI, the infix is a local identifier, and the optional suffix contains details about the format, or a named anchor. For example, the prefix of "http://liris.cnrs.fr/olivier.aubert/foaf.rdf#me" is "http://liris.cnrs.fr", the infix is "/olivier.aubert" and the suffix is "/foaf.rdf#me". Given a set of descriptions, this method creates one block for each token in the descriptions literal values and one block for each URI infix. It is constrained by the extent to which common naming policies are followed by the KBs. In a favourable scenario, it creates additional blocks than token blocking for the names of the descriptions, which enables to consider matching descriptions, even when they have no common tokens in their literal values.

Overall, Table I summarizes, simplified, the criteria employed by the aforementioned blocking techniques to place two descriptions into the same block.

Given this context, several interesting issues arise:

- Which blocking method performs best and for which dataset characteristcs?
- How could we evaluate a blocking method, and its ability to reduce the number of suggested comparisons versus its ability to correctly place matching descriptions in common blocks?
- Does blocking place all matches in common blocks and what are the characteristics of the missed matches?
- Could blocking be used for identifying other types of relations, e.g., geographical, between descriptions?

Table I
CRITERIA FOR PLACING DESCRIPTIONS IN THE SAME BLOCK.

Method	Criterion
Token Blocking	The descriptions have a common token in their values.
Attribute Clustering Blocking	The descriptions have a common token in the values of attributes that have similar values in overall.
Prefix-Infix(-Suffix) Blocking	The descriptions have a common token in their literal values, or a common URI infix.

In this study, we do not include recent works on ER that are orthogonal to blocking (e.g., [1], [6], [5]).

III. MAPREDUCE IMPLEMENTATION

Next, we present the MapReduce version of the evaluated techniques, designed to cope with Web data[2].

A. Token Blocking

Token blocking is essentially an inverted index of descriptions. Each token is a key in this index, associated with a list of all the descriptions containing it. In the map phase, one description of the local input split is processed at a time. For each token t in the values of a description e_i, a (t, e_i) pair is emitted by the mapper. In the reduce phase, all descriptions having a common token will be processed by the same reduce function, i.e., placed in the same block.

B. Attribute Clustering Blocking

Given two clean entity collections, our solution for attribute clustering blocking can be briefly sketched by the following steps, each representing a MapReduce job.

Attribute Creation. First, we gather the values of each attribute. In the map phase, we emit an (*attribute*, *value*) pair for each attribute-value pair in a description. We also keep the collection of this attribute in the *key*. In the reduce phase, all the values of an attribute are grouped together and their concatenation is emitted as the value of this attribute.

Attribute Similarities. In the second job, we compute the pairwise Jaccard similarities between the trigram sets of all attributes. A mapper outputs each input attribute, as many times, as the number of total mappers. Each time, a composite *key*, consisting of the current mapper id and another mapper id, will determine in which reducer the attribute will be placed, and to which other attributes it will be compared. For example, assuming 3 mappers in total, the mapper with id 2, emits for each input attribute, 3 different *keys*: 1_2, 2_2, and 2_3. The keys 1_2 and 2_3 will result in comparing the contents of mapper 2 to the contents of mappers 1 and 3, while 2_2 will result in comparing the contents of mapper 2 to each other. The *value* of each emitted pair is the input attribute with its values and the current mapper id. In the reduce phase, we compute similarities of attributes, ensuring that each comparison is performed once. For each pair of attributes, we emit a (*key*, *value*) pair, with one attribute being the *key* and the second attribute along with their similarity score being the *value*.

[2]Source code and datasets available at csd.uoc.gr/~vefthym/minoanER/

Best Match. In the third job, we use an identity mapper, which just forwards its input. A combiner keeps for each attribute of each collection, only the attribute of the other collection with the local highest similarity score. In the reduce phase, we pick for each attribute of each collection, the attribute with the maximum similarity score, in overall, from the other collection. Before this job ends, we also start the first step of clustering the most similar attributes together. To accomplish that, we emit for each best-matching attribute pair, two (attribute, clusterId) pairs, one for each attribute, with the same clusterId. Ids of clusters with common attributes are marked, in order to be merged at the next step.

Final Clustering and Blocking. In the final job, we associate each attribute with a final cluster id, according to the marks of the previous step. Then, we perform token blocking (Section III-A), with only difference that in each *key* emitted from a mapper, there is also a cluster prefix, enabling distinctions between blocks for the same token. For example, if the same token t appears in a description e_i for attributes in clusters c_j and c_k, then the mapper will emit the pairs $(c_j.t, e_i)$ and $(c_k.t, e_i)$, instead of a single (t, e_i).

C. Prefix-Infix(-Suffix)

Our MapReduce implementation of this method consists of three jobs. The first two jobs are the MapReduce adaptation of the infix extraction algorithm provided in [13]. The third job takes as input the entity descriptions, as well as the infixes produced by the second job and creates the blocks.

Prefix Removal. In the map phase, we output a (*key*, *value*) pair for each URI in a description. The *key* is the second token of the URI (after "http") and the *value* consists of the whole URI and the identifier of the entity description having this URI. This clusters the URIs according to their second token, which usually represents the domain (e.g., "dbpedia"), in the reduce phase. For each URI in a cluster, we find, among all its possible prefixes, the one with the largest set of distinct (immediately) next tokens. The part of the URI following the prefix is the *key* of each output pair, with *value* consisting of the input *key*, i.e., the second token of the URI, and the entity identifier having this URI.

Suffix Removal. We apply Prefix Removal, on the reverse string of each URI (without prefix), to remove the suffix.

Infix&Token Blocking. We create the final blocks, based on the output of Suffix Removal and the initial entity collection. We use two different mappers, operating in parallel; an identity mapper, forwarding the output of Suffix Removal and the mapper of token blocking, operating on the tokens of literal values only of the input descriptions. In the reduce phase, all the descriptions having a common token or infix in their literals or URIs will be placed in the same block.

IV. EXPERIMENTAL SETUP

In this section, we present the experimental framework we have designed for evaluating existing blocking algorithms.

We describe the datasets and the measures we employed to study the behavior of the blocking algorithms under different characteristics of entity descriptions in the LOD cloud. We have used a cluster of 15 Ubuntu 12.04.3 LTS servers (1 master, 14 slaves), each with 8 CPUs, 8GB RAM and 60GB of disk, provided by ~okeanos[3]. Each node could run simultaneously 4 map or reduce tasks, each with a heap size of 1250MB, leaving resources required for I/O and communication with the master. We used Apache Hadoop 1.2.0 and Java version 1.7.0_25 from OpenJDK.

A. Datasets

Our study relies on real data from the Billion Triples Challenge 2012 dataset[4] (BTC12), DBpedia, Kasabi[5] and the Linked Archives Hub project[6]. To capture the differences in the heterogeneity and semantic relationships of descriptions, we distinguish between data originating from sources in the *center* and the *periphery* of the LOD cloud. In general, central sources, such as DBpedia and Freebase, are derived from a common source, Wikipedia, from which they extract information regarding an entity. Such descriptions often refer to the original wiki page and feature synonym attributes whose values share a significant number of common tokens. Since they have been exhaustively studied in the literature, descriptions across central LOD sources are heavily interlinked using in their majority *owl:sameAs* links [16]. In our experiments, we used the DBpedia (*BTC12DBpedia*) and Freebase (*BTC12Freebase*) datasets from BTC12, and the raw infoboxes from DBpedia 3.5 (*Infoboxes*), i.e., two different versions of DBpedia. We also included a movies dataset[7], used in [15], extracted from DBpedia movies and *IMDB*, to validate the correctness of our algorithms.

On the other hand, data sources in the periphery of the LOD cloud are far more diverse to each other and sparsely interlinked. In our experiments, we considered the *BTC12Rest*, the *BBCmusic* and the *LOCAH* datasets. *BTC12Rest* originates from the BTC12 dataset, which consists of multiple data sources, like DBLP, geonames and drugbank. *BBCmusic* originates from Kasabi and contains descriptions regarding music bands and artists, extracted from MusicBrainz and Wikipedia. For *LOCAH*, we used the latest published version at Archives hub (March 2014). This, rather small dataset links descriptions of people, from UK archival institutions, with their descriptions in DBpedia.

Table II provides statistics about these datasets, for the number of contained triples, descriptions, attributes, and the average number of attribute-value pairs per description. We have also included the number of entity types, taken as the distinct values of the property *rdf:type*, when provided. Observe that *BTC12DBpedia* contains more types than attributes. This is due to the fact that DBpedia entities may have multiple types from taxonomic ontologies like Yago. *IMDB* is the dataset with the highest number of attribute-value pairs per description. Finally, we have included in each dataset the number of duplicate descriptions based on our ground truth, i.e., descriptions that have been reported to be equivalent (via *owl:sameAs* links) across all datasets of our testbed. Taking into account the transitivity of equality, those descriptions should be regarded as matches, too.

To investigate the ability of blocking algorithms in recognizing relatedness links beyond the *owl:sameAs* among descriptions, we considered the Kasabi *airports* and *airlines* datasets, containing data linked to DBpedia, the dataset with the highest number of references, with the *umbel:isLike* property. This property is used to associate entities that may or may not be equivalent, but are believed to be so. The *twitter* dataset contains data for the presentations of an ESWC conference. It is linked to DBpedia with the *dct:subject* property, which captures relatedness of entities to topics and it is also used in the *books* and *iati* datasets. *Books* describes books listed in the English language section of Dutch printed book auction catalogues of collections of scholars and religious ministers from the 17^{th} century. *Iati* contains data from the International Aid Transparency Initiative. *Iati* is also connected to DBpedia with the *dct:coverage* property, which associates an entity to its spatial or temporal topic, its spatial applicability, or the jurisdiction under which it is relevant. Finally, the *www2012* dataset contains data from the WWW2012 conference, linked to DBpedia with the *foaf:based_near* property, which associates an entity to an abstract notion of location. Table III details the type and the number of links of these datasets to DBpedia.

In this setting, we combine *BTC12DBpedia* with each of the datasets of Table II to produce the entity collections presented in Table IV, on which we finally ran our experiments.
- $\boldsymbol{D1}$ combines *BTC12DBpedia* with *Infoboxes*. Since it contains two versions of the same dataset, it is considered as a homogeneous collection. This is the biggest collection in terms of triples, as well as attributes.
- $\boldsymbol{D2}$ combines *BTC12DBpedia* with *BTC12Rest*. Since it is constructed by many different datasets, it is the most heterogeneous collection. Note that *BTC12Rest* has the highest number of attributes per entity type.
- $\boldsymbol{D3}$ combines *BTC12DBpedia* with *BTC12Freebase*. It is the biggest collection in terms of entity descriptions, matches, entity types and comparisons.
- $\boldsymbol{D4}$ combines *BTC12DBpedia* with *BBCmusic*. It has the lowest number of attribute-value pairs per description.
- $\boldsymbol{D5}$ combines *BTC12DBpedia* with *LOCAH*, the smallest dataset, both in terms of triples and entity descriptions.
- $\boldsymbol{D6}$ combines DBpedia movies and *IMDB*, as originally used in [15]. It is the most homogeneous collection, it only contains descriptions of movies (i.e., a single entity

[3]okeanos.grnet.gr
[4]km.aifb.kit.edu/projects/btc-2012/
[5]archive.org/details/kasabi
[6]data.archiveshub.ac.uk/
[7]l3s.de/ papadakis/erFramework.html

Table II
DATASETS CHARACTERISTICS.

	BTC12DBpedia	Infoboxes	BTC12Rest	BTC12Freebase	BBCmusic	LOCAH	DBpedia$_{mov}$	IMDB
RDF triples	102,306,242	27,011,880	849,656	25,050,970	268,759	12,932	180,680	816,012
entity descriptions	8,945,920	1,638,149	31,668	1,849,180	25,359	1,233	27,615	23,182
avg. attribute-value pairs per description	11.44	16.49	26.83	13.55	10.60	10.49	6.54	35.20
attributes	36,354	31,857	518	8,323	29	14	5	7
entity types	258,202	5,535	33	8,232	4	4	1	1
attributes/entity types	0.14	5.76	15.7	1.01	7.25	3.5	5	7
duplicates	0	0	863	12,058	372	250	0	0

Table III
CHARACTERISTICS OF DATASETS WITH DIFFERENT TYPES OF LINKS TO *BTC12DBpedia*.

	airports	airlines	twitter	books	iati		www2012
RDF triples	238,973	15,465	6,743	2,993	378,130		11,772
entity descriptions	12,294	1,141	2,932	748	31,868		1,547
avg. attribute-value pairs per description	19.44	13.55	2.30	4.00	11.87		7.61
link	*umbel:isLike*	*umbel:isLike*	*dct:subject*	*dct:subject*	*dct:subject*	*dct:coverage*	*foaf:based_near*
links	12,269	1,217	20,671	1,605	23,763	7,833	1,562

type) using the smallest number of attributes among all collections. However, the significantly greater (even by six orders of magnitude, compared to the other collections) ratio of matches to non-matches is not typical of the collections we can find in the Web of data.

Following the distinction of our datasets between central and peripheral, we also distinguish our collections between central ($D1$, $D3$ and $D6$), composed of central datasets and peripheral ($D2$, $D4$ and $D5$), part of which are peripheral datasets. For all the collections, we consider both their *clean-clean* and *dirty* versions. In practice, for our datasets, the *clean-clean* and *dirty* versions of a collection are the same; their distinction serves only as means for measuring how well a blocking technique can identify links across different datasets and within the same dataset. We finally combine *BTC12DBpedia* with each peripheral dataset of Table III to produce entity collections for studying the ability of blocking algorithms to discover different relatedness attributes.

GroundTruth. Our ground truths were built using a methodology met in the literature (e.g., [14], [15]). For $D2$-$D5$, we consider the *owl:sameAs* links to/from DBpedia 3.7 (the version used in BTC12). For $D1$, we consider the subject URIs of *Infoboxes* that also appear as subjects in *BTC12DBpedia*. The ground truth of $D6$, provided in [15], is made of DBpedia movies connected with IMDB movies through the *imdbId* property. Based on the ground truth and the generated blocks, we say that a known matching pair of descriptions is correctly resolved, i.e., a true positive (TP), if there is at least a block, to which both these descriptions belong. Pairs belonging to the same block are candidate matches. A false positive (FP) is a distinct candidate match not contained in the ground truth. In the opposite, if a known pair of matching descriptions is not a candidate match, this pair is considered a false negative (FN). All remaining pairs of descriptions are considered to be true negatives (TN).

Similarly to $D2$-$D5$, we used the available types of links of the datasets of Table III to *BTC12DBpedia*, instead of *owl:sameAs*, to produce the ground truth of the corresponding entity collections. From all datasets, except $D6$, we removed the triples present in the ground truth, since identifying those links is the goal of our tasks.

Table V
QUALITY MEASURES.

Name	Formula	Description
Recall	$\frac{TP}{TP+FN}$	Measure what fraction of the known matches are candidate matches.
Precision	$\frac{TP}{TP+FP}$	Measure what fraction of the candidate matches are known matches.
F-measure	$2\frac{Precision \cdot Recall}{Precision+Recall}$	The harmonic mean of precision and recall.
RR	$1-\frac{\text{comparisons with blocking}}{\text{comparisons without blocking}}$	Returns the ratio of reduced comparisons when blocking is applied.
$H3R$	$2\frac{RR \cdot Recall}{RR+Recall}$	The harmonic mean of recall and reduction ratio.

Our pre-processing, implemented in MapReduce, parses RDF triples in order to transform them into entity descriptions, which are the input of the methods used in our study. It simply groups the triples by subject, and outputs each group as an entity description, using the subject as the entity identifier, removing triples containing a blank node. Moreover, we kept only the entity descriptions for which we know their linked description in *BTC12DBpedia* and removed the rest. This way, we know that any suggested comparison between a pair of descriptions outside the ground-truth is false.

B. Measures

The employed quality measures along with a short description are summarized in Table V. The range of all measures is $[0, 1]$, with 1 being the ideal value. The recall of a blocking technique is the upper recall threshold of a non-iterative ER algorithm, which takes its generated blocks as input. Therefore, (1-recall) represents the *cost of blocking*. RR is the percentage of comparisons that we save if we apply the given blocking method. Consequently, it reflects the *benefit of blocking*, since the reason for using blocking in the first place, is the reduction in the required comparisons.

In general, a good blocking method should have a low impact on recall, i.e., a low cost, and a great impact on

Table IV
ENTITY COLLECTIONS CHARACTERISTICS.

	D1	D2	D3	D4	D5	D6
RDF triples	129,318,122	103,155,898	127,357,212	102,575,001	102,319,174	996,692
entity descriptions	10,584,069	8,977,588	10,795,100	8,971,279	8,947,153	50,797
avg attribute-value pairs per description	12.22	11.49	11.80	11.43	11.44	19.62
attributes	68,211	36,872	44,677	36,383	36,368	12
entity types	263,737	258,232	266,434	258,206	258,205	1
matches	1,564,311	30,864	1,688,606	23,572	1,087	22,405
matches (incl. duplicates)	1,564,311	31,727	1,700,664	23,944	1,337	22,405
matches/non-matches	$1.07 \cdot 10^{-7}$	$1.09 \cdot 10^{-7}$	$1.02 \cdot 10^{-7}$	$1.04 \cdot 10^{-7}$	$9.85 \cdot 10^{-8}$	$3.5 \cdot 10^{-5}$
matches/non-matches (dirty)	$2.79 \cdot 10^{-8}$	$7.87 \cdot 10^{-10}$	$2.92 \cdot 10^{-8}$	$5.95 \cdot 10^{-10}$	$3.34 \cdot 10^{-11}$	$1.74 \cdot 10^{-5}$
comparisons (w/o blocking)						
clean-clean	$1.47 \cdot 10^{13}$	$2.83 \cdot 10^{11}$	$1.65 \cdot 10^{13}$	$2.27 \cdot 10^{11}$	$1.1 \cdot 10^{10}$	$6.4 \cdot 10^{8}$
dirty	$5.6 \cdot 10^{13}$	$4.03 \cdot 10^{13}$	$5.83 \cdot 10^{13}$	$4.02 \cdot 10^{13}$	$4 \cdot 10^{13}$	$1.29 \cdot 10^{9}$

the number of required comparisons, i.e., a high benefit. Typically, this trade-off is captured by the F-measure, the harmonic mean of recall and precision. However, as we will see in the next section, the values of F-measure are dominated by the values of precision, which are many orders of magnitude lower than those of recall, so F-measure cannot be easily used to express this trade-off. Moreover, precision is not as important as recall is for blocking, since it can be improved in the actual matching phase that follows the blocking phase. Thus, we introduce $H3R$ as the harmonic mean of recall and reduction ratio. Similar to the F-measure, $H3R$ gives high values only when both recall and reduction ratio have high values. Unlike F-measure, $H3R$ manages to capture the trade-off between effectiveness and efficiency in a more balanced way. Note that $H3R$ does not estimate the performance of a blocking approach (as, for example, [14] does), but evaluates it based on the actual results.

V. EXPERIMENTAL EVALUATION

In this section, we analyze the quality and performance of the evaluated blocking techniques, taking into consideration the specific features of each dataset. Then, we present the results of blocking, when different kinds of links, other than *owl:sameAs*, are used as ground truth and conclude with a discussion of the lessons learned from this analysis. In our evaluation, we use the adaptation of token blocking ToB, attribute clustering AtC and prefix-infix(-suffix) blocking PIS in Map/Reduce. Both ToB and PIS can be used with either *clean-clean* Cl or *dirty* Di entity collections, while AtC is suitable for Cl collections. Moreover, the process of AtC requires a similarity function; we use Jaccard similarity over the set of trigrams from the values (similar to [15]).

A. Quality Results

1) Identified Matches (TPs): **Token blocking:** The premise of this algorithm is that matching descriptions should at least share a common token, disregarding the comparisons between descriptions that do not share common tokens. Therefore, the higher the number of common tokens, i.e., tokens shared by the datasets composing an entity collection, a description has, the higher the chances it will be placed in a block with a matching description, increasing recall. Figure 2 (left) presents the distributions of common tokens per description, showing that descriptions in central collections feature many more common tokens than those in peripheral ones[8]. For example, 41.43% and 44% of descriptions in $D1$ and $D3$ have 2-4 common tokens, while for $D2$, $D4$ and $D5$ the corresponding values are 33.26%, 26.03% and 12.97%. We observe a big difference in the distribution of $D6$, which contains many more common tokens per description, to those of the other collections, due to the ratio of matches to non-matches that is much higher than in the other collections (Table IV). Only 23.75% of the descriptions in this collection have 0 - 10 common tokens. This figure also shows that a big number of descriptions in peripheral collections, do not share any common tokens. Those are hints that the recall of token blocking in central collections is higher than in peripheral collections.

Indeed, $D6$ is the dataset with the highest recall (99.92%) and the highest number of common tokens per entity (19), while $D5$ is the dataset with the lowest recall (72.13%) and number of common tokens per entity (0). There is a big difference in the number of common tokens in $D6$, compared to $D1$ and $D3$, which is not reflected by their small difference in recall. Due to the high ratio of matches to non-matches in $D6$ (Table IV), descriptions in this collection have many common tokens and this leads to high recall.

Attribute clustering blocking: The goal of attribute clustering is to improve the precision of token blocking, while retaining its recall as much as possible (it cannot have higher recall). To do this, it restricts the number of attributes on which descriptions, featuring a common token, should be compared. Comparisons between descriptions that do not share a common token in a common attribute cluster, are discarded. Hence, descriptions with many common tokens in common clusters are more likely to be matched. Figure 2 (right) presents the distributions of the number of common tokens in common attribute clusters per entity. It shows a clearer distinction between central and peripheral collections than Figure 2 (left); the descriptions in central collections have many more common tokens in common

[8]We take the median values and not the averages, as the latter are highly influenced by extreme values and our distributions are skewed.

Table VI
STATISTICS AND EVALUATION OF BLOCKING METHODS.

	D1	D2	D3	D4	D5	D6
Token blocking statistics:						
blocks	1,639,962	122,340	1,019,501	57,085	2,109	40,304
comparisons (clean-clean)	$1.68 \cdot 10^{12}$	$3.74 \cdot 10^{10}$	$6.56 \cdot 10^{11}$	$2.39 \cdot 10^{10}$	$8.72 \cdot 10^{8}$	$2.91 \cdot 10^{8}$
RR (clean)	88.51%	86.81%	96.03%	89.48%	92.09%	54.50%
comparisons (dirty)	$5.56 \cdot 10^{12}$	$3.68 \cdot 10^{12}$	$4.27 \cdot 10^{12}$	$4.02 \cdot 10^{12}$	$1.01 \cdot 10^{12}$	$2.05 \cdot 10^{9}$
RR (dirty)	90.08%	90.87%	92.67%	90.01%	97.48%	−58.85%
common tokens per entity (median)	4	3	4	2	0	19
Attribute clustering blocking statistics:						
blocks	5,602,644	150,293	1,673,855	39,587	3,724	43,716
comparisons	$3.22 \cdot 10^{11}$	$4.20 \cdot 10^{9}$	$1.84 \cdot 10^{11}$	$1.43 \cdot 10^{9}$	$7.13 \cdot 10^{8}$	$2.13 \cdot 10^{8}$
RR	97.80%	98.52%	98.89%	99.37%	93.54%	66.80%
common tokens in common att. clusters per entity (median)	4	0	4	2	0	19
attribute clusters	16,886	124	2,106	6	8	4
attributes per attribute cluster (median)	2	142	9	4,261	3,946	3
Prefix-Infix(-Suffix) blocking statistics:						
blocks	3,266,798	141,517	789,723	45,403	2,098	N/A
comparisons (clean-clean)	$1.10 \cdot 10^{12}$	$1.78 \cdot 10^{10}$	$2.75 \cdot 10^{11}$	$2.30 \cdot 10^{9}$	$4.08 \cdot 10^{8}$	N/A
RR (clean)	92.48%	93.72%	98.34%	98.99%	96.30%	N/A
comparisons (dirty)	$4.39 \cdot 10^{12}$	$3.45 \cdot 10^{12}$	$5.34 \cdot 10^{12}$	$3.32 \cdot 10^{12}$	$1.76 \cdot 10^{12}$	N/A
RR (dirty)	92.16%	91.44%	90.84%	91.76%	95.59%	N/A
Recall:						
Token blocking (clean-clean)	98.38%	92.46%	95.52%	87.76%	72.13%	99.92%
Token blocking (dirty)	98.38%	89.99%	94.85%	87.95%	77.34%	99.92%
Attribute clustering blocking	97.31%	68.42%	92.10%	76.84%	71.11%	99.55%
Prefix-Infix(-Suffix) blocking (clean-clean)	100%	91.71%	87.68%	95.44%	68.17%	N/A
Prefix-Infix(-Suffix) blocking (dirty)	100%	89.25%	87.06%	95.50%	74.12%	N/A
Precision:						
Token blocking (clean-clean)	$1.56 \cdot 10^{-6}$	$1.00 \cdot 10^{-6}$	$2.49 \cdot 10^{-6}$	$1.30 \cdot 10^{-6}$	$1.13 \cdot 10^{-6}$	$1.21 \cdot 10^{-4}$
Token blocking (dirty)	$3.64 \cdot 10^{-7}$	$5.14 \cdot 10^{-9}$	$3.78 \cdot 10^{-7}$	$1.05 \cdot 10^{-8}$	$1.29 \cdot 10^{-9}$	$7.51 \cdot 10^{-5}$
Attribute clustering blocking	$8.51 \cdot 10^{-6}$	$5.76 \cdot 10^{-6}$	$1.01 \cdot 10^{-5}$	$1.41 \cdot 10^{-5}$	$1.35 \cdot 10^{-6}$	$1.52 \cdot 10^{-4}$
Prefix-Infix(-Suffix) blocking (clean-clean)	$1.87 \cdot 10^{-6}$	$2.19 \cdot 10^{-6}$	$5.72 \cdot 10^{-6}$	$1.01 \cdot 10^{-5}$	$2.05 \cdot 10^{-6}$	N/A
Prefix-Infix(-Suffix) blocking (dirty)	$6.04 \cdot 10^{-7}$	$8.21 \cdot 10^{-9}$	$2.77 \cdot 10^{-7}$	$1.23 \cdot 10^{-8}$	$6.99 \cdot 10^{-10}$	N/A
F-measure:						
Token blocking (clean-clean)	$3.13 \cdot 10^{-6}$	$2.00 \cdot 10^{-6}$	$9.72 \cdot 10^{-7}$	$2.06 \cdot 10^{-8}$	$1.94 \cdot 10^{-9}$	$2.42 \cdot 10^{-4}$
Token blocking (dirty)	$7.28 \cdot 10^{-7}$	$1.03 \cdot 10^{-8}$	$7.55 \cdot 10^{-7}$	$2.10 \cdot 10^{-8}$	$2.59 \cdot 10^{-9}$	$1.50 \cdot 10^{-4}$
Attribute clustering blocking	$1.70 \cdot 10^{-5}$	$1.15 \cdot 10^{-5}$	$2.02 \cdot 10^{-5}$	$2.82 \cdot 10^{-5}$	$2.69 \cdot 10^{-6}$	$3.04 \cdot 10^{-4}$
Prefix-Infix(-Suffix) blocking (clean-clean)	$3.75 \cdot 10^{-6}$	$4.38 \cdot 10^{-6}$	$9.98 \cdot 10^{-7}$	$2.02 \cdot 10^{-5}$	$4.11 \cdot 10^{-6}$	N/A
Prefix-Infix(-Suffix) blocking (dirty)	$1.21 \cdot 10^{-6}$	$1.64 \cdot 10^{-8}$	$5.55 \cdot 10^{-7}$	$2.46 \cdot 10^{-8}$	$1.40 \cdot 10^{-9}$	N/A
$H3R$:						
Token blocking (clean-clean)	93.18%	89.55%	95.77%	88.61%	80.90%	70.53%
Token blocking (dirty)	94.05%	90.43%	93.75%	88.97%	86.25%	N/A ($RR < 0$)
Attribute clustering blocking	97.55%	80.76%	95.37%	86.66%	80.80%	79.95%
Prefix-Infix(-Suffix) blocking (clean-clean)	96.09%	92.70%	92.70%	97.18%	79.83%	N/A
Prefix-Infix(-Suffix) blocking (dirty)	95.92%	90.33%	88.91%	93.59%	83.50%	N/A

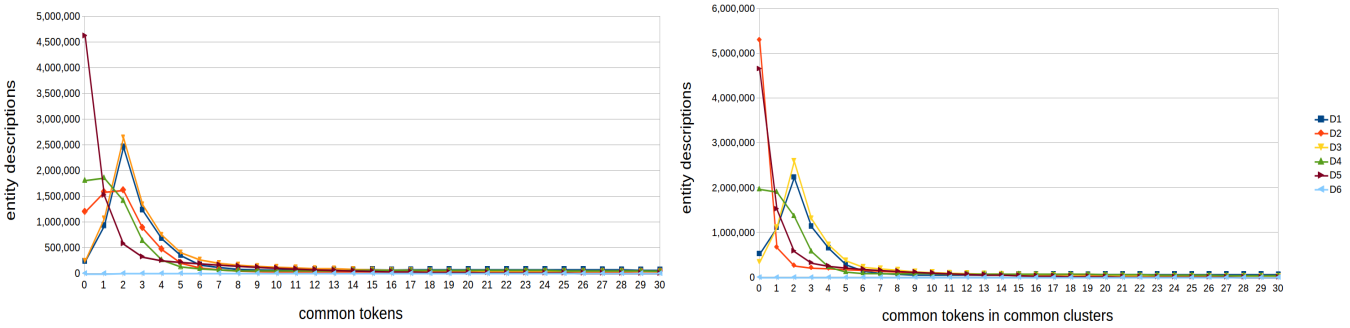

Figure 2. Common tokens (left) and common tokens in common clusters (right) per entity description distributions for $D1$-$D6$.

clusters, while many descriptions in peripheral collections do not have any common token in a common cluster. This occurs, because values in the descriptions of peripheral collections are much less similar than those of central collections, leading to a bad clustering of the attributes and, thus, to lower recall. In fact, $D6$ is the dataset with the highest recall (99.55%) and the highest number of common tokens in common attribute clusters per entity (19). On the other hand, $D2$ and $D5$, which have the lowest recall values (68.42% and 71.11%) also have the lowest number of common token in common attribute clusters per entity (0).

In central collections ($D1$, $D3$, $D6$), many, small clusters

of similar attributes are formed, as the values of the descriptions are similar. This leads to a minor decrease in recall, compared to token blocking, while it significantly improves its precision (even by an order of magnitude in $D3$). $D1$ forms many (16,886), small attribute clusters (of 2 attributes in the median case), since in most cases there is an 1-1 mapping between the attributes of the datasets that compose it. These clusters contain the same attribute used by the two versions of DBpedia that compose this collection.

However, this approach has a substantial impact on recall in peripheral collections ($D2$, $D4$, $D5$), even if it still improves precision in all collections (even by an order of magnitude for $D4$). The descriptions in those collections have few common tokens, in the first place, which leads to a bad clustering of attributes; few clusters of many attributes, not similar to each other, are formed. Hence, if we make the blocking criterion of token blocking stricter, by also considering attributes, then the more distinct attributes used per entity type, the more difficult it is for an entity description, to be placed in a common block with a matching description. For *BTC12Rest* (part of $D2$), the ratio between attributes and entity types (last row of Table II) is the highest (15.7), leading to a great impact on recall (-24.04%). This dataset has the biggest number of data sources that compose it and many different attribute names can be used for the same purpose; hence, big attribute clusters are formed. *LOCAH* (part of $D5$) only has 3.5 attributes per entity type. Thus, the recall of attribute clustering blocking is insignificantly reduced (-1.02%), compared to that of token blocking.

Prefix-Infix(-Suffix) blocking: Prefix-Infix(-Suffix) blocking is built on the premise that many URIs contain useful information. Its goal is to extend token blocking and improve both its recall, by also considering the subject URIs of the descriptions, and its precision, by disregarding some unneeded tokens in the URI values (either in the prefix or suffix). It achieves good recall values in datasets with similar naming policies in the URIs, as in $D4$, part of which is *BBCmusic*, which also has Wikipedia as a source. However, it misses many matching pairs of descriptions, when the names of the URIs do not contain useful information, as in $D3$ that uses random strings as ids, or have different policies, as in $D5$, which uses concatenations of tokens, without delimiters, as URIs. The recall of $D1$ is 100%, because the collection is constructed this way; it consists of two versions of the same dataset, DBpedia, and the URIs appearing as subjects in *Infoboxes* are only those URIs that also appear as subjects in *BTC12DBpedia*. PIS is not applicable (marked as N/A) to $D6$, since its URIs have been replaced with numerical identifiers in the provided datasets.

2) Missed Matches (FNs): A non-negligible number of matching pairs of descriptions do not share any common tokens at all. Such descriptions, constituting the false negatives of token blocking, should not be assumed faulty, or noisy. We distinguish two different sources of information that can be exploited for successfully placing descriptions of missed matches in common blocks:

i. The matches of their neighbors: Given that a description can have, as one of its values, another description, neighborhoods of related descriptions are formed, spinning the Web of data. The knowledge of the matches of the neighbors of a description is valuable for correctly matching this description. For example, if the description e_{10} is related to e_1, e_{20} is related to e_2 and we know that e_{10} and e_{20} match, then we can use this knowledge as a hint that e_1 and e_2 could possibly match, too.
ii. A third, matching description: In dirty collections (typically peripheral), which are composed of datasets that potentially contain duplicate descriptions, a description e_1 could have more than one matching description, e.g., both e_2 and e_3. Identifying one of these matches, e.g., (e_1, e_3), knowing that (e_2, e_3) is a match, leads to also identify the missing match (e_1, e_2).

Table VII provides details about the number and the characteristics of false negative pairs of descriptions, and the set of individual descriptions that constitute these pairs[9].

We focus first on the neighbors of these descriptions, namely descriptions that appear in their values. We found that almost all the descriptions in the false negatives have at least one neighbor (second row of Table VII). Looking more thoroughly, we counted the percentage of descriptions in false negatives that have at least one neighbor belonging to the ground truth (third row of Table VII). In all cases, this percentage is more than 10% and goes up to 58% for $D4$. This means that, not only do these descriptions have neighbors, but many of these neighbors can be matched to other descriptions in the same collection as well. Then, we counted the percentage of descriptions in false negatives that have neighbors, which have already been matched to another description (fourth row of Table VII). This percentage is over 20% in most collections, while it reaches up to 51.84% for $D4$. Finally, we counted the percentage of false negative pairs, whose descriptions have neighbors, which match to each other (fifth row of Table VII). This percentage is 0 for $D1$, as matches in this collection are defined as descriptions that have the same subject URI. However, in some peripheral collections ($D2$, $D4$), examining the matches of the neighbors of the descriptions is meaningful.

Another useful piece of information for the missed matches of dirty collections is whether their descriptions have been correctly matched to a third description. The last row of Table VII quantifies this statistic, showing that there are collections, both peripheral ($D2$, $D5$) and central ($D3$), for which this kind of information could, indeed, be useful.

3) Non-matches (FPs and TNs): Next, we examine the ability of blocking methods to identify non-matches, namely their ability to avoid placing non-matching descriptions in

[9] $D6$ is excluded, as it does not contain any descriptions with neighbors.

Table VII
CHARACTERISTICS OF THE MISSED MATCHES OF TOKEN BLOCKING.

	D1	D2	D3	D4	D5
FNs	25,419	3,176	87,672	2,886	303
descriptions in FNs, with neighbor(s)	99.64%	100%	99.99%	100%	100%
descriptions in FNs, with neighbor(s) in ground truth	22.60%	53.94%	36.43%	58.36%	11.57%
descriptions in FNs, with neighbor(s) with an identified match	20.94%	48.54%	34.05%	51.84%	7.59%
FNs with matching neighbors	0%	24.81%	0.38%	37.63%	0%
FNs with common, identified matches	0%	25.35%	10.54%	0.14%	8.58%

the same block. A key statistic for this, regarding the datasets, is the ratio of matches to non-matches (Table IV). The higher the ratio, the easier it is for a blocking method to have better precision, as it statistically has better chances of suggesting a correct comparison. $D6$ is the collection with the highest such ratio and precision, while $D5$ has the lowest ratio and, in most blocking methods, the lowest precision, too. It is clear from Table VI that attribute clustering is the most precise method, since, in almost every case, it results in the fewest wrong suggestions. On the contrary, the least precise method is token blocking, in all cases. The differences in precision, in some cases even by an order of magnitude, also determine F-measure, since the differences in recall are not that big. Note that all the evaluated methods have very low precision, meaning that the vast majority of suggested comparisons correspond to non-matches.

B. Performance Results

Table VI shows that all the evaluated methods manage to greatly reduce the number of comparisons that would be required if blocking was not employed, by one ($D1$-$D4$) or two ($D5$) orders of magnitude. This is reflected by high RR scores in all cases. An exception is $D6$, which is much smaller in terms of descriptions and, consequently, comparisons without blocking. Moreover, its descriptions contain many more common tokens than the other collections, leading to more comparisons per entity. Therefore, token blocking does not save many of the comparisons that would be required without blocking and, in the case of $D6$ dirty, it even produces twice as many comparisons.

With respect to $H3R$, we notice that, in general, central collections have higher scores, i.e., they present a better balance between recall and reduction ratio. This means that in these collections, comparisons that are discarded by blocking mostly correspond to non-matches, while many of the comparisons discarded by blocking in peripheral collections correspond to matches. Again, $D6$ has a different behaviour, since it initially contains a much smaller number of comparisons and a high ratio of matches to non-matches, so the reduction ratio for this collection is limited. These measures are not applicable to token blocking, when applied to $D6$ dirty, since in that case the reduction ratio is negative.

C. Different Types of Links

In order to evaluate the ability of blocking methods to identify more types of links, semantically close or even not that close to equivalence links, we have run a set of experiments with the peripheral collections consisting of each of the datasets of Table III and *BTC12DBpedia*. Table VIII provides the recall of token blocking, when applied to each of those collections. Similarly to the *owl:sameAs* links, token blocking performs well for links with the semantics of equivalence, as in the *airports* and *airlines* datasets with recall values close to 100%. It also manages to identify many subject associations, as in the cases of *books* and *iati* datasets. It performs poorly in identifying this kind of association, however, in the *twitter* dataset, where its recall values fall to below 10%. This could be justified by the nature of this dataset, which, in most cases, simply states who is the maker of some slides. Regarding spatial associations, token blocking manages to identify a mere 39% of the coverage associations of the *iati* dataset, but it performs much better in identifying the *based_near* associations of *www2012*, with a recall of 63%. The spatial relationships of *coverage* are looser than those of *based_near*, hence the related entities are not so strongly related in the former type of links. For example, in *iati*, the description of a project regarding the evaluation of cereal crop residues is linked to the DBpedia resource describing Latin America and the Caribbean, through the *coverage* relation, while, in *www2012*, a Greek professor is linked to the DBpedia resource describing Greece, through the *based_near* relation.

D. Lessons Learned

We now present the key points of our evaluation. *Central* collections are mostly derived from Wikipedia, from which they extract information regarding an entity. This way, descriptions in such collections follow similar naming policies and feature many common tokens (Figure 2) in the values of semantically similar, or equivalent attributes (see the small size of clusters in Table VI). Those are exactly the premises on which the evaluated blocking methods are built.

For these reasons, the recall achieved by token blocking in central entity collections is very high (ranges from 99.92% to 94.85%). With the exception of $D6$ (featuring a higher ratio of matching to non-matching descriptions), the precision achieved by token blocking in these collections ranges from $2.49 \cdot 10^{-6}$ to $3.64 \cdot 10^{-7}$. The gains in precision brought by attribute clustering blocking in central entity collections are up to one order of magnitude (for $D3$), with a minor cost on recall (from 0.37% to 3.42%). Prefix-infix(-suffix) blocking can improve both recall and precision of token blocking for central collections, as in $D1$, but, it can also deteriorate these values, as in the dirty case of $D3$, which uses random identifiers as URIs, in which recall drops by 7.79% and precision by 26.72%. In a nutshell, many redundant comparisons are suggested by blocking algorithms

Table VIII
RECALL OF THE COLLECTIONS COMPOSED OF DATASETS OF TABLE III AND *BTC12DBpedia*.

	airports	**airlines**	**twitter**	**books**	**iati**		**www2012**
link	*umbel:isLike*	*umbel:isLike*	*dct:subject*	*dct:subject*	*dct:subject*	*dct:coverage*	*foaf:based_near*
Recall of token blocking	97.47%	99.75%	9.52%	63.55%	49.13%	39.46%	62.61%

in all entity collections (see precision and F-measure in Table VI), due to the small ratio of matches to non-matches in the collections (Table IV). However, as the $H3R$ reveals, the comparisons that are discarded by blocking in central collections mostly correspond to non-matches.

On the contrary, descriptions in *peripheral* KBs are more diverse, following different naming policies and sharing few common tokens (Figure 2), since they stem from various sources. The lack of similar values in those descriptions leads to a bad clustering of attributes; big clusters of attributes not similar to each other are formed (Table VI).

For these reasons, the recall of token blocking for peripheral collections drops even to 72.13%, while precision ranges from $1.3 \cdot 10^{-6}$ to $1.29 \cdot 10^{-9}$. The gains in precision brought by attribute clustering blocking (up to one order of magnitude) in peripheral collections, come at the cost of a drop in recall up to 24.04% (corresponding to 7,421 more missed matches). Prefix-infix(-suffix) blocking can improve the precision of token blocking in peripheral collections, even by an order of magnitude (for $D4$), or decrease it by an order of magnitude (for $D5$), while it decreases recall from 0.74% to 3.96%, i.e., more matches are missed. In the case of $D4$, in which both datasets use Wikipedia as a source, recall is improved by up to 7.68%. In overall, however, $H3R$ reveals that many of the comparisons that are discarded by blocking in peripheral collections correspond to matches.

Nevertheless, information for the missed matches, e.g., from the neighborhoods of their descriptions (Table VII), sets the ground for a new generation of ER algorithms, which will exploit this information to identify more matches, in an iterative fashion. In preliminary experiments, we found that even a single match in the neighborhood of a candidate pair is a good match-indication for that that pair, too.

Finally, in peripheral collections, there are several types of relations, other than equivalence, between descriptions. Token blocking identifies some of them, depending on the dataset, the specific type of such links, and the immediacy of those relations (Table VIII). It does not perform well when the data do not contain much information (e.g., see the characteristics of *twitter* in Table III), or when the relationship of the entities is loose (e.g., see the recall of *iati* for *dct:coverage* in Table VIII). Thus, for a quantitative evaluation of blocking methods ground truth should not be restricted only to *owl:sameAs* links. We could potentially take other relations into account, to identify more such links, or more *owl:sameAs* links, using iterative algorithms.

VI. SUMMARY

In this work, we evaluated, for the first time, ER blocking algorithms for somehow (not only highly) similar descriptions in the Web of data. We have investigated the data characteristics of such descriptions that impact blocking algorithms' effectiveness and efficiency. Highly similar descriptions, met in central LOD collections, feature many common tokens in the values of common attributes, while somehow similar descriptions, met in peripheral collections, have significantly fewer common tokens in attributes that are not necessarily semantically related. Hence, the former can be compared only on their content (i.e., values), while the latter require contextual information, e.g., the similarity of neighborhood descriptions, linked with different types of relationships. Since a single similarity function cannot identify such matches in a single pass, multiple iterations of matching (focusing on context) and/or blocking (focusing on content) are needed. Towards this end, we are interested in progressive ER algorithms that try to maximize the benefit (e.g., number of resolved entities, number of links between resolved entities) of each iteration, by dynamically adapting their execution plan, based on results of previous iterations.

Acknowledgements. This work was partially supported by the EU FP7 DIACHRON (#601043) and H2020 PARTHENOS (#654119) projects.

REFERENCES

[1] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. LINDA: distributed web-of-data-scale entity matching. In *CIKM*, 2012.
[2] V. Christophides, V. Efthymiou, and K. Stefanidis. *Entity Resolution in the Web of Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2015.
[3] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *SIGMOD*, 2013.
[4] X. L. Dong and D. Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015.
[5] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, and T. Palpanas. Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data. In *IEEE Big Data*, 2015.
[6] O. Hassanzadeh, K. Q. Pu, S. H. Yeganeh, R. J. Miller, L. Popa, M. A. Hernández, and H. Ho. Discovering linkage points over web data. *PVLDB*, 6(6):444–456, 2013.
[7] E. Ioannou, N. Rassadko, and Y. Velegrakis. On generating benchmark data for entity matching. *J. Data Semantics*, 2(1):37–56, 2013.
[8] B. Kenig and A. Gal. MFIBlocks: an effective blocking algorithm for entity resolution. *Inf. Syst.*, 38(6):908–926, 2013.
[9] H. Kim and D. Lee. HARRA: fast iterative hashed record linkage for large-scale data collections. In *EDBT*, 2010.
[10] L. Kolb, A. Thor, and E. Rahm. Dedoop: Efficient deduplication with hadoop. *PVLDB*, 5(12):1878–1881, 2012.
[11] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.
[12] A. Metwally and C. Faloutsos. V-smart-join: A scalable mapreduce framework for all-pair similarity joins of multisets and vectors. *PVLDB*, 5(8):704–715, 2012.
[13] G. Papadakis, G. Demartini, P. Fankhauser, and P. Kärger. The missing links: discovering hidden same-as links among a billion of triples. In *iiWAS*, 2010.
[14] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl. Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data. In *WSDM*, 2012.
[15] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederée, and W. Nejdl. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Trans. Knowl. Data Eng.*, 25(12):2665–2682, 2013.
[16] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *ISWC*, 2014.