Web-scale Blocking, Iterative and Progressive Entity Resolution

Kostas Stefanidis
University of Tampere
Finland
kostas.stefanidis@uta.fi

Vassilis Christophides
INRIA-Paris & Univ. of Crete
France & Greece
vassilis.christophides@inria.fr

Vasilis Efthymiou
ICS-FORTH & Univ. of Crete
Greece
vefthym@ics.forth.gr

***Abstract*—Entity resolution aims to identify descriptions of the same entity within or across knowledge bases. In this work, we provide a comprehensive and cohesive overview of the key research results in the area of entity resolution. We are interested in frameworks addressing the new challenges in entity resolution posed by the Web of data in which real world entities are described by interlinked data rather than documents. Since such descriptions are usually partial, overlapping and sometimes evolving, entity resolution emerges as a central problem both to increase dataset linking, but also to search the Web of data for entities and their relations. We focus on Web-scale blocking, iterative and progressive solutions for entity resolution. Specifically, to reduce the required number of comparisons, blocking is performed to place similar descriptions into blocks and executes comparisons to identify matches only between descriptions within the same block. To minimize the number of missed matches, an iterative entity resolution process can exploit any intermediate results of blocking and matching, discovering new candidate description pairs for resolution. Finally, we overview works on progressive entity resolution, which attempt to discover as many matches as possible given limited computing budget, by estimating the matching likelihood of yet unresolved descriptions, based on the matches found so far.**

I. Introduction

Over the past decade, numerous knowledge bases (KBs) have been built to power large-scale knowledge sharing, but also an entity-centric Web search, mixing both structured data and text querying. These KBs offer comprehensive, machine-readable descriptions of a large variety of real-world entities (e.g., persons, places, products, events) published on the Web as Linked Data (LD). Traditionally, KBs are manually crafted by a dedicated team of knowledge engineers, such as the pioneering projects Wordnet and Cyc. Today, more and more KBs are built from existing Web content using information extraction tools. Such an automated approach offers an unprecedented opportunity to scale-up KBs construction and leverage existing knowledge published in HTML documents. Although KBs (e.g., DBpedia, Freebase) may be derived from the same data source (e.g., a Wikipedia entry), they may provide multiple, non-identical descriptions of the same real-world entities. This is mainly due to the different information extraction tools and curation policies employed by KBs, resulting to complementary and sometimes conflicting entity descriptions.

Entity resolution (ER) aims to identify descriptions that refer to the same real-world entity appearing either within or across KBs [8], [9]. 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 real-world entities, as well as the autonomy of KBs in terms of adopted processes for creating and curating entity descriptions. In particular:

- The number of KBs (aka RDF datasets) in the Linking Open Data (LOD) cloud[1] has roughly tripled between 2011 and 2014 (from 295 to 1014), while KBs interlinking dropped by 30%. The main reason is that with more KBs available, it becomes more difficult for data publishers to identify relations between the data they publish and the data already published. Thus, the majority of KBs are sparsely linked, while their popularity in links is heavily skewed. Sparsely interlinked KBs appear in the periphery of the LOD cloud (e.g., Open Food Facts, Bio2RDF), while heavily interlinked ones lie at the center (e.g., DBpedia, GeoNames). Encyclopaedic KBs, such as DBpedia, or widely used georeferencing KBs, such as GeoNames, are interlinked with the largest number of KBs [25].
- The descriptions contained in these KBs present a high degree of semantic and structural diversity, even for the same entity types. Despite the Linked Data principles, multiple names (e.g., URIs) can be used to refer to the same real-world entity. The majority (58.24%) of the 649 vocabularies currently used by KBs are proprietary, i.e., they are used by only one KB, while diverse sets of properties are commonly used to describe the entities both in terms of types and number of occurrences even in the same KB. Only YAGO contains 350K different types of entities, while Google's Knowledge Graph contains 35K properties, used to describe 600M entities.

The two core ER problems, i.e, how can we (a) effectively compute similarity of entity descriptions and (b) efficiently resolve sets of entities within or across sources, are challenged by the large scale (both in terms of the number of sources and entity descriptions), the high diversity (both in terms of number of entity types and properties) and the importance of relationships among entity descriptions (not committing to a particular schema defined in advance). In particular, in addition to *highly similar* descriptions that feature many common tokens in values of semantically related attributes, typically met in the center of the LOD cloud and heavily interlinked mostly

[1] http://lod-cloud.net

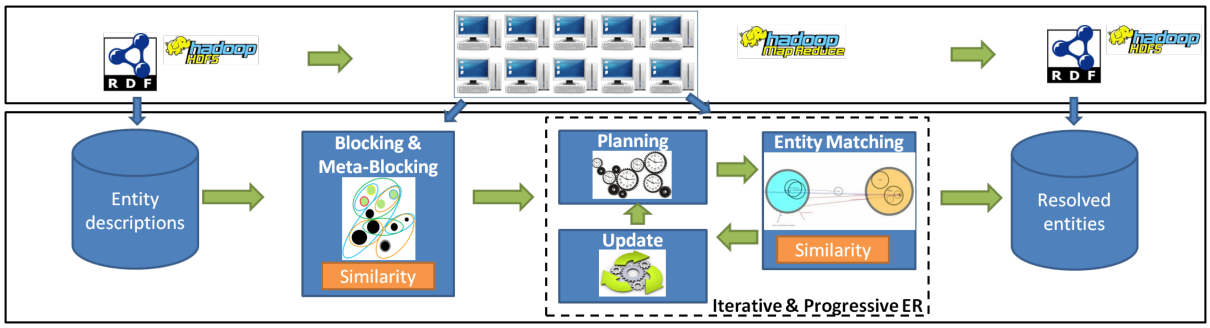

Fig. 1. The ER Framework.

using owl:sameAs predicates, we are encountering *somehow similar* descriptions with significantly fewer common tokens in attributes not always semantically related, that appear usually in the periphery of the LOD cloud and are sparsely interlinked with various kinds of predicates. Plainly, the coming up of highly and somehow similar semi-structured entity descriptions requires solutions that go beyond those applicable to duplicate detection. More precisely, deduplication techniques [6] are essentially ER techniques for highly (low) similar in structure (content) descriptions from one relation, record linkage for structured [6] or semi-structured Web data [15] targets highly (low) similar in content (structure) descriptions from two relations, while in the Web of data, descriptions hosted in a network of KBs exhibit low similarity both in content and structure (i.e., are somehow similar).

A promising area of research in this respect is cross-domain similarity search and mining (e.g., [29]), aiming to exploit similarity of objects described by different modalities (i.e., text, image) and contexts (i.e., facets) and support research by analogy. Such techniques could be also beneficial for matching highly heterogeneous entity descriptions and thus support ER at the Web scale.

In this tutorial, we present how to resolve entities described by linked data in the Web (e.g., in RDF). Figure 1 illustrates the general steps involved in our process.

II. Blocking and Meta-blocking for Entity Resolution

Grouping entity descriptions in blocks before comparing them for matching is an important pre-processing step for pruning the quadratic number of comparisons required to resolve a collection of entity descriptions [7]. The main objective of algorithms for entity blocking is to achieve a reasonable compromise between the number of comparisons suggested and the number of missed entity matches. In this tutorial, we briefly present traditional blocking algorithms proposed for relational records and explain why they cannot be used in the Web of data. Then, we detail a family of algorithms that relies on a simple inverted index of entity descriptions extracted from the tokens of their attribute values. Hence, two descriptions are placed into the same block if they share at least a common token (e.g., [21], [20]).

An alternative approach for blocking is to consider string-similarity join algorithms. In a nutshell, such algorithms construct blocks by identifying all pairs of descriptions whose string values similarities are above a certain threshold and potentially some pairs whose string values similarities are below that threshold. To achieve that, without computing the similarity of all pairs of descriptions, string-similarity join algorithms (e.g., [5], [28]) build an inverted index from the tokens of the descriptions values. A method to reduce the number of compared descriptions consists of building blocks for sets of tokens that appear together in many entity descriptions. Several variations of this problem have been proposed. For example, [19] studies the problem of scalable finding frequent sets of tokens that appear in specific sequences. Finally, [17] proposed the concept of multidimensional overlapping blocks. Specifically, this method firstly constructs a collection of blocks for each similarity function, and then, all blocking collections are aggregated into a single multidimensional collection, taking into account the similarities of descriptions that share blocks.

Blocking, even as a pre-processing step for entity resolution, is a heavy computational task. This way, several approaches exploit the MapReduce programming model for parallelizing blocking algorithms (e.g., [18], [12], [13]). Abstractly, in all cases, a collection of entity descriptions, given as input to a MapReduce job, is split into smaller chunks, which are then processed in parallel. A map function, emitting intermediate (key, value) pairs for each input split, and a reduce function that processes the list of values of an intermediate key, coming from all the mappers, should be defined.

To further improve the efficiency of blocking, several approaches (e.g., [26], [21], [20]) focus on different ways for discarding comparisons that do not lead to matches. More recently, [22], [10], [11] propose to reconstruct the blocks of a given blocking collection in order to more drastically discard redundant comparisons, as well as comparisons between descriptions that are unlikely to match. Meta-blocking essentially transforms a given blocking collection B into a blocking graph. Its nodes correspond to the descriptions in B, while its undirected edges connect the co-occurring descriptions. No parallel edges are allowed, thus eliminating all redundant comparisons. Every edge is associated with a weight, representing the likelihood that the adjacent entities are matching candidates. The low-weighted edges are pruned, so as to discard comparisons between unlikely-to-match descriptions.

III. Iterative Entity Resolution

The inherent distribution of entity descriptions in different knowledge bases, along with their significant semantic and structural diversity, yield incomplete evidence regarding the

similarity of descriptions published in the Web of data. Iterative ER approaches aim to tackle this problem by exploiting any partial result of the ER process in order to generate new candidate pairs of descriptions not considered in a previous step or even to revise a previous matching decision.

Abstractly, new matches can be found by exploiting merged descriptions of previously identified matches or relationships between descriptions. We call the iterative ER approaches that build their iterations on the merging of descriptions *merging-based*, and those that use entity relationships for their iteration step *relationship-based*. Intuitively, the merging-based approaches deal with descriptions of the same type, e.g., descriptions for persons, while relationship-based approaches presume upon the relationships between different types of entities. Interestingly, there is an important difference between the two families of iterative approaches. Namely, this has to do with the fact that triggers a new iteration in each family of iterative approaches. In our tutorial, we will highlight this fact, by exploring the general framework of iterative entity resolution approaches that are typically composed of an initialization phase and an iterative phase [16]. The goal of the initialization phase is to create a queue capturing the pairs of entity descriptions that will be compared, or even the order for comparing these pairs. For example, such a queue can be constructed automatically by exhaustively computing the initial similarity of all pairs of descriptions, or can be handled manually by domain experts, who specify, for instance, which entities will be compared. In the iterative phase, we get a pair of descriptions from the queue, compute the similarity of this pair to decide if it is a match, and with respect to this decision, we potentially update the queue. Actually, the updates in this phase trigger the next iteration of entity resolution.

In merging-based approaches (e.g., [2], [14]), when two matching descriptions are merged, the pairs in the queue in which these descriptions participate are updated, replacing the initial descriptions with the results of their merging. New pairs may be added to the queue as well, suggesting the comparison of the new merged description to other descriptions. In relationship-based approaches (e.g., [3], [24], [4]), when related descriptions are identified as matches, new pairs can be added to the queue, e.g., a pair of building descriptions is added to the queue, when their architects are matching, or even existing pairs can be re-ordered. Ideally, the iterative phase terminates when the queue becomes empty.

Recent works have proposed using an iterative ER process, interleaved with blocking. The intuition is that the ER results of a processed block, may help identifying more matches in another block. Specifically, iterative blocking [27], applied on the results of blocking, examines one block at a time looking for matches. When a match is found in a block, the resulting merging of the descriptions that match is propagated to all other blocks, replacing the initial matching descriptions. This way, redundant comparisons between the same pair of descriptions in different blocks are saved and, in addition, more matches can be identified efficiently. The same block may be processed multiple times, until no new matches are found. Inherently, iterative blocking follows a sequential execution model, since the results of ER in one block directly affect other blocks.

IV. Progressive Entity Resolution

Progressive works focus on maximizing the reported matches, given a limited computing budget, by potentially exploiting the partial matching results obtained so far. Essentially, they extend the typical ER workflow with a scheduling phase, which is responsible for selecting which pairs of descriptions, that have resulted from blocking, will be compared in the entity matching phase and in what order. The goal of this new phase is to favor more promising comparisons, i.e., those that are more likely to result in matches. This way, those comparisons are executed before less promising ones and thus, more matches are identified early on in the process. The optional update phase (Fig. 1) propagates the results of matching, such that a new scheduling phase will promote the comparison of pairs that were influenced by the previous matches. This iterative process continues until the pre-defined computing budget is consumed.

In a progressive ER setting, [1] uses a graph in which nodes are description pairs, i.e., candidate matches, and an edge indicates that the resolution of a node influences the resolution of another node. For the scheduling phase, it divides the total cost budget into several windows of equal cost. For each window, a comparison schedule is generated, by choosing among those whose cost does not exceed the current window, the one with the highest expected benefit. The cost of a schedule is computed by considering the cost of finding the description pairs and the cost of resolving them. Its benefit is determined by how many matches are expected to be found by this schedule, and how useful it will be to declare those nodes as matches, in identifying more matches within the cost budget. In the update phase, after a schedule is executed, the matching decisions are propagated to all the influenced nodes, whose expected benefit now increases and have, thus, higher chances of being chosen by the next schedule. The algorithm terminates when the cost budget has been reached.

Among other heuristics, [26] uses a hierarchy of description partitions. This hierarchy is built by applying different distance thresholds for each partitioning: highly similar descriptions are placed in the lower levels, while somehow similar descriptions in higher ones. By traversing bottom-up this hierarchy until the cost budget has been reached, we favor the resolution of highly similar descriptions. An other heuristic, also presented in [26], relies on a sorted list of description pairs, which can be generated by a blocking key as in sorted neighborhood, and then iteratively considers sliding windows of increasing size, comparing descriptions of increasing distance. Starting from a window of size 2, this heuristic favors comparisons of descriptions with more similar values on their blocking keys. [23] extends this heuristic, to capture cases in which matches appear in dense areas of the initial sorting, by adding a scheduling phase that performs a local lookahead - if the descriptions at positions (i, j) are found to match, then the descriptions at (i+1, j) and (i, j+1) are immediately compared, since they have a high chance of matching.

Presenters

Kostas Stefanidis is an Associate Professor at the University of Tampere, Finland. Previously, he worked as a research scientist at ICS-FORTH, Greece, and as a post-doctoral researcher at NTNU, Norway, and at CUHK, Hong

Kong. His research interests lie in the intersection of databases, Web and information retrieval, and include personalized and context-aware data management, recommender systems, and information extraction, resolution and integration. He has co-authored more than 50 papers in peer-reviewed conferences and journals, including ACM SIGMOD, IEEE ICDE and ACM TODS. He is the General co-Chair of the Workshop on Exploratory Search in Databases and the Web (ExploreDB), and served as the Web & Information Chair of SIGMOD/PODS 2016, and the Proceedings Chair of EDBT/ICDT 2016. He has also co-authored a book on entity resolution in the Web of data.

Vassilis Christophides is a Professor of Computer Science at the University of Crete. He has been recently appointed to an advanced research position at INRIA Paris-Rocquencourt. Previously, he worked as a Distinguished Scientist at Technicolor, R&I Center in Paris. His main research interests include Databases and Web Information Systems, as well as Big Data Processing and Analysis. He has published over 130 articles in high-quality international conferences and journals. He has been scientific coordinator of a number of research projects funded by the European Union and the Greek State. He has received the 2004 SIGMOD Test of Time Award and 2 Best Paper Awards (ISWC 2003 and 2007). He served as General Chair of the EDBT/ICDT Conference in 2014 and as Area Chair for the ICDE Semi-structured, Web, and Linked Data Management track in 2016. He has also co-authored a book on entity resolution in the Web of data.

Vasilis Efthymiou is a Ph.D. candidate at the University of Crete, Greece, and a member of the Information Systems Laboratory of the Institute of Computer Science at FORTH. The topic of his Ph.D. research is entity resolution in the Web of data. He has been an intern at IBM Thomas J. Watson Research Center, and he has received undergraduate and postgraduate scholarships from FORTH, working in the areas of Semantic Web, non-monotonic reasoning, and Ambient Intelligence. He has also co-authored a book on entity resolution in the Web of data.

REFERENCES

[1] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. *PVLDB*, 7(11):999–1010, 2014.

[2] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1):255–276, 2009.

[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *TKDD*, 1(1), 2007.

[4] C. Böhm, G. de Melo, F. Naumann, and G. Weikum. LINDA: distributed web-of-data-scale entity matching. In *CIKM*, 2012.

[5] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, 2006.

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

[7] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Trans. Knowl. Data Eng.*, 24(9):1537–1555, 2012.

[8] 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.

[9] X. L. Dong and D. Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015.

[10] 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.

[11] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, and T. Palpanas. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Inf. Syst.*, 65:137–157, 2017.

[12] V. Efthymiou, K. Stefanidis, and V. Christophides. Big data entity resolution: From highly to somehow similar entity descriptions in the Web. In *IEEE Big Data*, 2015.

[13] V. Efthymiou, K. Stefanidis, and V. Christophides. Benchmarking blocking algorithms for web entities. *IEEE Trans. Big Data*, 3, 2017.

[14] L. Galárraga, G. Heitz, K. Murphy, and F. M. Suchanek. Canonicalizing open knowledge bases. In *CIKM*, 2014.

[15] O. Hassanzadeh. *Record Linkage for Web Data*. PhD thesis, 2013.

[16] M. Herschel, F. Naumann, S. Szott, and M. Taubert. Scalable iterative graph duplicate detection. *IEEE Trans. Knowl. Data Eng.*, 24(11):2094–2108, 2012.

[17] R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *WebDB*, 2011.

[18] L. Kolb, A. Thor, and E. Rahm. Dedoop: Efficient deduplication with hadoop. *PVLDB*, 5(12):1878–1881, 2012.

[19] I. Miliaraki, K. Berberich, R. Gemulla, and S. Zoupanos. Mind the gap: large-scale frequent sequence mining. In *SIGMOD*, 2013.

[20] 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.

[21] 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.

[22] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolutionto the next level. *IEEE Trans. Knowl. Data Eng.*, 26(8):1946–1960, 2014.

[23] T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. *IEEE Trans. Knowl. Data Eng.*, 27(5):1316–1329, 2015.

[24] V. Rastogi, N. N. Dalvi, and M. N. Garofalakis. Large-scale collective entity matching. *PVLDB*, 4(4):208–218, 2011.

[25] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *ISWC*, pages 245–260, 2014.

[26] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Trans. Knowl. Data Eng.*, 25(5):1111–1124, 2013.

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

[28] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.*, 36(3):15, 2011.

[29] Y. Zhen, P. Rai, H. Zha, and L. Carin. Cross-modal similarity learning via pairs, preferences, and active supervision. In *AAAI*, pages 3203–3209, 2015.