Noisy-aware Blocking over Heterogeneous Data

Tiago B. Araújo[1,2], Carlos Eduardo Pires[1], and Kostas Stefanidis[2]

[1] Federal University of Campina Grande, Brazil
{tiagobrasileiro@copin,cesp@dsc}.ufcg.edu.br
[2] University of Tampere, Finland
kostas.stefanidis@uta.fi

Abstract. Entity resolution (ER) emerges as a fundamental step to integrate multiple knowledge bases or identify similarities between entities. Blocking is widely applied as an initial step of ER to avoid computing similarities between all pairs of entities. Heterogeneous and noisy data increase the difficulties faced by blocking, since these issues directly interfere the block generation. In this work, we propose a technique capable to tolerate noisy data to extract information regarding the data schema, and generate high-quality blocks. We apply Locality Sensitive Hashing (LSH) to hash the entities values, and enable the generation of high-quality blocks, even with the presence of noise in the values. In the experiments, we highlight that our approach has better effectiveness compared to the state-of-the-art technique, as well as less produced comparisons.

1 Introduction

Entity resolution (ER) in big Web data deals typically with two Vs: *volume*, as it handles large amounts of entities; and *variety*, since different formats are used to represent entities [3]. Beyond the two Vs, we highlight another problem tackled by ER: noisy data, which is commonly characterized by pronunciation/spelling errors and typos in the entities values [5]. In practical scenarios, people are less careful with the lexical accuracy of the content written informally or with some pressure. In ER, noisy data directly impacts the identification of similar entities, since spelling difference in their values determines that two entities, truly similar, are not regarded as similar by the ER task. In this work, the two most common noise on data will be considered: typos and misspelling errors [1].

To deal with the large volume of data handled by ER, blocking techniques are applied. Blocking groups similar entities into blocks and perform comparisons between entities within the same block. The *variety* of data is related to the fact that entities do not share the same schema. In this sense, schema-agnostic blocking techniques (e.g., token blocking) have been proposed to address the variety challenge, since they disregard the schema and consider only the values related to the entity attributes [3]. Among the schema-agnostic techniques, BLAST [7] emerges as one of the most promising technique regarding effectiveness. Although it is a schema-agnostic technique, it exploits schema information based on statistics collected from data, to enhance the quality of the blocks in a metablocking approach. However, the presence of noise in the attribute values

compromises the effectiveness of BLAST, since it relies on the accuracy of the values to exploit the schema information, and generate and prune the blocks.

Our work proposes a noise-aware schema-agnostic blocking method for ER. This a novel technique capable of tolerating noisy data to extract information regarding the schema from the data (i.e., group similar attributes based on the data) and enhance the quality of the generated blocks. The method applies Locality Sensitive Hashing (LSH) to hash the entities attribute values and enable the generation of high-quality blocks (i.e., blocks that contain a significant number of entities with high chances of being considered similar), even with the presence of noise in the values. This technique is evaluated against the state-of-the-art method regarding efficiency and effectiveness, using a real dataset.

2 Noise-aware Schema-agnostic Blocking

Our approach is based on metablocking [2, 6, 4], which exploits blocking information to improve the efficiency gains with a minimum impact on the effectiveness. To this end, metablocking restructures a set of blocks into a new one that involves significantly fewer comparisons, while maintaining the original level of effectiveness. Initially, a schema-agnostic blocking technique, e.g., token blocking, is applied to block the heterogeneous data. Token blocking extracts tokens from the entities attribute values, and creates an individual block for every token that appears in at least two entities. Blocks generated by token blocking result in a big number of redundant comparisons between entities. For this reason, blocks are transformed into a weighted graph, such that each entity is represented by a node and each edge between a pair of nodes infers that the nodes share at least one block in common. Based on the number of blocks in common between a pair of entities, metablocking performs pruning, which aims to discard comparisons between entities with few chances of being considered a correspondence.

Our technique performs in three steps: (i) schema information extraction, (ii) block generation, and (iii) pruning. It receives as input two data sources D_1 and D_2. Each data source is an entity collection $D = \{e_1, e_2, ..., e_n\}$ with attributes $A(D) = \{a_1, a_2, \cdots, a_k\}$. Since entities can follow different schemes, each entity $e \in D$ has a specific attribute set and a value associated to each attribute, denoted by $A_e = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, ..., \langle a_k, v_k\rangle\}$.

In the schema information extraction step, all attributes of the entities in D_1 and D_2 are extracted. All values associated with the same attribute a_i are grouped into a set V_{a_i}, i.e., $V_{a_i} = \bigcup_{e \in D}(v \mid \langle a_i, v\rangle \in A_e)$. So, the pair $\langle a_i, V_{a_i}\rangle$ represents the values associated with a_i. The attributes in D_1 and D_2 are grouped based on the similarity of their values: $G(D_1, D_2) = \{g_1, g_2, ..., g_m\} \mid \forall g \in G(D_1, D_2) : g \subseteq (A(D_1) \times A(D_2))$ and $\forall g \in G(D_1, D_2), \forall \langle a_i, a_j\rangle \in g : V_{a_i} \simeq V_{a_j}$. The sets V_{a_i} and V_{a_j} are considered similar if $sim(V_{a_i}, V_{a_j}) \geq \Phi$.

In the block generation step, each set of attribute values V (associated with an attribute a) is converted into a hash-signature S (provided by LSH), given by $hash(\langle a, V\rangle) = \langle a, S\rangle$. To compute the similarity of all pairs of attributes, the complexity is $\mathcal{O}(|U_{D_1}| \cdot |U_{D_2}|)$, where $U_{D_1} = \bigcup_{a_i \in A(D_1)}(V_{a_i} \mid V_{a_i} \in \langle a_i, V_{a_i}\rangle)$, $U_{D_2} = \bigcup_{a_j \in A(D_2)}(V_{a_j} \mid V_{a_j} \in \langle a_j, V_{a_j}\rangle)$. This complexity is impractical for semi-structured Web data, since data sources commonly have hundreds of attributes

and millions of attribute values [7]. For this reason, LSH that has a linear cost in relation to the set size, is applied to reduce the dimensionality of these sets, i.e., U_{D_1} and U_{D_2}, targeting at minimizing the complexity to a linear cost [8]. The set of LSH-signatures S ($\langle a, S \rangle$) guide the block generation, since entities with a similar LSH-signature are grouped into the same block. The loose-schema information (i.e., $G(D_1, D_2)$) is applied to the block generation step to avoid that similar LSH-signatures originated from attributes with different semantics (due to the fact that the attributes are not in the same g) being inserted into the same block by the blocking technique. The output of the block generation step is a collection of blocks B: $B = \{b_1, b_2, ..., b_x\},\ such\ that\ \forall b \in B : (e_1 \in b \wedge e_2 \in b) \Leftrightarrow (\exists \langle a_1, a_2 \rangle \in (A(D_1) \times A(D_2)) : \langle a_1, a_2 \rangle \in \bigcup_{g \in G(D_1, D_2)} g\ \wedge\ hash(\langle a_1, v_1 \rangle \in A_{e_1}) \sim hash(\langle a_2, v_2 \rangle \in A_{e_2}))$.

Finally, in the pruning step, metablocking discards comparisons between entities with low-weight edge, representing low similarity. In this sense, the collection B provided by block generation is restructured relying on the intuition that the more blocks two entities share, the more likely they result in a correspondence. The output of the pruning step is a restructured collection of blocks B'.

3 Experiments

We evaluate our approach[3] against BLAST [7], the state-of-the-art method. We used the IMDB (27,615 entities, four attributes) vs. DBpedia (23,182 entities, seven attributes) datasets with movies provided by imdb.com and dbpedia.org. For effectiveness, we apply: Pair Completeness (PC, similar to recall), Pair Quality (PQ, similar to precision), and F-Measure (FM, harmonic mean between PC and PQ) [7]. For efficiency, we measure the execution time of all steps of the techniques, and the number of comparisons for all generated blocks.

To evaluate effectiveness, we synthetically insert typos and misspellings (i.e., noise) into the attribute values. In order to simulate the occurrence of typos/misspellings [5], one character of each token (i.e., relevant words) present in the attribute values is randomly exchanged by other characters, or additional characters are inserted into the tokens. For instance, a token "snow" can be modified to "sn0w". In this sense, we vary the level of noise in the dataset in the interval 0 (i.e., no noise is inserted into the values) and 1 (i.e., noise is inserted into the values of all entities). For instance, noise level 0.3 indicates that 30% of the entities (contained in the first dataset) had their values modified.

Effectiveness: Figure 1 illustrates the results of our analysis. For all effectiveness measures, our approach outperforms BLAST for all variations of noise level. It is important to highlight that, as the noise level increases, the effectiveness metrics decrease for both techniques. This decrease occurs due to the noise on the data that negatively interferes the block generation. However, this decrease for BLAST occurs abruptly when compared to our approach. Since the latter applies strategies to tolerate noisy data, the effectiveness decrease is amortized.

For FM, we reach an average of 23% less than BLAST in terms of proportional decrease ($1 - \frac{FM(noise=1.0)}{FM(noise=0.0)}$). Our most significant result achieved for PQ was in

[3] https://bitbucket.org/tbrasileiro/na-blocker

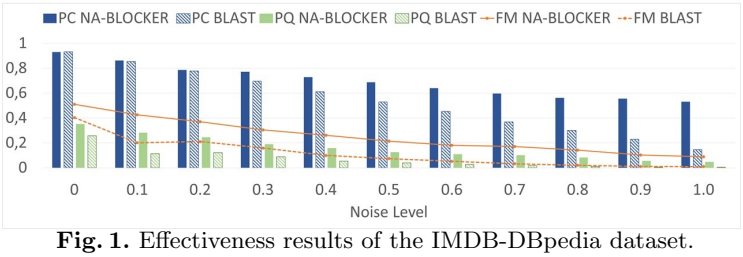

Fig. 1. Effectiveness results of the IMDB-DBpedia dataset.

the scenario without noise, which is 27% better than BLAST. The main reason is the generation of multiple tokens per attribute (based on a particular attribute value) as blocking keys, in BLAST. Since non-matching entities eventually share multiple tokens, they are included in the same block erroneously. On the other hand, our technique generates a single hash value for each particular value. Thus, non-matching entities sharing the same hash value are harder to occur than non-matching entities sharing common tokens. This is why we enhance PQ. Based on these results and the pair-wise (considering the noisy level) distribution *T-Student* test (with confidence 95%), our technique achieves better effectiveness.
Efficiency: Regarding efficiency (we omit the figures due to space limitations), BLAST achieves better results than our approach. On average, there is an around 38% increase on the execution time, as expected, due to the fact that our technique needs more time to generate the LSH-signatures and determine the similarity of values based on the approximate similarity. On the other hand, our technique produces less comparisons to be executed in the ER task. On average, the generated blocks indicate a total number of comparisons around 36% less when compared to the blocks generated by BLAST. Thus, the efficiency results achieved may be compensated by efficiency gains generated by the execution of fewer comparisons between entities to be performed at the following ER tasks.

References

1. S. Agarwal, S. Godbole, D. Punjani, and S. Roy. How much noise is too much: A study in automatic text classification. In *ICDM*, 2007.
2. T. B. Araújo, C. E. S. Pires, and T. P. da Nóbrega. Spark-based streamlined metablocking. In *ISCC*, 2017.
3. V. Christophides, V. Efthymiou, and K. Stefanidis. Entity resolution in the web of data. *Synthesis Lectures on the Semantic Web*, 5(3), 2015.
4. 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.
5. H. Liang, Y. Wang, P. Christen, and R. Gayler. Noise-tolerant approximate blocking for dynamic real-time entity resolution. In *PAKDD*, 2014.
6. G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolutionto the next level. *IEEE TKDE*, 26(8):1946–1960, 2014.
7. G. Simonini, S. Bergamaschi, and H. Jagadish. Blast: a loosely schema-aware meta-blocking approach for entity resolution. *Proceedings of the VLDB Endowment*, 9(12):1173–1184, 2016.
8. J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.