Information Fusion 53 (2020) 103–111

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full Length Article

Scalable entity resolution for Web product descriptions

Damir Vandic[a], Flavius Frasincar[a,*], Uzay Kaymak[b], Mark Riezebos[a]

[a] Erasmus University Rotterdam, Rotterdam, PO Box 1738, 3000 DR, the Netherlands
[b] Eindhoven University of Technology, Eindhoven, PO Box 513, 5600 MB, the Netherlands

ARTICLE INFO

Keywords:
Blocking schemes
Entity resolution
Web shop aggregators
E-commerce

ABSTRACT

Consumers are increasingly using the Web to find product information and make online purchases. This is reflected by the ongoing growth of worldwide e-commerce sales figures. Entity resolution is an important task that supports many services that have arisen from this growth, such as Web shop aggregators. In this paper, we propose a scalable framework for multi-source entity resolution. Our blocking approach employs model words to produce blocks that make our solution highly effective and efficient for the considered domains. An in-depth evaluation, performed using millions of experiments and three large datasets (on consumer electronics and software products), shows that our model words-based approach outperforms other approaches in most cases. Furthermore, we also evaluate our approach with an imperfect similarity function and find that model words-based blocking schemes provide the best blocks with respect to the F_1-measure.

1. Introduction

Over the last few years we have experienced a tremendous growth of online shopping. According to a recent report from Forrester Research, e-commerce spending in the United States will hit approximately $414 billion in 2018 [1], which is 11% of the estimated total retail sales in the same year. To keep up with this growth, various online aggregation services have arisen that allow the user to search for products and their reviews across multiple Web shops. One of the main tasks of such a multi-data fusion service is *entity resolution* [2], which is followed, in specialized contexts, by sentiment aggregation [3–5]. Entity resolution can be described as the process of matching entity descriptions, between two or more data sources that describe the very same real-world entities [6]. Performing this task on the Web is harder than in traditional, relational databases. For example, on the Web, we have to deal with highly heterogeneous data and loosely defined schema's. Furthermore, with Web data there are usually many more data sources than with relational databases, e.g., an online aggregation application needs to aggregate information from many different shops [7]. To the best of our knowledge, these characteristics make our approach unique with respect to other solutions present in the literature.

Various approaches for product entity resolution on the Web have been studied [8–10]. The focus of the current work is to investigate how the scalability of such entity resolution approaches can be improved using *blocking* schemes. Blocking schemes are used to reduce the number of pair-wise similarities that need to be computed, by assigning each product to one or more 'blocks' (i.e., clusters) and only computing the pair-wise similarity for pairs of products that have at least one block in common [11]. Differently than in our previous work [12,13] where we investigated the use of Locality-Sensitive Hashing [14] on a segment of product descriptions for detecting potential duplicates, in this paper, we investigate which blocking schemes work best with product description data on the Web for some specific domains and which part of a product description is the most valuable for the blocking process (i.e., the title or the description). We also focus on multi-source entity resolution, since existing methods in duplicates detection [15] suffer from the similarity transitivity problem. For example, A and B can be considered similar, as well as A and C, but B and C not. Our solution to this problem is an adaptation of hierarchical clustering using single linkage. Furthermore, we perform an in-depth analysis of the trade-off between blocking schemes that result in computing very few pair-wise similarities (more efficient approaches) versus ones with very high recall (more effective approaches).

The novelty of this study stems from several aspects. First, we propose a scalable framework for entity resolution operating on multiple sources at the same time (i.e., multiple Web shops). To our knowledge being able to combine blocks using logical operators is a specific feature of our framework. Second, we propose to use *model words*, previously employed for similarity matching [10], to create effective and efficient blocking schemes for the considered domains. Third, our findings give insight in which blocking schemes work well with product description data on the Web. Last, using millions of experiments, we perform an in-depth evaluation of our blocking approach, under the assumption of a perfect matching function, but, different from related blocking

* Corresponding author.
E-mail addresses: vandic@ese.eur.nl (D. Vandic), frasincar@ese.eur.nl (F. Frasincar), u.kaymak@tue.nl (U. Kaymak), riezebos@ese.eur.nl (M. Riezebos).

https://doi.org/10.1016/j.inffus.2019.06.002
Received 7 July 2018; Received in revised form 12 May 2019; Accepted 1 June 2019
Available online 3 June 2019

scheme studies [16], also using a non-perfect, but well-performing, matching function [10].

The structure of the paper is as follows. First, we discuss the related work in Section 2. In Section 3, we describe a scalable framework for multiple-source entity resolution. Next, in Section 4, we evaluate the performance of our blocking approaches and benchmark against a state-of-the-art solution that does not apply blocking. Last, in Section 5, we conclude and identify possible further research.

2. Related work

Entity resolution is a well-studied research topic [17–19]. Elmagarmid et al. [15] and Christen et al. [20] give overviews for this field and its scalability issues. In the literature, we can find several proposed solutions to scale entity resolution approaches, blocking schemes being the most commonly employed approaches [20]. Based on their focus, blocking schemes can be categorized into two broad categories [11], i.e., methods focusing on *block building* and methods focusing on *block transformations*.

Block building methods focus on producing blocks such that the number of detected duplicates is high and the number of required comparisons is low. A straightforward way of blocking in situations where a schema is available, is to simply group the entities based on the values of selected key(s) [20]. For example, persons stored in a database might be grouped based on postal code, greatly reducing the number of comparisons. With Web data, there is usually no predefined schema, so it is not possible to preselect particular 'keys' of entity descriptions. Instead, what is often done, is to extract *all values*, usually accompanied with a particular filter (e.g., a stop words filter) to reduce the number of comparisons [11,20–22].

In the literature, we can find various approaches that build on the basic approach of representing every entity by one or more keys and creating groups based on their value equality, where pair-wise comparisons are only made within these groups [21,22]. Suffix array approaches [23] employ suffixes of the extracted values (of fixed lengths), where entities that share a suffix of a value are placed in the corresponding block. Bigrams and q-gram blocking [16,24] work in a similar way, only instead of using suffixes, these methods create clusters of entities that share at least one bi- or q-gram of a value. The Sorted Neighborhood approach [25] sorts all extracted values and uses a sliding window of configurable size to create clusters iteratively (at each time moving the window one position down the sorted list). Canopy clustering [26] involves the use of a cheap approximate distance measure to divide the data in overlapping subsets.

Usually the above presented methods perform better when combined with a block transformation step. Block transformation methods aim to improve the effectiveness and/or efficiency by analyzing the produced blocks and transforming them into new ones. One approach that falls into this category is the iterative blocking approach [27,28].

Some approaches focus on scaling existing block building techniques, such as [29]. The authors of this work propose a map-reduce algorithm for executing the sorted neighborhood approach using multiple passes. In this work, we do not aim to parallelize existing approaches in order to be able to handle larger datasets. Instead, we are interested in blocking schemes that are highly scalable by design. This means that the blocking operation must have linear time complexity in terms of the number of input descriptions. In particular, we focus on methods that output blocks based on solely a single input description, i.e., we do not focus on methods that require the knowledge of the whole dataset. The reason for this is that this makes the approach easily parallelizable, as the block generation can be distributed across nodes in a cluster and the pairs (and corresponding blocks) can be efficiently collected (either offline or directly using a hash-based reduce phase). In order to keep the evaluation fair, we therefore disregard methods such as the Sorted Neighborhood approach [25] and Canopy clustering [26]. Furthermore, a common drawback of all the previously discussed methods is that they do not focus on product descriptions. For our proposed approach, we experiment with a token extraction technique (i.e., *model words*) that proved to work well for similarity computations between product descriptions [10]. Last, different from the discussed solutions, which focus on the two-source entity resolution problem, we address in this paper the multi-source entity resolution problem (i.e., handle descriptions of multiple Web shops at the same time).

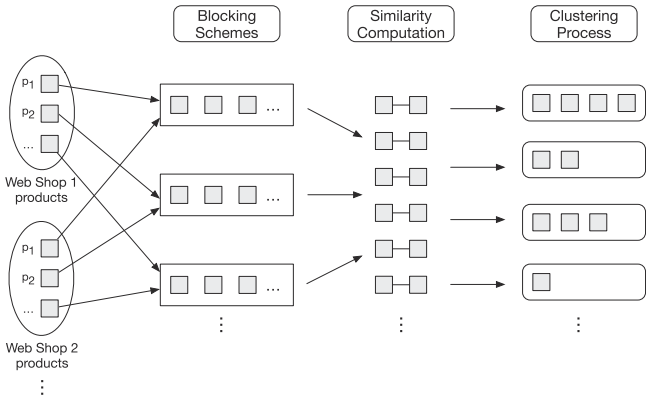

Fig. 1. An overview of the high-level steps in our entity resolution framework.

3. Product entity resolution

The proposed framework in this study entails three main steps, highlighted in Fig. 1. First, in Section 3.1, a blocking scheme is applied in order to reduce the number of pair-wise similarities that have to be computed. Second, in Section 3.2, a product similarity function is applied to the relevant product pairs, i.e, the pairs that have at least one block in common. Third and last, in Section 3.3, a clustering procedure is applied to identify duplicates.

There are three assumptions that we make in this work. First, the input is a collection of product descriptions from at least two Web shops. These product descriptions are represented by a title and a set of key/value pairs representing features of the product. Unlike in traditional database blocking literature, the key/value pairs do not follow any predefined schema, i.e., Web shops can use completely different vocabularies for describing the same set of features. Second, the entity resolution task concerns itself with multiple sources. Here we assume that we know the source of a product description and that within each source (e.g., a Web shop) there are no duplicates. In a two-source ER setting this would be referred to as 'Clean-Clean Entity Resolution' (Clean-Clean ER) task [11]. Third, because most product datasets are of such size that after computing the similarities, an efficient, non-distributed clustering procedure can be employed to uncover the duplicate entries, we focus on the scalability of the pair-wise computations. The reason for this is that, even with moderately sized datasets, the number of pair-wise similarities that need to be computed increases quadratically.

3.1. Blocking schemes

We define a blocking scheme as a function that provides a many-to-many mapping between a product description and a *block*. A block is an entity which is represented by (a part of) a value from the entity description.

The set of possible blocks is determined by the employed blocking scheme. In our approach, a blocking scheme consists of three components:

- the *source selector*;
- the *text tokenizer*;
- and an optional *block transformer*.

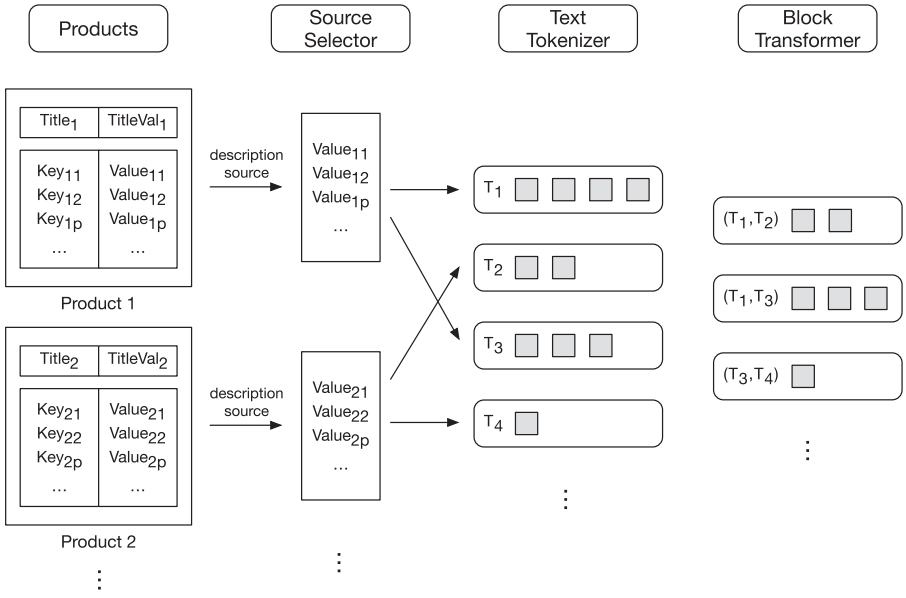

Fig. 2. A more in-depth look into the details of our blocking approach. Shaded squares represent product descriptions. $T_1, T_2, \ldots, T_n$ represent tokens extracted from a source.

Fig. 2 shows the details of our blocking framework. First, the source selector component determines from which part of the product description blocks are extracted. Second, the tokenizer component takes as input, from the source selector component, one or more strings, and outputs the union of the extracted blocks from each of these strings. Third, and last, the optional block transformer component transforms the produced blocks B into a new set of block B', with the goal of optimizing the efficiency of the previously produced blocks.

As previously mentioned, we assume that the product descriptions consist of a title and a set of key/value pairs. In this study, we therefore consider a *title source* and a *key/value pairs source*, where the latter will be referred to as the *description source* for brevity. The description source selector refers solely to the values of the key/value pairs; in our study we ignore the keys of the product descriptions. The reason for this is that these are usually not very discriminative, i.e., descriptions across Web shops have many keys in common. On the other extreme, the ones that are not that common, occur rather infrequently and are therefore not very useful.

For the text tokenizers we first consider the *word* tokenizer. This tokenizer first cleans the input string(s) by removing all punctuation characters (e.g., apostrophe, brackets, colons, comma's, etc.) and the dollar sign character, which might prevent some words from being tokenized properly. Second, the resulting string(s) are cleaned by lowercasing all letters. In the literature, this kind of tokenizer is very common in unstructured datasets, where the schema is unknown and the used values are highly heterogeneous [11]. For example, the title "Philips 4000 Series, 29" - Best Buy" would result in the following extracted blocks: `philips`, `4000`, `series`, `29"`, `best`, `buy`. Besides the regular *word* tokenizer we also consider the q-gram (in particular, trigram) tokenizer [16,24] and suffix array tokenizers [23]. These tokenizers use a scheme that performs the tokenizing technique on words extracted from the input string.

The titles of product descriptions can contain a lot of information for products with many technical specifications. For example, for TV's, the titles often contain terms like `1080p` and `100hz`. We refer to such terms as *model words* and they are defined as words that have at least one digit character and at least one character that is not a digit. The following are examples of extracted model words:

- `Philips 4000 Series29" Class LED 720p 60Hz HDTV 29PFL4508F7` yields model words `720p`, `60hz`, `29pfl4508f7`, `29"`
- `Panasonic -42" Class / Plasma / 720p / 600Hz / HDTV` yields model words `720p`, `42"`, `600hz`

It has been shown that these model words can be effectively used to obtain high accuracy entity resolution [10]. We therefore propose the *model words tokenizer* that extracts these model words from the input string(s). For the previous example ("Philips 4000 Series, 29" - Best Buy"), this would mean that we only extract the following block: `29"`.

Last, for our approach, we consider the use of a *combinations* block transformer. This transformer takes as input the blocks from a tokenizer and outputs all the k combinations (sets of k elements) of these blocks. Again, with the previous example and a particular tokenizer, a $k = 2$ *combinations* block transformer would yield all pairs of the blocks that the tokenizer outputted. With $k = 3$, it would yield all triplets of the blocks that the tokenizer outputted. This means that for the example `Panasonic -42" Class / Plasma / 720p / 600Hz / HDTV` with model words `720p`, `42"` and `600hz`, a $k = 2$ combinations block transformer would yield {`720p`, `42"`}, {`720p`, `600hz`} and {`42"`, `600hz`}. This transformation will result in a reduction of the total number of similarity-pairs that need to be computed, i.e, the longer the combination, the fewer products will match the combination.

The above discussed components, i.e., the source selector, the text tokenizer, and the optional block transformer components, form together a *blocking scheme*. Once these three components have been chosen, the blocks are generated per given product description. The corresponding product is then mapped to each of the generated blocks.

3.1.1. Blocking schemes aggregators

Similar to the transformers for the blocks, on the level of blocking schemes, we consider *aggregations of blocking schemes* that might improve the effectiveness and/or efficiency. For this purpose, we consider two type of aggregations: (1) the *union* of the output blocks, focusing on improving effectiveness, and (2) the *conjunction* of the output blocks, moving the focus on improving the efficiency of the outputted blocks.

The *union* aggregator blocking scheme takes two blocking schemes and outputs mappings that are given by *either* of the two blocking schemes. For example, if blocking scheme x maps entity $p1$ to block $b1$ and blocking scheme y maps entity $p1$ to block $b2$, then this blocking scheme would map $p1$ to both $b1$ and $b2$.

The *conjunction* aggregator blocking scheme takes two blocking schemes, say $s1$ and $s2$, and combines their outputs in the following

way. First, the mappings are computed using $s1$. Then, for each of the blocks in the mapping, the entities mapped to the block are fetched. The blocking scheme $s2$ is called on these entities, yielding a new set of mappings. The final outputted mappings are a conjunction of the blocks from $s1$ and the blocks from $s2$.

Finally, we also consider the *all pairs blocking scheme*, which is actually a scheme that does not perform blocking at all, it considers all pairs. The other approaches are evaluated w.r.t. the percentage of pairs computed compared to the all pairs method.

3.1.2. Blocking schemes identifiers

Throughout this paper we adhere to a blocking scheme naming convention that represents a combination of the previously discussed components. A blocking scheme is represented in the form of `$source.$tokenizer$transformer`, where the `$transformer` is optional (words that start with a dollar sign represent a variable). For `$source`, we use `t` for the title source and `d` for the description source (recall that for the description source, keys are ignored, such that only the key-related values are used). We use `wo` for the word tokenizer, `3q` for the qgram (trigram) tokenizer, `sx` for the suffix array tokenizer, and `mw` for the model words tokenizer. For example, `t.mw` would represent a blocking scheme that extracts model words from the title and `d.wo` a blocking scheme that extracts words from the description source. Additionally, when a combinations block transformer is applied, we append k to the name of the tokenizer. For example, `d.mw3` represents the blocking scheme that extracts model words from the description source and forms triples of these to output blocks. The union and conjunction block transformers are represented like `a_$operator_b`, where `a` and `b` are blocking schemes and `$operator` can be either `+` (for union transformations) or `x` (for conjunction transformations). For example, `t.mw3_x_d.mw2` represents the conjunctive combination of two blocking schemes that extract model words triples from the title and model word pairs from the description. This means that this model will output only blocks that are both outputted by the two blocking schemes. If the title source would be `"Hisense 42"1080p 60Hz LCD HDTV LTDN42V68US - Newegg.com"` and the description would contain, among others, the key/value pairs `"Response Time": "5ms", "Recommended Resolution": "1080p"`, `t.mw3_x_d.mw2` would yield ({`"42"`", `"1080p"`, `"60Hz"`}, {`"5ms"`, `"1080p"`}). As thus can be seen, model words are only regarded if they are in the title as well as the description, which illustrates the increased efficiency of the conjunction aggregator.

Algorithm 1: MSM Similarity.

Required functions/parameters:

- $\texttt{KeySim}(a, b)$ computes the similarity between two product descriptions a and b based solely on the product attributes (described in Algorithm 2);
- $\texttt{titleSim}(t_a, t_b)$ gives the similarity between title t_a and t_b;
- $\texttt{mw}(W)$ extracts the *model words* for a given set of words W;
- $\texttt{mwSim}(mw_a, mw_b)$ computes the similarity between model words mw_a and mw_b;
- μ is the weight for the title similarity.

```
// Computes similarity between product descriptions a and b
1  def ProdSim(a, b, μ):
2      (m, I, J, keySim*) := KeySim(a, b);
       // m is the number of key matching keys between a and b
       // I is the set of keys from a that do not match
       // J is the set of keys from b that do not match
       // keySim* is in [0,1] and represents the similarity between the keys of a and b
       // extract model words from set of keys that do not match
3      I_mw := mw(I);
4      J_mw := mw(J);
5      mwSim := mwSim(I_mw, J_mw) ;    // similarity between the extracted model words
6      titleSim := titleSim(a.title, b.title) ; // similarity between the titles
7      if titleSim = 0 then
8          θ1 := m/min(|a.keys|, |b.keys|);
9          θ2 := 1 − θ1;
10     else
11         θ1 := (1 − μ) · m / min(|a.keys|, |b.keys|);
12         θ2 := 1 − μ − θ1;
13     end
       // final similarity, consisting of keySim*, mwSim, and titleSim
14     sim* := θ1 · keySim* + θ2 · mwSim + μ ∗ titleSim
15     return sim*
```

3.2. Similarity computation

The approach we employ in this paper for computing the similarities is the state-of-the-art MSM approach, proposed by van Bezu et al. [10]. MSM is able to compute similarities for products that are represented in a semi-structured data model, e.g., tabular Web product information.

Algorithm 1 shows a high-level overview of how the product similarity is computed. In Algorithm 2 we see the function $\text{KeySim}(a, b)$, which is used to compute the similarity between product descriptions a and b based solely on the product attributes. This function is used by Algorithm 1 on line 2 to compute the final product description similarity. The similarity function of MSM takes as input two product descriptions and outputs a similarity between 0 and 1. The main idea of the approach is to first compute a similarity based on the keys of two product descriptions (i.e., attributes). Then, with the remaining, unmatched keys, we compute a similarity based on the proposed *model words*. Last, we compute a title similarity and combine the three similarities into one final weighted average similarity. For the titleSim() function we employ a cosine-based similarity. The keySim() and valSim() functions are based on the q-gram similarity measure. Furthermore, the keysMatch() function uses the keySim() function to compute the similarity between two keys, and if the similarity is larger than a threshold, the keys match. For more details we refer the reader to [10].

3.3. Clustering

After the similarities between the relevant entity pairs have been computed, we apply a clustering step to discover the duplicate entities. The reason for this is that we have multiple sources and the similarity function is not transitive. In other words, if one has A sim B, B sim C, then this does not imply A sim C. Using a clustering approach, these transitive similarities can be taken into account.

As part of our framework, the clustering method takes as input the product description similarities and outputs clusters of products in which its members are considered to be duplicates. Because our approach supports multiple sources and we assume that within each source there are no duplicates, the algorithm should not cluster products that are originating from the same Web shop.

The MSM similarity [10] is used in an approach where an adaptation of hierarchical clustering is employed. The main idea of this clustering

Algorithm 2: MSM Key Similarity.

Required functions:

- `clean`(w) removes punctuation characters at end of word w;
- `keysMatch`(k_a, k_b) determines whether keys k_a and k_b match (i.e., represent the same attribute);
- `keySim`(k_a, k_b) gives the similarity between keys k_a and k_b;
- `valSim`(k_a, k_b) gives the similarity between the values of keys k_a and k_b.

```
// Computes the similarity between two descriptions
   using only the attributes
def KeySim(a, b):
    m := 0 // number of matches
    w := 0 // weight of matches
    I := a.keys and J := b.keys // keys without a match
    sim := 0;
    foreach k_{i,a} in a.keys do
        k_{i,a} := clean(k_{i,a});
        foreach k_{j,b} in b.keys do
            k_{j,b} := clean(k_{j,b});
            if keysMatch(k_{i,a}, k_{j,b}) then
                I := I \ k_{i,a};
                J := J \ k_{j,b};
                keySim := keySim(k_{i,a}, k_{j,b});
                valueSim := valSim(k_{i,a}, k_{j,b});
                sim := sim + keySim * valueSim;
                m := m + 1;
                w := w + keySim;
            end
        end
    end
    keySim* := 0;
    if w > 0 then
        keySim* := sim / w;
    end
    return (m, I, J, keySim*)
```

approach is to use single linkage in all cases except for those where a infinite distance is encountered, in which case complete linkage is applied instead. These infinite distances are assigned to product description pairs originating from the same Web shop, thus applying complete linkage in these cases yields an infinite distance measure. Therefore, it is unlikely that the modified linkage criterion in this approach will assign such descriptions into the same cluster.

4. Evaluation

For the evaluation of our framework, we use three datasets:

- *4shops* - this dataset consists of 1624 descriptions of televisions, obtained from Amazon.com, Newegg.com, BestBuy.com, and TheNerds.net;
- *abt-buy* - this dataset consists of 2175 descriptions of various electronics products;
- *amz-ggl* - this dataset consists of 4591 descriptions of various software products.

These datasets are available online.[1] Two examples, the first one for consumer electronics (from Newegg.com) and the second one for software products (from Amazon.com), are as follows:

- "title": "Toshiba - 32\" Class / LED / 720p / 60Hz / HDTV",
"Internet Connectable": "No",
"Remote Control Type": "Standard",
"Screen Size Class": "32\"",
"DVI Inputs": "0"
- "title": "ca international - arcserve lap/desktop oem 30pk",
"description": "oem arcserve backup v11.1 win 30u for laptops and desktops",
"manufacturer": "computer associates",
"price": "0"

First, in Section 4.1, we evaluate the effectiveness and efficiency of the proposed blocking approach. Then, in Section 4.2, we analyze how the performance of the proposed entity resolution algorithm is affected by the considered blocking schemes.

4.1. Blocking evaluation

We evaluate our blocking approach using three measures that are commonly used in literature [11,20,21]. These measures are computed based on the pairs that need to be considered as a result of a blocking scheme. First, we consider a measure of efficiency, which is referred in literature as *pairs quality* (PQ). It is defined as

$$PQ = \frac{\text{found duplicates}}{\text{executed comparisons}}$$

This can be considered as the precision measure from information retrieval. Second, we consider a measure of effectiveness, also known as the *pairs completeness* (PC). Essentially, it is the recall measure from information retrieval and it is defined as

$$PC = \frac{\text{found duplicates}}{\text{total number of duplicates}}$$

Last, we consider the reduction rate (RR), defined as

$$RR = 1 - \frac{\text{executed comparisons with blocking}}{\text{executed comparisons without blocking}}$$

This is the reduction in the percentage of pairs that need to be considered after applying a blocking scheme, compared to the total number of pairs (without applying a blocking scheme). The total number of pairs excludes pairs of product descriptions that are from the same Web shop, as we assume that we are dealing with 'Clean-Clean ER' and that in most contexts, the source of the product description is known.

4.1.1. Considered configurations

These are the configurations that we consider:

1. `t.{wo,mw}{2..4}` and `t.{wo,mw}` (8 combinations)
2. `t.{3q,sx}` (2 combinations)
3. `d.{wo,mw}{2..4}` and `d.{wo,mw}` (8 combinations)
4. `d.{3q,sx}` (2 combinations)
5. `a_+_b` where `a` and `b` are unique pairs from (1), (2), (3), and (4) (190 combinations)
6. `a_x_b` where `a` and `b` are unique pairs from (1) and (3), respectively (120 combinations)

The total number of considered configurations is therefore 330 (20 + 190 + 120).

For each of these methods, and the three considered datasets, we computed the previously discussed measures on 40 bootstraps, where each bootstrap is a stratified random sample of the original data to obtain samples that are representative for the original data set (and to reduce sampling error and variance). Next, we only kept methods that had an average PC higher than 0.50 and an average RR measure greater than 0.50 (measured across datasets). For these methods we then compute the average PC and average RR for each dataset. Then, for each dataset, we

[1] http://damirvandic.com/wp-content/uploads/2016/05/TVs-all-merged.zip and http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution.

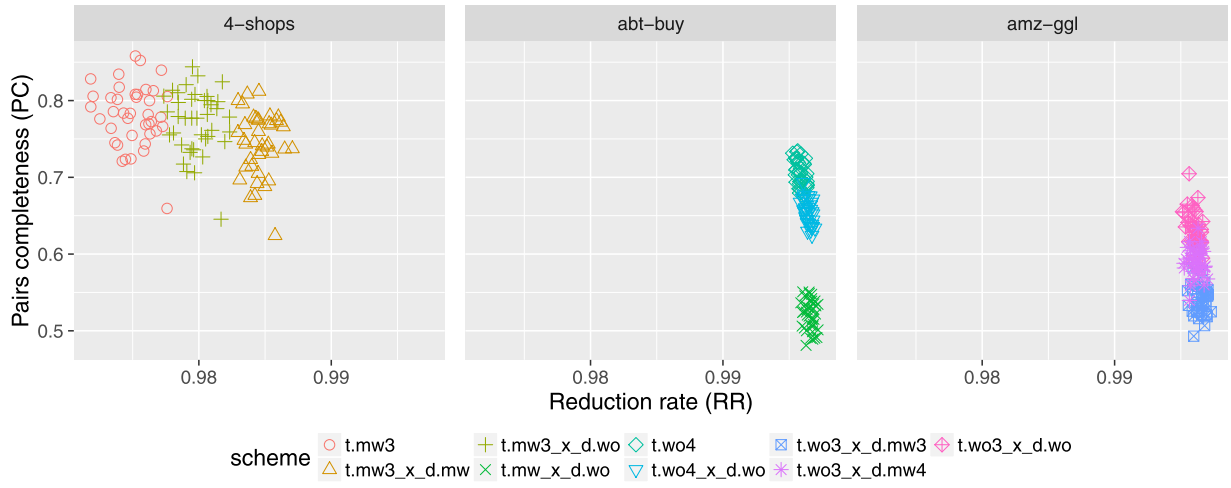

Fig. 3. Scatter plots for the 'low' PC category blocking methods showing the trade-off between the reduction rate (RR) and pairs completeness (PC), where each point represents one of the 40 bootstraps.

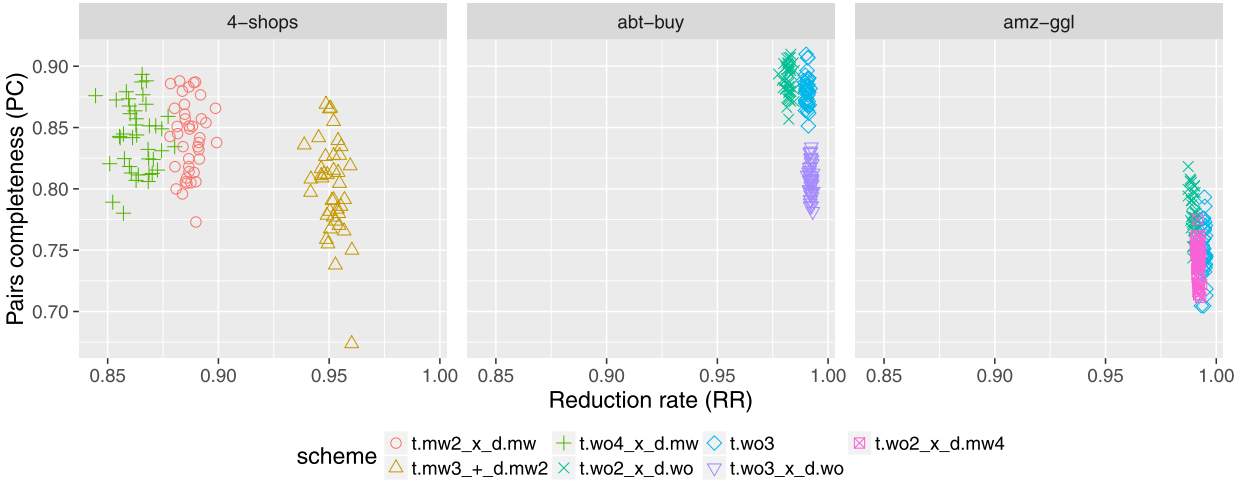

Fig. 4. Scatter plots for the 'medium' PC category blocking methods showing the trade-off between the reduction rate (RR) and pairs completeness (PC), where each point represents one of the 40 bootstraps.

Table 1
Blocking evaluation results for the considered blocking methods in '**low** category' (w.r.t to the PC measure). For each measure, we aggregated the scores achieved over the 40 bootstraps. Each cell shows (1) the mean and (2) the standard deviation. The best values per column for each dataset are given in bold font.

Scheme	dataset	PC	RR	PQ	Block count	Blocking duration (ms)
`t.mw3_x_d.mw`	4-shops	0.74 / 0.04	**0.98** / 0.00	**0.07** / 0.01	49331.45 / 2122.70	55.00 / 18.50
`t.mw3_x_d.wo`	4-shops	0.77 / 0.04	**0.98** / 0.00	0.05 / 0.00	134687.58 / 5640.74	144.90 / 12.94
`t.mw3`	4-shops	**0.78** / 0.04	**0.98** / 0.00	0.04 / 0.00	**2539.20** / 98.60	**2.28** / 3.77
`t.mw_x_d.wo`	abt-buy	0.52 / 0.02	**1.00** / 0.00	0.39 / 0.04	**16998.68** / 504.64	**20.75** / 4.58
`t.wo4_x_d.wo`	abt-buy	0.65 / 0.02	**1.00** / 0.00	**0.47** / 0.03	1039981.45 / 62697.68	1441.33 / 217.08
`t.wo4`	abt-buy	**0.71** / 0.02	**1.00** / 0.00	0.44 / 0.03	60376.30 / 7362.27	54.43 / 22.35
`t.wo3_x_d.mw3`	amz-ggl	0.54 / 0.02	**1.00** / 0.00	0.41 / 0.05	**90529.18** / 11833.81	**209.98** / 68.36
`t.wo3_x_d.mw4`	amz-ggl	0.59 / 0.02	**1.00** / 0.00	0.41 / 0.05	133298.35 / 38686.92	262.43 / 103.16
`t.wo3_x_d.wo`	amz-ggl	**0.63** / 0.02	**1.00** / 0.00	**0.42** / 0.04	1151049.95 / 103369.37	1453.68 / 173.81

split the methods into three equally-sized groups ('low', 'medium', and 'high'), which is based on the average PC where 'high' represents the group having the highest PC measure. Finally, for each group in each dataset, we select only the top 3 methods w.r.t. the RR.

Figs. 3–5 show the trade-off between the PC and RR measures for each group and each dataset. Each dot in a graph represents the obtained performance for a particular bootstrap. Furthermore, Tables 1–3 show for each top method in each dataset the mean and standard deviation for PC, RR, PQ, block count, and blocking duration. For the PC measure, we have used paired t-tests with Bonferroni corrected p-values to determine if differences are statistically significant. For sake of brevity, in our discussions, we will only highlight the non-significant differences (because most of the differences are significant).

What we first notice is that the suffix array tokenizers and trigram tokenizers do not appear in the results as top-performing methods. This might suggest that traditional tokenizing techniques are not ef-

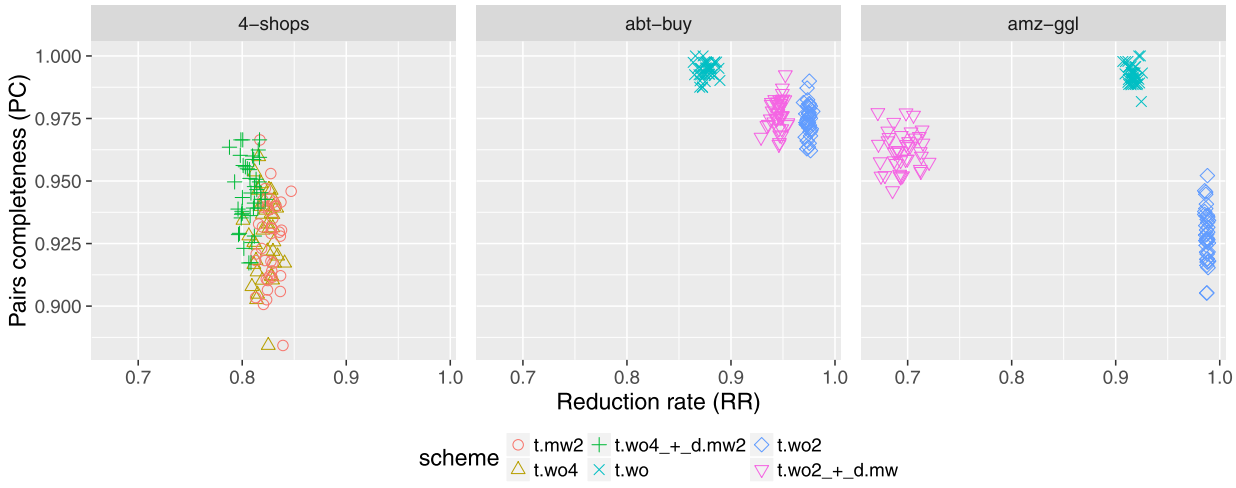

Fig. 5. Scatter plots for the 'high' PC category blocking methods showing the trade-off between the reduction rate (RR) and pairs completeness (PC), where each point represents one of the 40 bootstraps.

Table 2
Blocking evaluation results for the considered blocking methods in '**medium** category' (w.r.t to the PC measure). For each measure, we averaged the scores achieved over the 40 bootstraps. Each cell shows (1) the mean and (2) the standard deviation. The best values per column for each dataset are given in bold font.

Scheme	dataset	PC	RR	PQ	Block count	Blocking duration (ms)
t.mw3_+_d.mw2	4-shops	0.80 / 0.04	**0.95** / 0.00	**0.02** / 0.00	**5268.55** / 171.25	**11.60** / 19.34
t.mw2_x_d.mw	4-shops	**0.84** / 0.03	0.89 / 0.00	0.01 / 0.00	45208.00 / 1527.39	55.78 / 18.08
t.wo4_x_d.mw	4-shops	**0.84** / 0.03	0.86 / 0.01	0.01 / 0.00	2674844.73 / 154913.98	4,064.73 / 607.06
t.wo3_x_d.wo	abt-buy	0.81 / 0.01	**0.99** / 0.00	**0.27** / 0.02	611670.70 / 25527.48	816.03 / 83.14
t.wo3	abt-buy	0.88 / 0.01	**0.99** / 0.00	0.24 / 0.02	**31452.78** / 1760.35	**27.43** / 10.56
t.wo2_x_d.wo	abt-buy	**0.89** / 0.01	0.98 / 0.00	0.13 / 0.01	248044.33 / 7655.92	325.93 / 27.40
t.wo2_x_d.mw4	amz-ggl	0.74 / 0.02	**0.99** / 0.00	0.24 / 0.02	95143.25 / 36259.98	188.58 / 88.68
t.wo3	amz-ggl	0.75 / 0.02	**0.99** / 0.00	**0.36** / 0.04	**29719.25** / 2441.90	**27.58** / 10.95
t.wo2_x_d.wo	amz-ggl	**0.78** / 0.02	**0.99** / 0.00	0.20 / 0.02	537230.35 / 32067.84	688.13 / 165.69

Table 3
Blocking evaluation results for the considered blocking methods in '**high** category' (w.r.t to the PC measure). For each measure, we averaged the scores achieved over the 40 bootstraps. Each cell shows (1) the mean and (2) the standard deviation. The best values per column for each dataset are given in bold font.

Scheme	dataset	PC	RR	PQ	Block count	Blocking duration (ms)
t.mw2	4-shops	0.93 / 0.02	**0.83** / 0.01	**0.01** / 0.00	**1902.48** / 60.47	**2.20** / 3.50
t.wo4	4-shops	0.93 / 0.02	0.82 / 0.01	**0.01** / 0.00	157152.50 / 10788.26	156.70 / 97.48
t.wo4_+_d.mw2	4-shops	**0.95** / 0.01	0.81 / 0.01	**0.01** / 0.00	159881.88 / 10788.41	255.70 / 111.22
t.wo2	abt-buy	0.97 / 0.01	**0.97** / 0.00	**0.10** / 0.01	11140.50 / 385.18	10.70 / 6.74
t.wo2_+_d.mw	abt-buy	0.98 / 0.01	0.95 / 0.01	0.05 / 0.00	12006.83 / 400.41	23.83 / 12.56
t.wo	abt-buy	**0.99** / 0.00	0.88 / 0.01	0.02 / 0.00	**1573.63** / 40.29	**5.93** / 11.33
t.wo2	amz-ggl	0.93 / 0.01	**0.99** / 0.00	**0.20** / 0.02	9706.18 / 450.01	10.28 / 8.47
t.wo2_+_d.mw	amz-ggl	0.96 / 0.01	0.70 / 0.01	0.01 / 0.00	10121.03 / 455.96	71.35 / 55.31
t.wo	amz-ggl	**0.99** / 0.00	0.92 / 0.00	0.03 / 0.00	**1305.78** / 40.24	**4.75** / 11.16

ficient enough for Web data, such as the product descriptions in the three considered datasets. Furthermore, from Figs. 3–5, we can see that for the 'low' group, the methods are all in the upper range w.r.t. the reduction rate (RR), with the lowest average being 0.98 for the '4-shops' dataset. The average PC in this group varies between 0.52 and 0.78 across datasets. All differences w.r.t. the PC measure are found to be statistically significant (across datasets), except the difference between t.mw3_x_d.wo and t.mw3, and t.wo3_x_d.mw3 and t.mw_x_d.wo. Taking into account, besides the PC and RR measures, the number of blocks and the duration, we would argue that, for this group, t.mw3 provides the optimal balance between speed and recall.

For the 'medium' group we can see that there is more variation w.r.t. the reduction rate (RR), ranging from an average of 0.86 to 0.99. For the PC measure, the variation is similar compared to the 'low' group but the range is smaller, with averages going from 0.74 to 0.89. For this group, the difference between the PC measure is for the following methods statistically insignificant (across datasets):

- t.wo4_x_d.mw and t.mw2_x_d.mw
- t.wo2_x_d.wo and t.mw2_x_d.mw
- t.wo2_x_d.wo and t.wo4_x_d.mw
- t.wo3 and t.mw2_x_d.mw
- t.wo3 and t.mw3_+_d.mw2
- t.wo3 and t.wo2_x_d.wo
- t.wo3_x_d.wo and t.mw3_+_d.mw2
- t.wo3_x_d.wo and t.mw3_+_d.mw2
- t.wo3_x_d.wo and t.mw3_+_d.mw2

We can also notice that t.wo4_x_d.mw generates a large number of blocks. This is caused by the fact that the word tokenizer produces many tokens and that the high number of combinations lead to

Table 4
The top-3 best performing methods (w.r.t. the F_1-measure) and the all-pairs method for the entity resolution evaluation, showing the average of each of the measures, taken over the 100 considered bootstraps. The scheme `ap` stands for the scheme that does not apply any blocking but instead just considers all pairs. The best values per column are given in bold font.

Scheme	F_1-measure	Precision	Recall	Mean sample duration (ms)
`t.mw3_x_d.wo`	**0.5382**	0.5411	0.5380	447.15
`t.mw3_x_d.mw`	0.5376	**0.5511**	0.5272	221.68
`t.mw3`	0.5368	0.5389	0.5376	**49.52**
`ap`	0.5248	0.4723	**0.5922**	406.8

an exponential growth in the number of produced blocks. Looking at the average durations for the '4-shops' dataset, we can see that two schemes stand out because of their low duration: `t.mw2_+_d.mw` and `t.mw3_+_d.mw2`. For this group, we notice that the model words tokenizer is part of the best performing schemes for the '4-shops' dataset. For the *amz-ggl* dataset, the word tokenizers seems to achieve better efficiency. This is as expected, because the model words work best with technical descriptions, which are most common among consumer electronics.

In the 'high' group, most methods are able to achieve a relatively high reduction rate, except for the '4-shops' dataset. The reason for this is that there are multiple Web shops and thus, there is more variation in the descriptions. As the results for the other two groups demonstrated, the '4-shops' dataset seems to be the most difficult for the methods to achieve a high reduction rate. With respect to the PQ measure, all methods are found to statistically differ, except for `t.wo4` and `t.mw2`, and `t.wo2` and `t.wo4_._d.mw2`. Given these results, for the '4-shops' dataset there is not a clear winner. For the other two datasets, the scheme `t.wo` seems to give the best performance with respect to PC and RR.

The results from these experiments demonstrate the scalability of our approach. For example, in Table 3, which shows the results for the 'high' group, we can see that the method `t.wo2_+_d.mw` takes on average 71.35 ms for *amz-ggl* and 23.83 ms for *abt-buy* for a bootstrap sample (an increase of factor 2.99). Given the fact that the *amz-ggl* data set has 4591 descriptions and the *abt-buy* data set has 2175 descriptions (a factor 2.11 difference in size), we can see that the increase in blocking duration is very close to linear.

4.2. Entity resolution evaluation

In order to validate the found results in the previous section, we perform another evaluation, i.e., one that assesses the performance of the entity resolution algorithm when a blocking scheme is applied. For this part, we used only the '4-shops' dataset as this dataset has the required characteristics for the MSM algorithm (unstructured key/value data). The other two datasets have only a single description field, making a comparison with these datasets not possible. Different than the blocking evaluation, the entity resolution evaluation is implemented on top of a cluster computing framework, i.e., the Apache Spark framework [30]. Because we had access to a large cluster, we were able to run 100 bootstraps of the MSM algorithm with each of the blocking schemes. For MSM we tried 375 different parameter configurations. Previously, in Section 4.1.1, we have explained which are the scheme configurations that we consider. Using the considered configurations, we arrive at a total number of experiments of 12,375,000 (100 bootstraps $\times$ 375 MSM instances $\times$ 330 blocking scheme configurations). We used a cluster of 640 cores and an additional caching layer for computing the MSM similarities.

Table 4 shows the results of the top-3 methods and the all-pairs method w.r.t. the F_1-measure using the MSM matching function. We found that only the methods `t.mw3_x_d.wo` and `t.mw3` are statistically different w.r.t the F_1. With respect to the average duration per bootstrap sample, we can report that all measured mean differences are statistically significant, except for the difference between `t.mw3_x_d.wo` and `ap`. From this we can conclude that the top-2 methods exhibit the same performance, and that, considering the relatively small difference between the top-3 methods, one can expect a relatively good performance from all top-3 methods (`t.mw3_x_d.wo`, `t.mw3_x_d.mw`, and `t.mw3`).

It is interesting to see that the all pairs scheme does not give the best performance and that the best performing methods do not come from the group that scores the highest on PC. At first, this seems counterintuitive, however, there is a reasonable explanation for this. It seems that the blocking schemes are able to factor a significant amount of pairs that would lead to false positives. This is also reflected by the increased precision. Apparently, the decrease in recall (due to the computation of less pairs) is less than the increase in precision, and thus resulting in a higher F_1-measure. This is a very interesting observation since most studies on blocking schemes are performed with the assumption of the availability of a perfect matching function. The found results in these experiments suggest that with imperfect matching functions, such as MSM, the best performing methods are not necessary the ones with the highest PC measure.

Further experiments are needed to generalize the findings regarding the contribution of the considered tokenizers. However, the results from our experiments do suggest interesting research directions. For example, the top-3 methods all utilize the model words tokenizer. This might suggest that for product descriptions containing technical specifications, such as the '4-shops' dataset, model word tokenizers are more suitable than word tokenizers.

5. Conclusion

In this paper, we have proposed a scalable approach for multi-source entity resolution using various blocking schemes. Our approach consists of three main components: (1) a blocking scheme, (2) a product similarity function, and (3) a clustering procedure. Various blocking schemes, which operate on the title, the description, or both, are evaluated with a perfect similarity function, as well as an imperfect, but well-performing, one [10]. The framework has been evaluated using an extensive set of experiments and three large datasets used commonly in entity resolution studies.

The most important finding with regards to the blocking evaluation is that experimental setups that evaluate blocking schemes with only a perfect matching function are not sufficient. The results suggest that for our similarity function and clustering procedure, which together achieve an F_1-measure of 0.525 when using all pairs, blocking methods that compute very few pairs achieve a higher performance than ones that have a high PC. In particular, we find that the blocking scheme that extract model word triples from the title (i.e., `t.mw3`) gives the best trade-off between effectiveness and efficiency on our test dataset, achieving an F_1-measure of 0.537. The higher F_1-measure, compared to the F_1-measure of 0.525 for all pairs, suggests that the employed similarity function is sensitive to the number of pairs it needs to consider.

The blocking approach used in this paper has proved to make our solution highly effective and efficient for the considered domains. However, this blocking approach can be improved. For example, recent work

on zero-shot entity typing [31–33] can be considered to detect to which category an entity description corresponds, such as computers, televisions, tablets, audio, video games, or another category (since our focus is on consumer electronics and software products). In this way, the number of pair-wise similarities that need to be computed can be further reduced, and, in addition, this could benefit the accuracy of duplicates detection as the comparisons are performed only per product type.

Acknowledgment

Sophie Eulderink, Sylvie Kok, Dorenda Slof, and Corinca Zwijsen contributed in the early stages of this work. The experiments were run on a Hadoop cluster of the Dutch national e-infrastructure, with the support of SURF Foundation. Damir Vandic is supported by an NWO Mosaic scholarship for project 017.007.142: *Semantic Web Enhanced Product Search (SWEPS)*.

References

[1] S. Levien, Forrester: US Mobile and Tablet Commerce to Top $293B by 2018, Total Ecommerce to Hit $414, 2014. http://goo.gl/O1R4dS.
[2] G. Nachouki, M. Quafafou, Multi-data source fusion, Inf. Fus. 9 (4) (2008) 523–537.
[3] S. Poria, E. Cambria, R. Bajpai, A. Hussain, A review of affective computing: from unimodal analysis to multimodal fusion, Inf. Fus. 37 (2017) 98–125.
[4] I. Chaturvedi, E. Cambria, R.E. Welsch, F. Herrera, Distinguishing between facts and opinions for sentiment analysis: survey and challenges, Inf. Fus. 44 (2018) 65–77.
[5] A. Valdivia, M.V. Luzon, E. Cambria, F. Herrera, Consensus vote models for detecting and filtering neutrality in sentiment analysis, Inf. Fus. 44 (2018) 126–135.
[6] G. Costa, G. Manco, R. Ortale, An incremental clustering scheme for data de-duplication, Data Min. Knowl. Discov. 20 (1) (2010) 152–187.
[7] L.J. Nederstigt, S.S. Aanen, D. Vandic, F. Frasincar, FLOPPIES: A framework for large-Scale ontology population of product information from tabular data in e-commerce stores, Decis. Support Syst. 59 (2014) 296–311.
[8] M. de Bakker, F. Frasincar, D. Vandic, A hybrid model words-driven approach for web product duplicate detection, in: Proceedings of the 25th International Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science, 7908, Springer, 2013, pp. 149–161.
[9] M. de Bakker, D. Vandic, F. Frasincar, U. Kaymak, Model words-driven approaches for duplicate detection on the web, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM, 2013, pp. 717–723.
[10] R. van Bezu, S. Borst, R. Rijkse, J. Verhagen, F. Frasincar, D. Vandic, Multi-component similarity method for web product duplicate detection, in: Proceedings of the 30th Annual ACM Symposium On Applied Computing, ACM, 2015, pp. 761–768.
[11] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, W. Nejdl, A blocking framework for entity resolution in highly heterogeneous information spaces, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2665–2682.
[12] I. van Dam, G. van Ginkel, W. Kuipers, N. Nijenhuis, D. Vandic, F. Frasincar, Duplicate detection in web shops using LSH to reduce the number of computations, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 772–779.
[13] A. Hartveld, M. van Keulen, D. Mathol, T. van Noort, T. Plaatsman, F. Frasincar, K. Schouten, An LSH-based model-words-driven product duplicate detection method, in: Proceedings of the 30th International Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science, 10816, Springer, 2018, pp. 409–423.
[14] P. Indyk, R. Motwani, Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality, in: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, 1998, pp. 604–613.
[15] A.K. Elmagarmid, P.G. Ipeirotis, V.S. Verykios, Duplicate record detection: a survey, IEEE Trans. Knowl. Data Eng. 19 (1) (2007) 1–16.
[16] R. Baxter, P. Christen, T. Churches, A comparison of fast blocking methods for record linkage, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 3, ACM, 2003, pp. 25–27.
[17] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S.E. Whang, J. Widom, Swoosh: a generic approach to entity resolution, Int. J. Very Large Data Bases 18 (1) (2009) 255–276.
[18] X. Dong, A. Halevy, J. Madhavan, Reference reconciliation in complex information spaces, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 2005, pp. 85–96.
[19] N. Koudas, S. Sarawagi, D. Srivastava, Record linkage: similarity measures and algorithms, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 2006, pp. 802–803.
[20] P. Christen, A survey of indexing techniques for scalable record linkage and deduplication, IEEE Trans. Knowl. Data Eng. 24 (9) (2012) 1537–1555.
[21] G. Papadakis, G. Alexiou, G. Papastefanatos, G. Koutrika, Schema-agnostic vs schema-Based configurations for blocking methods on homogeneous data, Proc. VLDB Endow. 9 (4) (2015) 312–323.
[22] G. Simonini, G. Papadakis, T. Palpanas, S. Bergamaschi, Schema-agnostic progressive entity resolution, in: Proceedings of the 34th IEEE International Conference on Data Engineering, IEEE Computer Society, 2018, pp. 53–64.
[23] T. De Vries, H. Ke, S. Chawla, P. Christen, Robust record linkage blocking using suffix arrays, in: Proceedings of the 18th ACM Conference on Information and Knowledge Management, ACM, 2009, pp. 305–314.
[24] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava, et al., Approximate String Joins in a Database (Almost) for Free, in: Proceedings of the 27th International Conference on Very Large Data Bases, 1, 2001, pp. 491–500.
[25] M.A. Hernández, S.J. Stolfo, Real-World data is dirty: data cleansing and the merge/purge problem, Data Min. Knowl. Discov. 2 (1) (1998) 9–37.
[26] A. McCallum, K. Nigam, L.H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2000, pp. 169–178.
[27] S.E. Whang, D. Menestrina, G. Koutrika, M. Theobald, H. Garcia-Molina, Entity resolution with iterative blocking, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 2009, pp. 219–232.
[28] H.-S. Kim, D. Lee, HARRA: fast iterative hashed record linkage for large-scale data collections, in: Proceedings of the 13th International Conference on Extending Database Technology, ACM, 2010, pp. 525–536.
[29] L. Kolb, A. Thor, E. Rahm, Multi-pass sorted neighborhood blocking with mapreduce, Comput. Sci. Res. Devel. 27 (1) (2012) 45–63.
[30] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 10, 2010, p. 17.
[31] L. Huang, J. May, X. Pan, H. Ji, Building a fine-grained entity typing system overnight for a new X (X= Language, Domain, Genre), 2016 CoRR 1603.03112.
[32] Y. Ma, E. Cambria, S. Gao, Label embedding for zero-shot fine-grained named entity typing, in: Proceedings of the 26th International Conference on Computational Linguistics, ACL, 2016, pp. 171–180.
[33] B. Zhou, D. Khashabi, C. Tsai, D. Roth, Zero-shot open entity typing as type-compatible grounding, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, ACL, 2018, pp. 2065–2076.