Semantic Web 0 (2016) 1–0
IOS Press

Learning Expressive Linkage Rules from Sparse Data

Petar Petrovski, Christian Bizer
Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim
E-mail: ppetrovski@gmail.com, chris@informatik.uni-mannheim.de

Abstract.
Identifying entities in different data sources that describe the same real-world object is a central challenge in the context of the Web of Data as well as in data integration in general. Due to the importance of the problem, there exists a large body of research on entity resolution in the Linked Data as well as in the database community. Interestingly, most of the existing research focuses on entity resolution on dense data, meaning data that does not contain too many missing values. This paper sets a different focus and explores learning expressive linkage rules from as well as applying these rules to sparse data, i.e. data exhibiting a large amount of missing values. Such data is a common challenge in the context of the Web as different websites describe entities at different levels of detail. We propose and compare three entity resolution methods that employ genetic programming to learn expressive linkage rules from sparse data. First, we introduce the *GenLinkGL* algorithm which learns groups of linkage rules and applies specific rules out of these groups depending on which values are missing from a pair of records. Next, we propose *GenLinkSA*, which employs selective aggregation operators within rules. These operators exclude misleading similarity scores (which result from missing values) from the aggregations, but on the other hand also penalize the uncertainty that results from missing values. Finally, we introduce *GenLinkComb*, a method which combines the central ideas of the previous two into one integrated method. We evaluate all methods using six benchmark datasets: three of them are e-commerce product datasets, the other datasets describe restaurants, movies, and drugs. We show improvements of up to 16% F-measure compared to handwritten rules, on average 12% F-measure improvement compared to the original GenLink algorithm, 15% compared to EAGLE, and 8% compared to FEBRL.

Keywords:, Link Discovery, Entity Resolution, Sparse Data, Linkage Rules, Genetic Programming

1. Introduction

As companies move to integrate data from even larger numbers of internal and external data sources and as more and more structured data is becoming available on the public Web, the problem of finding records in different data sources that describe the same real-world object is moving into the focus within even more application scenarios.

Due to the relevancy of the problem, there exists an extensive body of research on entity resolution in the Linked Data [27] as well as the databases community [6, 14]. However, most existing approaches focus on dense data [8, 17, 28, 29]. This paper sets an alternative focus and explores learning expressive linkage (matching) rules from as well as applying these rules to sparse data, i.e. data that contains a large amount of missing values.

An example of an application domain that involves data exhibiting lots of missing values is e-commerce. Matching product data from different e-shops is difficult as the shops publish heterogeneous product descriptions using proprietary schemata which vary widely concerning their level of detail [26]. In [34], we analyzed product data from 32 popular e-shops. The shops use within each product category (mobile phones, headphones, TVs) approximately 30 different attributes to describe items. The subset of the attributes that is used depends on the e-shop and even on the specific product. This leaves a data aggregator that collects product data for many e-shops into a rich schema with lots of missing values.

In [17], we presented GenLink, a supervised learning algorithm that employs genetic programming to learn expressive linkage rules from a set of existing reference links. These rules consist of attribute-specific preprocessing operations, attribute-specific comparisons, linear and non-linear aggregations, as well as different weights and thresholds. The evaluation of GenLink on various benchmark datasets showed that the algorithm constantly performed in the group of top methods with F-measures above 95% [17]. As shown in the evaluation section of this paper, GenLink as well as other entity resolution methods run into problems once the data sets to be matched are not dense, but contain larger amounts of missing values.

In order to overcome the challenge of missing values, this paper introduces and evaluates three methods that build on the GenLink algorithm. First, we present *GenLink Group Learning (GenLinkGL)*, an approach that groups linkage rules based on product attribute diversity, thus successfully pivoting around missing values. Next, we introduce the *GenLink Selective Aggregations (GenLinkSA)* algorithm which extends the original approach with selective aggregation operators that ignore and penalize comparisons that include missing values. Finally, we introduce *GenLinkComb*, an algorithm that combines the central ideas of the previous two into a integrated method. We evaluate all methods using six benchmark datasets: three of them are e-commerce product datasets, the other datasets describe restaurants, movies, and drugs.

The rest of this paper is structured as follows: Section 2 formally introduce the problem of entity resolution. Section 3 gives an overview of the GenLink algorithm. Section 4 introduces the GenLinkGL, GenLinkSA, and GenLinkComb entity resolution methods for dealing with sparse data. Section 5 presents the results of the experimental evaluation in which we compare the methods to existing approaches. Section 6 discusses the related work.

2. Problem Statement

We consider two datasets, A the source, and B the target dataset. Each entity $e_a \in A$ and $e_b \in B$ consists of a set of attribute-value pairs (properties) $e_{a,b} = \{(p_1, v_1), (p_1, v_2), \ldots, (p_n, v_n)\}$, where the attributes are numeric, categorical or free-text. For instance, an entity representing a product might be described by the name, UPC, color, camera properties as shown in Figure 1. Our goal, is to learn a matching rule that determines whether a pair of entities (e_a, e_b) represents the same real world object. Or formally, given the two datasets A and B, the objective is to find the subset M consisting of all pairs of entities for which the equality relation holds and is defined by [15]:

$$M = \{(e_a, e_b); a = b, e_a \in A, e_b \in B\} \quad (1)$$

Additionally, find its complement subset U defined as:

$$U = \{(e_a, e_b); a \neq b, e_a \in A, e_b \in B\} \quad (2)$$

To infer a rule which specifies the conditions which must hold true for a pair of entities to be part of M, we rely on a set of positive correspondences $R_+ \subseteq M$ that contains pairs of entities for which the equality relation is known to hold. Analogously, we rely on negative correspondences $R_- \subseteq U$ that contains pairs of entities for which the inequality relation is known to hold.

Given the correspondences, we can define the purpose of the learning algorithm as learning matching rules from a set of correspondences:

$$m : 2^{(A \times B)} * 2^{(A \times B)} \rightarrow (A \times B \rightarrow [0, 1]) \quad (3)$$

The first argument denotes a set of positive reference links, while the second argument denotes a set of negative reference links. The result of the learning algorithm is a linkage rule which should cover as many reference links as possible while generalising to unknown pairs.

3. Preliminaries

GenLink is a supervised algorithm for learning expressive linkage rules for a given entity matching task. As all three algorithms that are introduced in this paper build on GenLink, this section summaries the main components of the GenLink algorithm. The full details of the algorithm are presented in [17].

3.1. Linkage Rule Format

GenLink represents linkage rules as a tree built out of four basic types of operators: (i) property operators,

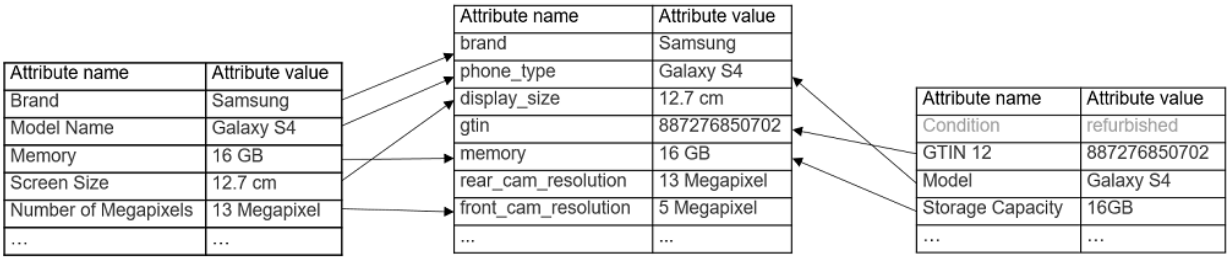


Fig. 1. Examples of product specifications sharing only a subset of the employed attributes: (left) Specification from walmart.com, (center) Central product Catalog and (right) Specification from ebay.com

(ii) transformation operators, (iii) comparison operators and (iv) aggregation operators. The linkage rule tree is strongly typed i.e. only specific combinations of the four basic operators are allowed. Figure 2 shows two examples of linkage rules for matching data describing mobile phones.

Property operators. Retrieves all values of a specific property p of each entity. For instance, in Figure 2a the left most leaf in the tree retrieves the value for the *"phone_type"* property from the source dataset.

Transformation operators. Transforms the values of a set of property or transformation operators. Examples of common transformation functions include case normalization, tokenization, and concatenation of values from multiple operators.

Comparison operators. GenLink offers three types of comparison operators: The first type of operators are character-based comparisons: equality, Levenshtein distance, and Jaro-Winkler distance. The second type includes token-based comparators: Jaccard similarity and soft Jaccard similarity. The comparison is done over a single property or a specific combination of properties. The third type of comparison operator calculated the similarity of two numbers. Examples of comparison operators can be seen in Figure 2a as the parents of the leaf nodes.

Aggregation operators. Aggregation operators combine the similarity scores from multiple comparison operators into a single similarity value. GenLink implements three aggregation operators: The maximum aggregation operator aggregates similarity scores by choosing the maximum score. The minimum aggregation operator chooses the minimum from the similarity score. Finally, the average aggregation operator combines similarity scores by calculating their weighted average.

Note that, these aggregation functions can be nested, meaning that non-linear hierarchies can be learned. For instance, in Figure 2a, four different properties are being compared (*"phone_type"*, *"brand"*, *"memory"* and *"display_size"*). Next, two average aggregations are applied to aggregate scores from phone_type and brand, and memory and display_size, respectively. Finally, a third average aggregation is applied to aggregate scores from the previous aggregators.

Compared to other linkage rule formats, GenLink's rule format is rather expressive and allows rules to closely adjust to the requirements of a specific matching situation by choosing a subset of the properties of the records for the comparison, choosing property-specific similarity functions, property-specific similarity thresholds, assigning different weights to different properties, and combining similarity scores using hierarchies of potentially non-linear aggregation functions.

3.2. The GenLink Algorithm

The GenLink algorithm starts with an initial population of candidate solutions which is iteratively evolved by applying a set of genetic operators.

Generating initial population. The algorithm finds a list of property pairs which hold similar values before the population is generated. Based on that, random linkage rules are built by selecting property pairs from the list and building a tree by combining random comparisons and aggregations.

Selection. The population of linkage rules is bred and the quality of the linkage rules is assessed by a fitness function relying on user-provided training data. The purpose of the fitness function is to assign a value to each linkage rule which indicates how close the given linkage rule is to the desired solution. The algorithm uses Matthews correlation coefficient (MCC) as fitness measure. MCC [23] is defined as the degree of the correlation between the actual and predicted classes or formally:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}} \tag{4}$$

The training data consists of a set of positive matches (linking entities identifying the same real world object) and a set of negative matches (linking entities identifying different objects). The prediction of the linkage rule is compared with the positive correspondences, counting true positives and false negatives and the negative correspondences, counting false positives and true negatives. In order to prevent linkage rules from growing indefinitely and potentially overfitting to the training data, we penalize linkage rules based on their number of operators:

$$fitness = MCC - 0.05 * operatorcount \quad (5)$$

Once the fitness is calculated for the entire population, GenLink selects individuals for reproduction by employing the tournament selection method. Tournament selection involves running several "tournaments" among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover. The method allows selection pressure to be easily adjusted by changing the tournament size.

Crossover. There are six types of crossover operators in GenLink: (i) Function crossover, (ii) Operators crossover, (iii) Aggregator crossover, (iv) Transformation crossover, (v) Threshold crossover and (vi) Weight crossover. The function crossover selects one operator at random in a pair of linkage rules and exchanges the functions between the operators. The operators crossover is designed to combine aggregations from two linkage rules. For this, it selects two aggregations, one from each linkage rule and combines theirs comparisons. The comparisons are combined by selecting all comparisons from both aggregations and removing each comparison with a probability of 50%. In order to learn aggregation hierarchies, the aggregation crossover operator selects a random aggregation or comparison operator in the first linkage rule and replaces it with a random aggregation or comparison operator from the second linkage rule. The last three types of crossovers are used to recombine transformation functions, thresholds and weights for two linkage rules. An in-depth discussion of the crossover operators is provided in [17].

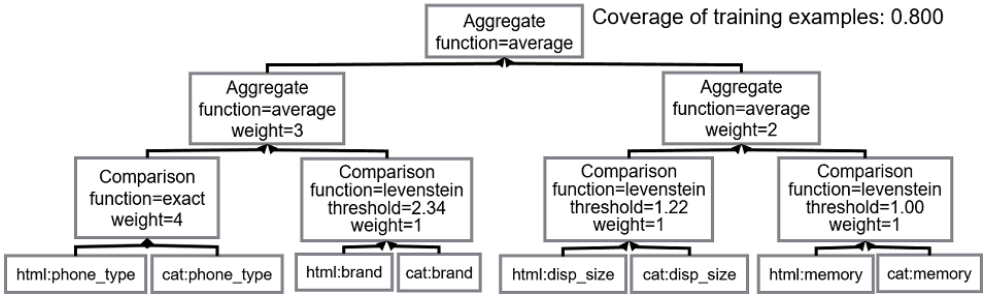
4. Approaches

In [33] we have shown that the GenLink algorithm struggles to optimise property selection for datasets that contain a lot of missing values: On an e-commerce data set containing many low-density attributes the algorithm only reached an F-measure of less than 80%, in contrast to the above 95% results that are often reached on dense datasets. In the following we propose three algorithms that build on the GenLink algorithm and enable it to properly exploit sparse attributes. The GenLinkGL algorithm builds a group of matching rules for the given matching task (*group generation*) and applies the group of matching rules to create new correspondences (*group application*). Next we introduce *selective aggregations*, new operators within the GenLink algorithm that can better deal with missing values. Finally, we introduce *GenLinkComb*, that integrates the central ideas of the previous two methods into a single combined method.

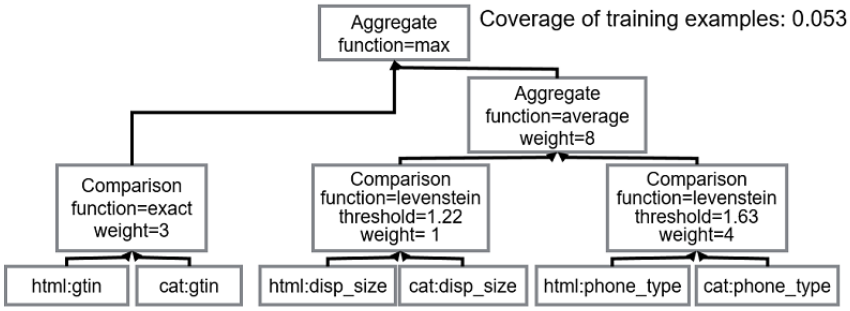
4.1. The GenLinkGL Algorithm

The GenLink algorithm lacks the capability to optimise property selection when dealing with sparse data. The algorithm will select a combination of dense properties while sparse properties will rarely be selected. This behavior has negative consequences for the cases in which values from relatively dense properties are missing. For instance, when matching product data describing mobile phones from different e-shops, the brand, phone type, and memory properties will be rather important for the matching decisions and these attributes will also likely be rather dense as they are provided by many e-shops. Thus, GenLink will focus on these attributes and due to the penalty on large rules (compare Equation 5) will not include alternative attribute combinations involving low density properties, such as gtin number[1], display size, or operating system. In cases in which a value of one of these important attributes is missing, the algorithm will likely fail to discover the correct match, while it could still have been possible to discover the correct match by exploiting a combination of alternative attributes which might be filled in this concrete case. Including all alternative attribute combinations into a single linkage rule would result in rather large rules containing multiple alternative branches that encode the different attribute combinations. Due to the penalty for large rules from Equation 5, only the most important alternative attribute combinations will be included into the rules

[1] Global Trade Item Number (GTIN) is an identifier for trade items, developed by GS1. – www.gtin.info/



(a)



(b)

Fig. 2. Example of two rules from the group for the phone category together with the coverage of each rule

and combinations having a lower coverage will be left unused.

A way to deal with this problem could be to loosen the size penalty in Equation 5. With GenLink Group Learning (GenLinkGL), we choose an alternative approach: Instead of trying to grow very large rules that cover different attribute combinations, we try to learn sets of rules in which each rule is optimized for a specific property combination. This way, we more clearly separate the issue of avoiding overfitting rules while still be able to cover multiple property combinations. By combining multiple combinations of properties in a group, the learning algorithm is given the freedom to optimize matching rules not only for the most common attribute combinations but also for less common combinations involving sparse properties, thus increasing the overall recall. In the following, we describe how GenLinkGL combines rules into groups and later selects a rule from the group in order to match a pair of records having a specific property combination.

Group generation. The basic idea of the first algorithm, presented in Algorithm 1, is that by grouping different linkage rules with different properties we could circumvent the missing values in the data. The initial group is populated with the top fitness individual from the population generated by GenLink. Subsequently, an initial fitness for this group is computed using the MCC (compare Equation 4).

Algorithm 1 Generating a group

Input:
$Group \leftarrow rule$ top fitness matching rule
$P \leftarrow Rules$ All matching rules in the available population
Output:
The top fitness group
for all $i \in P$ **do**
  if $i.properties \not\subset Group.properties$ **then**
    $PotentialGroup \leftarrow insert(Group, i)$
    if $fitness(PotentialGroup) > fitness(Group)$ **then**
      $Group = PotentialGroup$
    end if
  end if
end for
return G

Motivated by the GenLink algorithm, our algorithm builds a group that maximises fitness. To do that at each learning iteration, the algorithm iterates through the entire population of linkage rules and combines their individual fitness. We restrict the combination to linkage rules whose properties are not a subset of the

properties of the group and include a linkage rule that has at least one new property that is not present in the group. We combine the fitness of the linkage rules by summing the number of correctly predicted instances in the training set, calculating for each individual the percentage of the coverage of training examples in the group. Once the correctly predicted instances are summed the current fitness function is applied to the group. If the fitness of that combination is greater than the current top fitness group, the new group becomes the best group. As an output the algorithm gives the top fitness group.

Algorithm 1 can potentially lead to groups containing a large number of rules, up to the complete population of learned rules. In such case the algorithm is prone to overfitting, since the population might capture the entire training set. In order to prevent this, we penalize groups containing a large number of rules: $fitness = MCC - c * rulecount$. Where, $c = (0.001, 0.003, 0.005)$ is a small constant, which strictly depends on the number of individuals in the population. Namely, the larger the population, the bigger the chance for overfitting. Therefore, the constant should be higher for larger populations in order to penalise the fitness more. By penalizing the fitness by the number of members in the group we ensure that there will be no unneeded bloating of the learned group.

For example, let the linkage rule in Figure 2a be the top fitness individual after the $n-th$ learning iteration of the algorithm. The initial group contains this linkage rule. The group would not be able to correctly predict correspondences that could only have been matched by a combination of the gtin, phone_type and memory properties. At the first iteration we combine the group with the linkage rule in Figure 2b containing the gtin property. As a result, the correspondences above could be captured by the group leading to better fitness.

Group application. As an input the second algorithm, presented in Algorithm 2, takes the output of Algorithm 2 and a set of pairs to be matched. The individuals in the input group are sorted by the percentage of coverage. Sorting enables the Algorithm 2 to find the more influential individual rules in less iterations. For each pair the algorithm iterates through the group of matching rules. If the pair to be matched contains the same properties as in the matching rule, the matching rule is picked. If there is no matching rule which has the exact properties as the instances, the top matching rule is picked. For instance, when matching *(a) the specification from walmart.com with the product catalog* and *(b) the specification from ebay.com with the product catalog* from Figure 1, the algorithm would use the first rule from Figure 2 for the *a* pair, but use the second matching rule from Figure 2 for the *b* pair since in *b* one of the specifications does not have a value for the display_size attribute, however it contains a gtin attribute.

Algorithm 2 Applying a group to set of pairs for matching

```
Input:
G ← group of matching rules
Pairs ← pairs for matching
Output:
Linked instances
Result ← nil
for all pair ∈ Pairs do
  for all rule ∈ G do
    if pair.properties ≡ rule.properties then
      Result ← match(pair, rule)
      break
    end if
  end for
  if ∄match then
    Result ← match(pair, G.top)
  end if
end for
return Result
```

Property diversity is an underlying factor behind this method. Since the prime goal is to enlarge the combination of properties that are used for matching, it is imperative that the dataset contains a diverse range of properties. More precisely, if the dataset has a smaller number of properties, the number of combination of properties that can be made by grouping linkage rules is smaller. Therefore, this approach would not improve much upon the GenLink when dealing with datasets with smaller number of properties.

4.2. The GenLinkSA Algorithm

An alternative to learning groups of small rules specializing on a specific property combination each is to learn larger rules covering more properties and apply a penalty for the uncertainty that arises from values missing in these properties. For instance, a larger rule could rely on five properties for deciding whether two records match. If two of the five properties have missing values, the remaining three properties can still be used for the matching decision. Nevertheless, a decision based on three properties should be considered

less certain than a decision based on five properties. In order to compensate for this uncertainty, we could require the values of the remaining three properties to be more similar than the values of the five properties in the original case in order to decide for a match. The GenLink Selective Aggregations (GenLinkSA) algorithm implements this idea by changing the behavior of the comparison operators as well as the aggregation operators in the original GenLink algorithm.

Null-enabled Comparison Operators. The original GenLink algorithm does not distinguish between a pair of different values and a pair of values containing a missing value. In both cases, the algorithm assigns the similarity score 0. This is problematic when similarity scores from multiple comparison operators are combined using the aggregation function average or minimum, as the resulting similarity score will be unnaturally low for the case of missing values. In order to deal with this problem, GenLinkSA amends the comparison operators with the possibility to return the value *null*: A GenLinkSA comparison operator will return *null* if one or both values are missing. If both values are filled, the operator will apply its normal similarity function and return a value in the range $[0, .., 1]$.

Selective Aggregation Operators. The GenLink aggregation operators calculate a single similarity score from the similarity values of multiple comparison operators using a specific aggregation function such as weighted average, minimum, or maximum. GenLinkSA adjusts the aggregation operators to apply the aggregation function only to non-null values. In order to compensate the uncertainty that results from missing values (comparison operators returning the value *null*), the similarity score that results from the aggregation in reduced by constant factor α for each comparison operators that returns a *null* value. In this way, all non-null similarity scores are aggregated and a penalty is applied for each property pair containing missing values. Formally, a GenLink aggregation is defined by the following equation:

$$\begin{aligned} S^a : (S^* \times N^* \times F^a) &\rightarrow S \\ (\bar{s}, \bar{w}, f^a) &\rightarrow ((e_a, e_b) \rightarrow f^a(s_e, w)) \\ \textit{with} \quad s_e : (s_1(e_a, e_b), s_2(e_a, e_b), .., s_n(e_a, e_b)) & \end{aligned} \tag{6}$$

Given the aggregation operators, we can now define GenLinkSA's *selective aggregation operators* as:

$$\begin{aligned} S^a : (S^* \times N^* \times F^a) &\rightarrow S \\ (\bar{s}, \bar{w}, f^a) &\rightarrow ((e_a, e_b) \rightarrow f^a(s_e, w)) - \upsilon \\ \textit{with} \quad s_e : (s_1(e_a, e_b), s_2(e_a, e_b), .., s_n(e_a, e_b)), & \\ \upsilon = \beta * \mid \{\, s_i(e_a, e_b) \mid s_i(e_a, e_b) \rightarrow \textit{null} \wedge s_i \in s_e \,\} \mid & \end{aligned} \tag{7}$$

Where the uncertainty factor υ is defined as the number of *null* values multiplied by a small valued constant factor $\beta = (0.01, 0.03, 0.05)$. The uncertainty factor serves to penalize the rule for each null similarity operator. As the overall similarity score is reduced by the uncertainty factor, the values of the non-null properties must be more similar in order to reach the same similarity score as for a pair in which all properties are filled.

For example, let the rule that was learned by the GenLinkSA algorithm be the one shown in Figure 3 and let instances for matching be *(a) the specification from walmart.com that should be matched with the product catalog* and *(b) the specification from ebay.com to be matched with the product catalog* from Figure 1. When matching (a) only a small penalty will be applied since for five out of six comparisons a non-null similarity score will be returned and only the comparison for one property (*comp_os*) will be penalised. On the other hand, the pair (b) will be heavily penalized since four of the six comparisons will return null values. Evidently, this method will discourage high similarity scores in the presence of missing values and will thus refrain from considering borderline cases with missing values as matches, resulting in a higher precision.

4.3. The GenLinkComb Algorithm

Both *GenLinkGL* and *GenLinkSA* tackle the issue of missing values differently. Namely, GenLinkGL strives to group matching rules exploiting different combinations of properties and thus be able to apply alternative rules given that values of important properties are missing. By being able to exploit alternative property combinations, GenLinkGL is tailored to improving recall. On the other hand, by penalizing comparisons with missing values, GenLinkSA incentives learning matching rules that include more properties

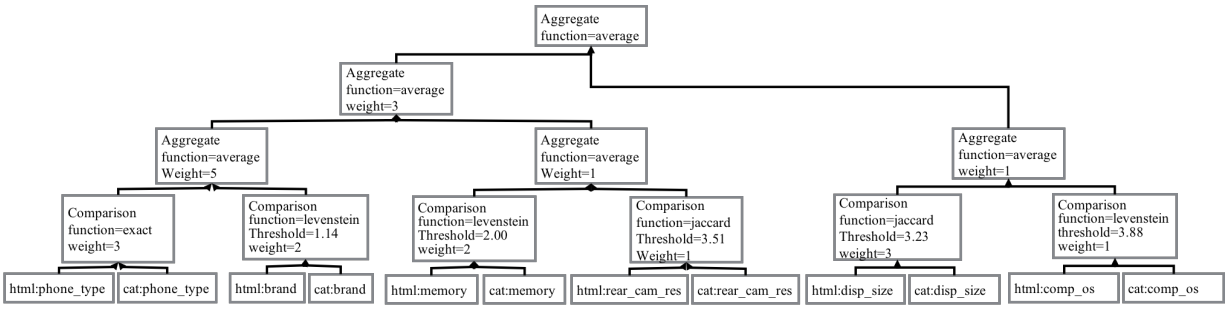


Fig. 3. GenLink SA learned rule for the Phone category

and substantially lowers the similarity scores of uncertain pairs, and by that improves precision. As the basic ideas behind GenLinkGL and GenLinkSA do not exclude each other but are complementary, a combination of both methods into a single integrated method could combine the advantages of both methods: Optimize rules for alternative attribute combinations while at the same time dealing with the uncertainty that arises from missing values inside the rules. The *GenLinkComb* algorithm achives this by combining the GenLinkSA and the GenLinkGL algorithms as follows: GenLinkComb uses the GenLinkSA algorithm to evolve the population of linkage rules. In each iteration of the learning process, GenLinkComb groups the learned rules together using the GenLinkGL algorithm. By being able to deal with missing values either inside the rules using the selective aggregation operators or within the grouping of rules, the GenLinkComb learning algorithm has a higher degree of freedom in searching for a good solution.

5. Evaluation

Evaluation of the aforementioned methods was conducted using six benchmark datasets: three e-commerce product datasets, and thee other datasets describing restaurants, movies, and drugs. Beside of comparing GenLinkGL, GenLinkSA, and GenLinkComp with each other, we also compare the approaches to existing systems including FEBRL, EAGLE, COSY, MARLIN, ObjectCoref, and RiMOM. The following section will describe the six benchmark datasets, give details about the experimental setup, and present and discuss the results of the matching experiments.

5.1. Datasets

Product Matching Datasets. We use three different product datasets for the evaluation:

Abt-Buy dataset: The dataset includes correspondences between 1081 products from Abt.com and 1092 from Buy.com. The full input mapping contains 1.2 million correspondences, from which 1000 are annotated as positive correspondences (matches). Each entity of the dataset might contain up to four properties: product name, description, manufacturer and price. The dataset was introduced in [19]. Since the content of the product name property is a short text listing various product features rather than the actual name of the product, we extract the product properties shown in Table 1 from the product name values using the dictionary-based method presented in [35]. We choose the Abt-Buy dataset because it is widely used to evaluate different matching systems[5, 9].

Amazon-Google dataset: The dataset includes correspondences between 1363 products from Amazon and 103,226 from Google. The full input mapping contains 4.4 million correspondences, from which 1000 are annotated as matches. Each entity of the dataset contains the same properties as the Abt-Buy dataset. This dataset is presented in [19]. We perform the same extraction of properties as in the Abt-Buy dataset. The Amazon-Google data set has also been widely used as benchmark data set [19].

WDC Product Matching Gold Standard: This gold standard [34] for product matching contains correspondences between 1500 products (500 each from the categories headphones, mobile phones, and TVs), collected from 32 different websites that provide schema.org annotations and a unified product catalog containing 150 products with

Table 1

Properties together with their density in the Abt-Buy and Amazon-Google datasets.

Dataset	Property	Density (A / B) %
Abt-Buy	Original Attributes	
	Product Name	100
	Description	63
	Manufacturer	48
	Price	36
	Extracted Attributes	
	Model	91
	Brand	72
Amazon-Google	Original Attributes	
	Product Name	100
	Description	70
	Manufacturer	52
	Price	31
	Extracted Attributes	
	Model	88
	Brand	76

the following distribution: (1) Headphones-50, (2) Phones-50, and (3) TVs-50. The data in the catalog has been scraped from leading shopping services, like Google Shopping, or directly from the vendor's website. The gold standard contains 500 positive correspondences (matches) and more than 25000 negative correspondences (non-matches) per category. Compared to the Amazon-Google and Abt-Buy datasets, the WDC Product Matching Gold Standard is more heterogeneous as the data has been collected from different websites. The gold standard also uses a richer schema containing about 10 properties per product category.

Other Entity Resolution Datasets. In order to be able to compare our approaches to more reference systems, as well as to showcase the ability of our algorithms to perform on datasets from other domains than products we use three more benchmark datasets used in [17]:

Restaurant dataset: The dataset contains correspondences between 864 restaurant entities from the Fodor's and Zagat's restaurant guides. Specifically, there have been identified 112 duplicate records.

Sider-Drugbank dataset: The dataset contains correspondences between 924 drug entities in the Linked Data version of the Sider dataset and 4772 drug entities in the Linked Data version of the Drugbank dataset. Specifically, there have been identified 859 duplicate records.

Table 2

Properties and property density of the WDC Product Matching Gold Standard, Restaurants, Sider-Drugbank and LinkedMDB datasets.

Dataset	Property	Density (A / B) %
WDCPr Gold Standard	Headphones	
	Brand	97 / 100
	Product Name	87 / 100
	MPN	60 / 86
	Color	56 / 96
	Sensitivity	53 / 88
	Impedance	53 / 92
	Cup Type	47 / 38
	Form Factor	43 / 77
	Magnet Mat.	27 / 51
	Diaphragm	25 / 35
	Phones	
	Product Name	91 / 100
	Memory	87 / 95
	Brand	86 / 100
	Color	79 / 43
	Display Size	71 / 92
	Rear Cam. Res.	70 / 85
	OS	64 / 64
	Display Res.	48 / 53
	Processor	28 / 36
	Front Cam. Res.	20 / 66
	TVs	
	Brand	100 / 100
	Product Name	91 / 100
	Display Type	81 / 85
	Display Size	65 / 96
	Display Res	55 / 87
	Tot. Size	51 / 74
	Ref. Rate	50 / 96
	Img. Asp. Rat.	38 / 60
	Connectivity	35 / 61
	Resp. Time	10 / 25
Restaurant	Name	100
	Address	100
	Contact	100
	Type	100
Sider-Drugbank	Name	100 / 100
	Indication	100 / 93
LinkedMDB	Name	100 / 100
	Director	100 / 100
	Rel Date	100 / 100
	Studio	95 / 97

LinkedMDB dataset This dataset contains 100 correspondences between 373 movies. The authors note that special care was taken to include relevant corner cases such as movies which share the same title but have been produced in different years.

Tables 1 and 2 give an overview of densities of properties in the six evaluation datasets. If the density of a property differs in the source (A) and the target (B) dataset, both densities are reported. For the

Abt-Buy and Amazon-Google datasets, we show all original property densities as well as the density of the extracted properties. As stated before, the product datasets exhibit more sparsity. The Abt-Buy and Amazon-Google datasets follow a similar distribution in which only the product name property has a density of 100%. It is worth to be noted that the product name property in these datasets is actually a short description of the product mentioning different properties rather than the actual product name. WDC Product Matching Gold Standard contains a small set of properties with a density above 90% while most properties belong to the long tail of rather sparse properties [34].

5.2. Experiment Setup

The GenLinkGL, GenlinkSA, GenLinkComb algorithms were implemented on top of the Silk Framework[2]. The source code of the original GenLink implementation[3] as well as the source code of GenLinkGL, GenlinkSA, GenLinkComb algorithms[4] is publicly available, so all results presented in the paper can be replicated. Table 3 gives an overview of the aggregation, comparison, and transformation functions the algorithms could choose from within the experiments. To be noted that for each aggregation operator there exists also a selective aggregation operator. Table 4 summarises the parameters that were used for GenLink and its variants in the experiments. All experiments are run 10 times and the results are averaged.

In order to set the GenLink results into context, we also ran the WDC Gold Standard experiments with EAGLE [31], a supervised matching system that also employs genetic programming[5] and FEBRL [9][6], an entity resolution system that internally employs an SVM. GenLink and its variants as well as EAGLE were trained on a balanced data set consisting of 66% matching correspondences and the same number non-matching correspondences. The systems were evaluated afterwards using the remaining 33% of the correspondences. For training FEBRL, we calculated TF-IDF scores and cosine similarity for all pairs given in the dataset. As with GenLink and EAGLE, FEBRL was trained on 66% of the data and evaluated on the rest. For the experiments on the Abt-Buy and Amazon-Google datasets, all systems were trained using the original as well as the extracted attribute-value pairs.

Table 3

Available aggregation comparison and transformation functions. The transformation functions are used only for non-product datasets

Comparison	Aggregation	Transformations
Exact Similarity	Average	Tokenize
Levenstein Distance	Maximum	Lower Case
Jaccard Similarity	Minimum	Concatenate
Number Similarity		

Table 4

GenLink (GL/SA/Comb) Parameters

Parameter	Value
Population size	1000
Maximum iterations	100
Selection method	Tournament selection
Tournament size	10
Probability of Crossover	50%
Probability of Mutation	50%
Stop Condition	F-measure = 1.0
Matching Rule Penalty	0.03
Uncertainty constant	0.05

The restaurants, movies, and drugs datasets have a original density of over 90%. In order to use them to evaluate how the different approaches perform on sparse data, we systematically removed 25%, 50% and 75% of the values. More precisely, we first randomly sample 50% of properties (not including the *name* property) and for those we randomly select 25%, 50% and 75% of the values and removed the rest, thus introducing greater percentage of null values in the datasets. We do not remove values from all properties since we want to recreate the sparseness as in the product datasets as close as possible.

5.3. Product Matching Results

Table 5 gives an overview of the matching results on the WDC Product Matching Gold Standard dataset. As baselines, we repeat TF-IDF cosine similarity and Paragrph2Vec experiments presented in [34]. The first baseline, considers pair-wise matching of product descriptions for which TF-IDF vectors are calculated using the bag-of-word feature extraction method. The second baseline, considers building a Paragraph2Vec model [21] for product names using 50 latent features and the Distributed Bag-of-Words model. Moreover, we compare results from: (i) handwritten matching rules, (ii) the GenLink algorithm, (iii) GenLinkGL, (iv) GenLinkSA and (v) GenLinkComb.

[2] www.silkframework.org

[3] https://github.com/silk-framework/silk. To be noted that the 2.6.0 version was used for the experiments.

[4] https://github.com/petrovskip/silk.2.6-GenLinkSA and https://github.com/petrovskip/silk.2.6-GenLinkGL

[5] http://aksw.org/Projects/LIMES.html

[6] https://sourceforge.net/projects/febrl/

Table 5
Matching results per category for the WDC Product Matching Gold Standard

Headphones			
	Precision	Recall	F-measure
Baseline TF-IDF Cosine	0.622	0.559	0.588
Baseline Pargraph2vec	0.667	0.685	0.675
Handwritten Rule	0.841	0.838	0.839
EAGLE [31]	0.661	0.905	0.763
GenLink [17]	0.692	0.946	0.799
FEBRL [9]	0.884	0.837	0.850
GenLinkGL	0.837	0.924	0.888
GenLinkSA	0.922	0.925	0.923
GenLinkComb	0.920	0.961	**0.940**
Phones			
	Precision	Recall	F-measure
Baseline TF-IDF Cosine	0.385	0.676	0.491
Baseline Pargraph2vec	0.497	0.624	0.553
Handwritten Rule	0.656	0.722	0.687
EAGLE [31]	0.699	0.672	0.685
GenLink [17]	0.708	0.715	0.712
FEBRL [9]	0.792	0.748	0.776
GenLinkGL	0.742	0.894	0.808
GenLinkSA	0.813	0.737	0.773
GenLinkComb	0.815	0.886	**0.849**
TVs			
	Precision	Recall	F-measure
Baseline TF-IDF Cosine	0.661	0.474	0.554
Baseline Pargraph2vec	0.654	0.553	0.572
Handwritten Rule	0.782	0.716	0.747
EAGLE [31]	0.722	0.674	0.697
GenLink [17]	0.790	0.711	0.748
FEBRL [9]	0.807	0.747	0.775
GenLinkGL	0.791	0.875	0.819
GenLinkSA	0.864	0.745	0.810
GenLinkComb	0.863	0.815	**0.838**

The handwritten rules are composed using six properties for each product category and are written by an expert in the field of matching and similarity functions. Additionally, we compare to two state-of-the-art matching systems for this dataset: (i) EAGLE [31] and (ii) FEBRL [19] as explained above.

As expected both baselines perform poorly for each product category. Specifically, TF-IDF could not capture enough details of a given entity. Paragaph2Vec, improves on the TF-IDF baseline by including the semantic relations between the words of a given record. However, the semantic relationships do not prove to be sufficient. EAGLE [31] and GenLink [17] improve on the baselines since they have the ability to optimise the thresholds for comparisons and the weights within aggregations. However, EAGLE [31] has significantly lower results on each category than GenLink [17]. Both methods have comparable results with the handwritten rules. The first method that shows a better performance than the handwritten rules for all product categories is FEBRL [9]. Because of FEBRL's SVM implementation is optimized for entity resolution, the system seems to be able to capture more nuanced relationships between data points than the handwritten rules. The main difficulty of the FEBRL is recall. In addition, the method has problems with matching corner cases.

All of the GenLinkGL, GenLinkSA, and GenLinkComb consistently outperform or show comparable results to FEBRL and the handwritten rules. For instance, when comparing FEBRL to the GenLinkGL algorithm, we can notice significantly worse recall results. The GenLinkGL algorithm decreases the number of false negatives by learning sets of rules in which each rule is optimized for a specific property combination. Hence, the algorithm is successfully pivoting around missing values, and in turn exhibits a jump in recall. Correspondingly, the GenLinkSA algorithm gives comparable results in F-measure compared to FEBRL, mostly due to the jump in precision. The precision jump happens since the selective aggregation operators substantially lower matching scores of uncertain pairings due to the uncertainty factor. Due to this penalty, pairs with missing values which otherwise would have borderline similarity will not be considered matches. Both the jump in recall of GenLinkGL and the jump in precision of GenLinkSA contribute to improve the matching and the algorithms have comparable results in F-measure. Finally, the GenLinkComb algorithm shows significantly better performance in F-measure than the rest of the tested field, due to the fact that the combination method is able of both preserving precision by penalising borderline cases with missing values and preserving recall by successfully exploiting alternative attribute combinations.

Category wise, the headphones category shows with 94% F-measure the best results by a significant margin (9%). Headphones have a smaller number of distinct properties and therefore e-shops tend to more consistently describe products with the same attributes compared to the other two categories. The TVs and phones category reach similar F-measures of 83.8% and 84.9% respectively.

Comparison of the learned matching rules. In order to explain the differences in the results of GenLinkSA, GenLinkGL, and GenLinkCombm, we analyze and compare the rules that were learned by the three algorithm for matching mobile phones. Figure 3 shows the GenLinkSA rule that was learned. As we can see, the rules uses six properties which are combined

using a hierarchy of average aggregations. Within the hierarchy, more weight is put onto branch containing four properties, as well as on the properties brand and phone_type within this branch. The GenLinkGL algorithm has learned a group consisting of 12 matching rules that use 15 distinct properties for matching phones. Table 6 shows the top five rules from the GenLinkGL approach sorted by their coverage. More than 50% of the rules contain the model (phone_type) and the display size (disp_size) attributes. It is interesting to examine the coverage of the learned rules: The first rule was applied to match 80% of the pairs in the training data. The second rule was only used for 5% of the cases, the next rule for 2% and so on, meaning that the data contained one dominant attribute combination (the one exploited by the first rule) while by specializing on alternative combinations (like the second rule involving the gtin property) still improved the overall result. Furthermore, most of the learned matching rules use similar combinations of aggregation functions (average aggregation). The only exception is the second rule which uses the property gtin. Namely, the gtin property by itself is enough to identify the specific product, thus the maximum aggregation function is used. For matching phones, the GenLinkComb algorithm has learned a group that only consists of five matching rules which use 10 distinct properties. Meaning that it achieves a better F1-performance using less rules and less properties as GenLinkGL. Table 7 shows the rules that were learnt by the GenLinkComb algorithm, again sorted by coverage. Interestingly, the rules have a more homogenous coverage distribution than the GenLinkGL rules. Instead of generating low-coverage rules for exotic property combinations as GenLinkGL does, GenLinkComb generate less groups which exploit more properties each and uses the selective aggregations and the uncertainty penalty to deal with missing values within these properties. The property composition also supports this argument: The robust property composition of GenLinkComb suggests that the learned matching rules in the group contain more nuanced differences, while GenLinkGL has more irregular property composition.

Table 6

Property, Comparisons, Aggregations and Training example coverage for the top 5 rules in the learned group for phone category learned by GenLinkGL

Properties	Comps.	1st Agg.	2nd Agg.	Coverage
phone_type	Exact	Avg	Avg	0.800
brand	Levens.			
dips_size	Levens.	Avg		
memory	Levens.			
gtin	Exact	Avg	Max	0.053
memory	Levens.			
phone_type	Levens.			
phone_type	Exact	Avg	Avg	0.020
brand	Levens.			
proc_type	Exact	Avg		
core_count	Exact			
phone_type	Exact	Avg	Avg	0.017
comp_os	Levens.			
rear_cam_res	Jaccard	Avg		
front_cam_res	Jaccard			
disp_size	Exact	Avg	Avg	0.013
brand	Exact			
rear_cam_res	Jaccard	Avg		
disp_res	Jaccard			

Amazon-Google and Abt-Buy Results. In order to also evaluate the algorithms on datasets having lower number of distinct properties (see Table 1), we applied the algorithms to the Amazon-Google and Abt-Buy datasets. The results of these experiments are given in Table 8 and Table 9). As reference systems, apart of FEBRL, the best performing approaches found in literature are listed. Table 8 gives results on the matching experiment done on the Amazon-Google dataset. Conversely, GenLinkComb outperforms a commercial system [19] based on manually set attribute-level similarity thresholds. The commercial system [19] derives matching rules similar to the handwritten rules in WDC Product Matching Gold Standard and therefore is inferior to the GenLinkComb. While GenLinkGL underperforms on this dataset, due to the low number of distinct properties, it still able to fit the dataset and achieves the best recall score.

Table 9 gives results on the matching experiment done on the Abt-Buy dataset. As with previous datasets GenLinkComb shows the best performance in terms of F-Measure. Both, FEBRL's SVM classifier [9] and MARLIN [5][7] give comparable results to both GenLinkSA and GenLinkGL. This is to be expected, as the features for both FEBRL and MRLIN were manually engineered for the given datasets whereas our methods select features automatically. Moreover, the SVM's for both FEBRL and MARLIN were trained with larger feature sets than our approaches (five matchers on two properties).

When comparing the results of the experiments with WDC Product Matching Gold Standard to the results of the Abt-Buy and Amazon-Google datasets it becomes evident that the GenLink variants perform better on datasets containing a large number of properties

[7] Results from experiments with FEBRL and MARLIN are published in [19]

Table 7

Property, Comparisons, and Training example coverage and Normalized threshold mean for the top 5 rules in the learned group for phone category learned by GenLinkComb

Properties	Comps.	1st Agg.	2nd Agg.	3rd Agg.	Coverage
phone_type brand memory dips_size memory phone_type	Levens. Levens. Jaccard Jaccard Exact. Levens.	Avg Avg Min 	Min	Avg	0.492
phone_type memory rear_cam_res memory dips_size	Exact Exact. Jaccard Levens. Levens.	Min Avg 	Avg	Min	0.221
phone_type brand memory rear_cam_res dips_size comp_os	Exact Levens. Levens.. Jaccard Jaccard Levens.	Avg Avg Avg 	Avg	Avg	0.215
phone_tupe memory phone_type proc_type	Exact Levens. Levens. Exact	Min 	Min	Avg	0.037
phone_type memory memory front_cam_res disp_res phone_type	Levens. Exact Levens. Jaccard Jaccard Jaccard	Min Min Avg 	Avg	Min	0.035

Table 8

Product matching results for the Amazon-Google dataset

	Precision	Recall	F-measure
GenLink [17]	0.493	0.571	0.513
GenLinkGL	0.501	0.813	0.604
GenLinkSA	0.691	0.632	0.643
GenLinkComb	0.690	0.651	**0.669**

Reference Systems	F-measure
FEBRL [9]	0.601
COSY [19]	0.622

than on dataset containing only a smaller number of properties.

5.4. Other Domains Results

Generally, for all datasets we can conclude that our methods find it difficult to find the correct matches when dealing with severely sparse data (25%). Additionally, GenLinkComb and GenLinkSA have similar performance and both tend to outperform GenLinkGL for every dataset for the sparser settings. Contrarily, when the datasets have 75% property density, our methods perform close to the results of reference systems achieved on the datasets with more than 90% property density.

Table 9

Product matching results for the Abt-Buy dataset

	Precision	Recall	F-measure
GenLink [17]	0.632	0.694	0.661
GenLinkGL	0.650	0.833	0.730
GenLinkSA	0.721	0.714	0.717
GenLinkComb	0.723	0.798	**0.758**

Reference Systems	F-measure
FEBRL [9]	0.713
MARLIN [5]	0.708

Table 10

Results for the Restaurants dataset

	Density		
	25%	**50%**	**75%**
	F-measure	F-measure	F-measure
GenLink [17]	0.651	0.654	0.909
GenLinkGL	0.642	0.661	0.905
GenLinkSA	0.654	0.660	0.938
GenLinkComb	0.653	0.664	0.936

Reference Systems on 100% density	F-measure
GenLink [17]	0.993
Carvalho et al.[7]	0.980

Table 10 gives results on the matching experiment done on the Restaurant dataset. GenLinkSA and GenLinkComb perform closest to the reference systems, while GenLinkGL does not show any improvement on this dataset. Due to low number of properties that this dataset has GenLinkComb and GenLinkGL show little improvement compared to the other methods. Consequently, GenLInkComb and GenLinkGL cannot find enough matching rules with alternative attributes to group, making GenLinkComb to boil down to GenLinkSA and GenLinkGL to boil down to GenLink. Density wise, all three methods follow the same downward trend when the dataset is more sparse, keeping the relative improvements of GenLinkSA and GenLinkGL in comparison to GenLink.

Table 11 gives results on the matching experiment done on the Sider-Drugbank dataset. Even though we systematically lowered the quality of the dataset, GenLink still outperforms the state-of-the-art [16, 37] systems for the case of 75% property density. With that said, GenLinkGL and GenLinkSA reach considerably better results in recall and precision respectively. When the data become severely sparse, like in the case of 25% our methods show an increase of 5% in F-measure compared to GenLink. Similarly to the Restaurant dataset the GenLinkComb does not improve over GenLinkSA as again the grouping algo-

Table 11
Results for the Sider-Drugbank dataset

	Density		
	25%	**50%**	**75%**
	F-measure	F-measure	F-measure
GenLink [17]	0.345	0.388	0.837
GenLinkGL	0.399	0.424	0.875
GenLinkSA	0.401	0.422	0.871
GenLinkComb	0.402	0.422	0.872

Reference Systems on 100% density	F-measure
ObjectCoref [16]	0.464
RiMOM[38]	0.504
GenLink [17]	0.970

Table 12
Results for the LinkMDB dataset

	Density		
	25%	**50%**	**75%**
	F-measure	F-measure	F-measure
GenLink [17]	0.540	0.587	0.873
GenLinkGL	0.550	0.627	0.911
GenLinkSA	0.559	0.624	0.920
GenLinkComb	0.611	0.658	0.952

Reference Systems on 100% density	F-measure
EAGLE [31]	0.941
GenLink [17]	0.999

rithm could not find any suitable rules with alternative attributes for grouping.

Table 12 gives results on the matching experiment done on the LinkMDB dataset, which contains more properties compared to the other two datasets. In this case GenLinkComb outperforms other variations of GenLink even when data spareness is severe. Unlike with the Restaurants and Sider-Drugbank datasets GenLinkComb successfully finds rules with alternative attributes to group and thus increasing F-measure by 5% compared to GenLinkSA.

6. Related Work

Entity resolution has been extensively studied under different names such as link discovery [27], record linkage [1, 8, 29], reference reconciliation [13], coreference resolution [22, 28]. In the following, we discuss different entity resolution approaches with respect to their ability to deal with missing values, while we refer the reader to the existing surveys [6, 10, 14, 27, 37] for a more comprehensive comparison of the approaches.

Distance-based entity resolution approaches focus on learning a pairwise distance metric between entities and then either set a distance threshold or build a pairwise classifier to determine which entities are merged. Such pairwise classifiers can be categorised into threshold based boolean classifiers and linear classifiers. One of the first generic approaches for entity resolution based on boolean classifiers is presented at [2]. The approach is based on the assumption that the entity resolution process consists of iterative matching and merging which results in a set of merged records that cannot be further matched or merged with each other. The authors also assume that matching and merging can be done if similar values exits, therefore their approach would not be able to match or merge records with missing values.

One of the most popular method to model distance-based entity resolution approaches is with linear classifiers. There are two popular applications of SVMs to entity matching MARLIN (Multiply Adaptive Record Linkage with INduction) [5] and FEBRL (Freely Extensible Biomedical Record Linkage) [9]. While there are numerous studies that propose approaches for handling missing values in SVMs, for instance [32], these optimizations are often expensive and to our knowledge are not used in matching approaches.

An important use cases of entity resolution is matching of product data. Following the same trend from above various studies show optimization approaches of linear classifiers for product resolution. For instance, Kannan et al. [18] learn a logistic regression model on product attributes extracted from a dictionary model. Similarly, in [20] the authors extend the FEBRL approach from [19] with more detailed features. Finally, in [35], the authors compare various classifiers for product resolution (SVMs, Random Forest, Naive Bayes) with features extracted from a dictionary method and multiple Conditional Random Fields (CRFs) models.

The entire process of entity resolution can be unsupervised [11, 24], supervised [28, 29], or a hybrid of these two [8, 17]. LIMES [29] and Silk [17] are examples of supervised entity resolution systems that focus on combining expressive comparisons with good run-time behavior. Both LIMES and Silk learn linkage rules employing similar genetic programming approaches, i.e EAGLE [31] and GenLink [17] respectively. As shown throughout this paper, both algorithms do not handle missing values well.

Contrary to the above, in Ngomo et al. [30], the authors present RAVEN - an entity resolution approach based on perceptron learning. Namely, RAVEN treats the discovery of link specifications as a classification problem. It discovers link specifications by first finding class and property mappings between knowledge

bases. Afterward, it computes linear and boolean classifiers that can be used as link specifications. However, similar to FEBRL, a limitation of RAVEN is that only linear and boolean classifiers can be learned, meaning that the potential of non-linear similarity aggregation [17] is not exploited.

There is another direction of work that is focused on collective entity resolution approaches. For instance, Bhattacharya and Getoor [4] proposed a novel relational clustering algorithm that uses both property and relational information between the entities of same type for determining the underlying entities. However, the defined cluster similarity measure depends primarily on property value similarity, thus missing values will have effect on the cluster similarity measure. Another collective entity resolution approach is introduced in [3] where the authors use an extended LDA model to perform entity resolution for authors and publications simultaneously.

In contrast, [25, 36] use probabilistic model for capturing the dependence among multiple matching decisions. Specifically, CRFs have been successfully applied to the entity resolution domain [25] and is one of the most popular approaches in generic entity resolution. On another hand, a well-founded integrated solution to the entity-resolution problem based on Markov Logic is proposed in [36]. However the approach apply the closed-world assumption, i.e., whatever is not observed is assumed to be false in the world.

CoSum [39] and idMesh [12] are two representative unsupervised graph based entity resolution approaches. CoSum and idMesh are both treating entity resolution are graph summarisation problem, i.e. building complex “super-nodes" as entity identifiers, based on the equality of various properties. The approaches would potentially have problems dealing with missing values when multiple “key” properties are sparse, since they depend on calculating equality between various properties.

7. Conclusion

This paper introduced three entity resolution methods for sparse data. Firstly, our approach introduces learning groups of matching rules that are intended to enlarge the property combination usage and thus increasing efficiency. Moreover, we presented a new operator to the GenLink algorithm: *selective aggregation operator*. This, enables lower matching scores in pairings with missing values which in turn boosts precision. Finally, we presented a method that integrated the central ideas from the previous two methods into one combination method. We evaluate these three methods on six different datasets, three of them are of the e-commerce domain (as one of the representative domains of sparse datasets), and the other three datasets are existing benchmark datasets. We show improvements of up to 16% F-measure compared to handwritten rules, up to 12% F-measure compared to the GenLink algorithm and up to 8% F-measure compared to standard machine learning techniques optimized for entity resolution (FEBRL). Additionally, we show that the method using *group matching rules* improves recall up to 15%, while *selective aggregation operators* mostly improve precision of up to 16%. The combination that encompasses these methods allows for improvement of up to 5% F-measure compared to the GenLinkGL and GenLinkSA themselves.

As a general conclusion, the high gains in F-measure clearly show that identity resolution systems should take the use case of sparse data into account and not only focus on dense datasets. When benchmarking and comparing systems, it is important to not only to use dense evaluation datasets, but also test on data exhibiting varying attribute density, such as the WDC Product Matching Gold Standard [34].

References

[1] Arasu, A., Götz, M., Kaushik, R.: On active learning of record matching packages. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. pp. 783–794. SIGMOD ’10, ACM, New York, NY, USA (2010)

[2] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: a generic approach to entity resolution. The VLDB Journal—The International Journal on Very Large Data Bases 18(1), 255–276 (2009)

[3] Bhattacharya, I., Getoor, L.: A Latent Dirichlet Model for Unsupervised Entity Resolution, pp. 47–58 (2006)

[4] Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data 1(1) (Mar 2007)

[5] Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 39–48. KDD ’03, ACM, New York, NY, USA (2003)

[6] Brizan, D.G., Tansel, A.U.: A. survey of entity resolution and record linkage methodologies. Communications of the IIMA 6(3), 5 (2015)

[7] de Carvalho, M.G., Gonçalves, M.A., Laender, A.H.F., da Silva, A.S.: Learning to deduplicate. In: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries. pp. 41–50. JCDL ’06, ACM, New York, NY, USA (2006)

[8] Christen, P.: Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 151–159. KDD ’08, ACM, New York, NY, USA (2008)

[9] Christen, P.: Febrl: A freely available record linkage system with a graphical user interface. In: Proceedings of the Second Australasian Workshop on Health Data and Knowledge Management - Volume 80. pp. 17–25. HDKM ’08, Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2008)

[10] Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE transactions on knowledge and data engineering 24(9), 1537–1555 (2012)
[11] Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 475–480. KDD '02, ACM, New York, NY, USA (2002)
[12] Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., De Meer, H.: idmesh: Graph-based disambiguation of linked data. In: Proceedings of the 18th International Conference on World Wide Web. pp. 591–600. WWW '09, ACM, New York, NY, USA (2009)
[13] Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. pp. 85–96. SIGMOD '05, ACM, New York, NY, USA (2005)
[14] Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. Knowledge and Data Engineering, IEEE Transactions on 19(1), 1–16 (2007)
[15] Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
[16] Hu, W., Chen, J., Cheng, G., Qu, Y.: Objectcoref & falcon-ao: Results for oaei 2010. In: Proceedings of the 5th International Conference on Ontology Matching - Volume 689. pp. 158–165. OM'10, CEUR-WS.org, Aachen, Germany, Germany (2010)
[17] Isele, R., Bizer, C.: Learning expressive linkage rules using genetic programming. Proceedings of the VLDB Endowment 5(11), 1638–1649 (2012)
[18] Kannan, A., Givoni, I.E., Agrawal, R., Fuxman, A.: Matching unstructured product offers to structured product specifications. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 404–412. KDD '11, ACM, New York, NY, USA (2011)
[19] Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. Proceedings of the VLDB Endowment 3(1-2), 484–493 (2010)
[20] Köpcke, H., Thor, A., Thomas, S., Rahm, E.: Tailoring entity resolution for matching product offers. In: Proceedings of the 15th International Conference on Extending Database Technology. pp. 545–550. EDBT '12, ACM, New York, NY, USA (2012)
[21] Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1188–1196. PMLR, Bejing, China (22–24 Jun 2014)
[22] Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., Jurafsky, D.: Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4), 885–916 (2013)
[23] Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure 405(2), 442–451 (1975)
[24] McCallum, A., Nigam, K., Ungar, L.H.: 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. pp. 169–178. KDD '00, ACM, New York, NY, USA (2000)
[25] McCallum, A., Wellner, B.: Conditional models of identity uncertainty with application to noun coreference. In: NIPS. pp. 905–912 (2004)
[26] Meusel, R., Petrovski, P., Bizer, C.: The WebDataCommons Microdata, RDFa and Microformat Dataset Series, pp. 277–292. Springer International Publishing, Cham (2014)
[27] Nentwig, M., Hartung, M., Ngomo, A.N., Rahm, E.: A survey of current link discovery frameworks. Semantic Web 8(3), 419–436 (2017), https://doi.org/10.3233/SW-150210
[28] Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. pp. 104–111. ACL '02, Association for Computational Linguistics, Stroudsburg, PA, USA (2002)
[29] Ngomo, A.C.N., Auer, S.: Limes: A time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three. pp. 2312–2317. IJCAI'11, AAAI Press (2011)
[30] Ngomo, A.C.N., Lehmann, J., Auer, S., Höffner, K.: Raven - active learning of link specifications. In: Proceedings of the 6th International Conference on Ontology Matching - Volume 814. pp. 25–36. OM'11, CEUR-WS.org, Aachen, Germany, Germany (2011), http://dl.acm.org/citation.cfm?id=2887541.2887544
[31] Ngonga Ngomo, A.C., Lyko, K.: Eagle: Efficient active learning of link specifications using genetic programming. The Semantic Web: Research and Applications pp. 149–163 (2012)
[32] Pelckmans, K., Brabanter, J.D., Suykens, J., Moor, B.D.: Handling missing values in support vector machine classifiers. Neural Networks 18(5âĂŞ6), 684 – 692 (2005), {IJCNN} 2005
[33] Petrovski, P., Bryl, V., Bizer, C.: Integrating product data from websites offering microdata markup. In: Proceedings of the 23rd International Conference on World Wide Web. pp. 1299–1304. WWW '14 Companion, ACM, New York, NY, USA (2014)
[34] Petrovski, P., Primpeli, A., Meusel, R., Bizer, C.: The WDC Gold Standards for Product Feature Extraction and Product Matching, pp. 73–86. Springer International Publishing, Cham (2017)
[35] Ristoski, P., Mika, P.: Enriching Product Ads with Metadata from HTML Annotations, pp. 151–167. Springer International Publishing, Cham (2016)
[36] Singla, P., Domingos, P.: Entity resolution with markov logic. In: Sixth International Conference on Data Mining (ICDM'06). pp. 572–582 (Dec 2006)
[37] Winkler, W.E.: Matching and record linkage. Business survey methods 1, 355–384 (1995)
[38] Winkler, W.E.: Methods for record linkage and bayesian networks. Tech. rep., Technical report, Statistical Research Division, US Census Bureau, Washington, DC (2002)
[39] Zhu, L., Ghasemi-Gol, M., Szekely, P., Galstyan, A., Knoblock, C.A.: Unsupervised Entity Resolution on Multi-type Graphs, pp. 649–667. Springer International Publishing, Cham (2016)