Unsupervised Learning of Link Discovery Configuration

Andriy Nikolov, Mathieu d'Aquin, Enrico Motta

Knowledge Media Institute, The Open University, Milton Keynes, UK
{a.nikolov, m.daquin, e.motta}@open.ac.uk

Abstract. Discovering links between overlapping datasets on the Web is generally realised through the use of fuzzy similarity measures. Configuring such measures is often a non-trivial task that depends on the domain, ontological schemas, and formatting conventions in data. Existing solutions either rely on the user's knowledge of the data and the domain or on the use of machine learning to discover these parameters based on training data. In this paper, we present a novel approach to tackle the issue of data linking which relies on the unsupervised discovery of the required similarity parameters. Instead of using labeled data, the method takes into account several desired properties which the distribution of output similarity values should satisfy. The method includes these features into a fitness criterion used in a genetic algorithm to establish similarity parameters that maximise the quality of the resulting linkset according to the considered properties. We show in experiments using benchmarks as well as real-world datasets that such an unsupervised method can reach the same levels of performance as manually engineered methods, and how the different parameters of the genetic algorithm and the fitness criterion affect the results for different datasets.

1 Introduction

Identity links between data instances described in different sources provide major added value of linked data. In order to facilitate data integration, newly published data sources are commonly linked to reference repositories: popular datasets which provide good coverage of their domains and are considered reliable. Such reference repositories (e.g., DBpedia or Geonames) serve as hubs: other repositories either link their individuals to them or directly reuse their URIs. However, establishing links between datasets still represents one of the most important challenges to achieve the vision of the Web of Data. Indeed, such a task is made difficult by the fact that different datasets do not share commonly accepted identifiers (such as ISBN codes), do not rely on the same schemas and ontologies (therefore using different properties to represent the same information) and often implement different formatting conventions for attributes.

Automatic data linking often relies on fuzzy similarity functions comparing relevant characteristics of objects in the considered datasets. More precisely, a data linking task can be specified as the evaluation of a *decision rule* establishing

whether two individuals should be considered equivalent, based on the value of a function aggregating the similarity comparisons of some properties of these individuals. In most systems, establishing the appropriate decision rule is left to the user, who needs to rely on his/her knowledge of the domain, of the data in both datasets, and on his/her intuition regarding the performance of various similarity functions in the considered linking situation. Other systems try to alleviate the issue of establishing the decision rule for linking by using machine learning techniques. They however require a substantial set of training data in the form of pre-established links within a subset of the considered datasets.

In this paper, we investigate the question: can a suitable decision rule for linking two datasets be learned without possessing labelled training data, based only on the characteristics of the datasets and on the distribution of similarity values amongst their instances? Our hypothesis is that in a scenario which involves establishing links to reference datasets, available information (e.g., knowledge that the datasets do not contain duplicates and have high degree of overlap) can provide sufficient evidence to learn a decision rule which would determine identity mappings between instances in two datasets with high accuracy. To learn such rules, we propose an approach based on a genetic algorithm, which evolves a set of initially random solutions to a problem according to a fitness criterion. Following research in the area of record linkage in databases, we devise an applicable fitness criterion which relies on the distribution of links and similarity values generated by applying a particular decision rule.

To test our assumptions, we apply this approach to the benchmark datasets from the OAEI 2010 and 2011 instance matching contests. We show that applying the learned decision rule for data linking achieves results at the level of the best state-of-the-art tools, without the need to configure linking parameters for each task. We also experiment with subsets of real-world linked datasets to demonstrate the robustness of the approach to different types of datasets in different domains and discuss the effects of some of the parameters of the genetic algorithm on its behaviour in data linking tasks. The remainder of this paper is structured as follows. In section 2, we provide an overview of the basic notions of the link discovery problem and relevant work in both Semantic Web and database research communities. Section 3 describes our algorithm in detail. Section 4 describes the experiments we performed in order to validate our approach. Section 5 concludes the paper and discusses directions for future work.

2 Problem Definition and Related Work

In this section, we specify the tasks of link discovery and of establishing the necessary decision rule, together with a brief description of the relevant existing work.

2.1 Link Discovery Problem

The problem of reconciliation was originally studied in the database community where it is known as record linkage or object identification [3]. With the devel-

opment of the linked data initiative, it gains importance in the Semantic Web community where it is studied under the name of link discovery [14]. The link discovery task takes as inputs two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ and tries to discover all pairs of individuals (I_{1i}, I_{2j}) belonging to these datasets such that they describe the same entity ω according to a chosen identity criterion. In the context of linked data, datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ represent RDF graphs and their individuals are identified by URIs.

Existing techniques solving this task can be divided into two main categories: *individual matching* and *dataset matching*. We essentially focus on individual matching in this paper. *Dataset matching* techniques are built on top of individual matching ones: they take as input two datasets as a whole together with the initial set of mappings produced by individual matching and further refine them. These techniques take into account additional available information such as relations between individuals, axioms defined in the ontological schema, and mutual impact of different mappings. The individual matching task can be defined as follows.

Definition 1*: Let $I_{1i} \in \mathcal{I}_1$ and $I_{2j} \in \mathcal{I}_2$ represent two individuals in instance sets $\mathcal{I}_1$ and $\mathcal{I}_2$. The individual matching task takes I_{1i} and I_{2j} as input and makes a decision whether $I_{1i} \equiv I_{2j}$ (in which case they are said to be matching) or not. This decision is made based on the comparison of the profiles of two individuals. A* ***profile*** *$P(I)$ is defined as a set of pairs $\{(a_i, V_i)\}$, where a_i represent attributes describing an individual (e.g., name, age, colour, etc.), each of which has a set of values V_i. The output of individual matching is a set of mappings $M = \{(I_{1i}, I_{2j})\}$ believed to represent equivalent individuals $I_{1i} \equiv I_{2j}$.*

Most individual matching techniques follow the approach proposed in a seminal paper by Fellegi & Sunter [6], in which the decision is based on a **similarity function** $sim(P(I_1), P(I_2))$ which returns a degree of confidence that $I_1 \equiv I_2$. The similarity function commonly takes the form of aggregated similarity over attributes $sim(P(I_1), P(I_2)) = f_{agg}(\{sim_i(V_{1i}, V_{2i})\})$, where f_{agg} is an aggregation function and sim_i is a comparison function, which returns a degree of similarity between two values of the attribute a_i. The decision rule then takes the form of applying a **filtering criterion** which determines whether the confidence degree returned by the similarity function is sufficient to consider a pair of individuals as identical. The threshold-based criterion is commonly used: a mapping (I_1, I_2) is returned if $sim(P(I_1), P(I_2)) \geq t$, where t is a threshold.

2.2 Establishing a Decision Rule for Individual Matching

As can be seen from the description above, the key component of an individual matching method is the decision rule. For a given pair of datasets to link, a decision rule has to be established that incorporates comparisons between relevant pairs of properties using appropriate similarity functions, weights, and thresholds to obtain an adequate discriminative ability.

Some systems assume that a pre-established, generic similarity measure can be employed across domains. This approach is often followed by systems targeted for the global scale link discovery (e.g., OKKAM [13]), generic ontology

matching systems (e.g., RiMOM [9]), or systems which primarily rely on the dataset matching stage (e.g., CODI [12]). However, in most other cases, a dedicated decision rule has to be established for each link discovery task (i.e., each pair of datasets to link). Existing systems in the Semantic Web area take two different approaches to realise this:

Manual configuration where the decision rule is specified by the user. Besides requiring user effort, the clear disadvantage of such an approach is that it relies on extensive knowledge from the user of the structure and content of the two datasets to link, as well as on a reasonable level of intuition regarding the performance of (often complex) similarity functions in a particular situation.

Learning from training data where the appropriate decision rule is produced by analyzing the available labeled data. This method is followed, for example, by the ObjectCoref system [7]. This alleviates the need for user input to establish the decision rule, but requires the availability of a substantial set of robust training data (although some methods, like active learning [10] can reduce the required amount of data).

Here we investigate a third category of approaches that relies on the characteristics of the datasets and of the similarity distributions resulting from comparing them to establish high performing decision rules in an unsupervised way. Several solutions in the database research community proposed to use the distribution features of similarity functions. For example, in [2] individuals are clustered into matching and non-matching classes based on the structure of their neighbourhood rather than on simple threshold filtering. Zardetto et al [15] proposed to use prior knowledge about the features of the similarity distribution – namely, that correct mappings are dominant in the area of high similarity values and that matches are very rare in comparison with non-matches. These features are used to build a mixture model, which is later used for classifying candidate mappings into matching and non-matching.

Considering the task of linking to a reference repository, we can make several assumptions about the datasets and the desired instance matching output:

- ***Assumption 1***: While different URIs are often used to denote the same entity in different repositories, distinct URIs within one dataset can be expected to denote distinct entities.
- ***Assumption 2***: Datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ have a strong degree of overlap.
- ***Assumption 3***: A meaningful similarity function produces results in the interval $\{0..1\}$ and returns values close to 1.0 for pairs of matching individuals.

The method described in this paper proposes to use a genetic algorithm guided by a fitness criterion using these assumptions to assess the expected quality of a decision rule, and of the derived set of links. Our method goes a step further than existing methods, as it chooses an appropriate similarity function for a given matching task as well as a suitable filtering criterion, rather than

relying on given similarity functions. Hence, producing a solution requires selecting multiple parameters of the decision rule simultaneously, such as similarity functions, comparable attributes, and weights.

For such problems where a suitable complex function has to be found based on its desired output, genetic algorithms are known to perform well on many practical tasks, and have already been applied to the instance matching problem in the context of supervised learning [1], [8]. The idea here is to use such an approach to evolve a population of candidate solutions (i.e., decision rules) using selection and variation mechanisms to favour the "fittest" solutions in each generation, therefore presumably converging to decision rules that can be optimally applied to link the two given datasets.

3 Algorithm

Applying a genetic algorithm to the problem of optimizing a decision rule requires solving three issues: how relevant parameters of a decision rule are encoded as a set of genes, what fitness measure to use to evaluate candidate solutions, and how to use selection and variation operators to converge on a good solution.

3.1 Representing individual matching in terms of a genetic algorithm

Definition 2*: Let C_i represent a candidate solution to a given optimization task $\mathcal{T}$[1]. Assume that C_i can be encoded as a set of numeric parameters. Then, the term* ***gene*** *g_{ij} denotes the jth parameter of the candidate solution C_i,* ***genotype*** *or* ***chromosome*** *$G(C_i) =< g_{i1}, \ldots, g_{in} >$ denotes a set of genes representing a candidate solution C_i, and* ***population*** *$\mathcal{G} = \{G_1, \ldots G_N\}$ represents a set of N chromosomes encoding candidate solutions for the task. A* ***fitness function*** *$F_{fit}(C_i)$ is a function which provides an estimation of the quality of a solution.*

An initial population is used as a pool of candidates, from which the algorithm selects the best chromosomes according to the fitness function. In order to find a solution which optimizes the fitness function, the algorithm updates the initial population by using *selection* and *variation* operators:

- *Selection* chooses a subset of chromosomes in the original population to be used in the creation of the new one.
- *Variation* changes the genes of the selected chromosomes to generate new candidate solutions from the old ones. Commonly used variation operators include *crossover*, which recombines elements of several "parent" chromosomes to produce several new chromosomes (or "children"), and *mutation*, which produces a new chromosome by randomly tweaking the genes of the original one.

[1] The term "individual" is used both in the Semantic Web domain to denote ontological instances and in the evolutionary computation area, where it refers to candidate solutions. To avoid confusion, we use it only in its first sense, while using the term "candidate solution" when talking about the output of the genetic algorithm.

The updated population is created by applying these operators to selected chromosomes from the original one. Then, the same steps are performed for the updated population, and the algorithm continues iterating until the optimal solution (or one sufficiently close to the optimum) is produced or a termination condition is satisfied: e.g. maximal number of iterations is reached or the fitness of the population does not improve for a long time. The candidate solution $C_{best} = argmax(F_{fit}(C_i))$ is returned by the algorithm as its output.

To apply a genetic algorithm to the individual matching problem, we need to represent candidate decision rules as a set of genes. Similarly to many existing approaches (see section 2), we represent a decision rule using an aggregated attribute similarity function.

***Definition 3**: A **decision rule** for an individual matching task is defined as: $filt(sim(P(I_1), P(I_2)))$ where $sim(P(I_1), P(I_2))$ is the **similarity function** comparing profiles of two individuals, and $filt(sim(P(I_1), P(I_2)))$ is a boolean **filtering function**. The similarity function takes the form*

$$sim(P(I_1), P(I_2)) = f_{agg}(w_{11}sim_{11}(V_{11}, V_{21}), \ldots, w_{mn}sim_{mn}(V_{1m}, V_{2n}))$$

- *sim_{ij} is the function which measures similarity between the values of the attributes a_{1i} of $P(I_1)$ and a_{2j} of $P(I_2)$,*
- *w_{ij} is a numeric weight ($0 \leq w_{ij} \leq 1$),*
- *f_{agg} is an aggregation function.*

We considered two alternative filtering criteria: the *threshold-based* one and the *nearest neighbour* one. The former requires that $sim(P(I_1), P(I_2)) \geq t$, where t is a threshold value. The latter chooses for each instance I_1 in the source dataset such I_2 that $sim(P(I_1), P(I_2)) = max(sim(P(I_1), P(I_j)))$. This criterion is applicable in cases where we expect each I_1 to have a matching I_2.

Each of these parameters is represented by a gene in the following way:

- sim_{ij} are encoded as nominal values representing corresponding attribute similarity functions (or *nil*, if a_{1i} and a_{2j} are not compared). We included a number of character-based functions (edit distance, Jaro, I-Sub, etc., and the corresponding token-based similarity metrics. The latter divide both string values into sets of tokens, then compare each pair of tokens using a character-based similarity function and try to find the best match between them.
- Weights of each attribute comparison pair w_{ij} and the threshold t are encoded using their real values.
- f_{agg} is encoded as a nominal value representing one of two types of aggregation functions: weighted average $avg(P(I_1), P(I_2)) = \frac{\sum w_{ij} sim_{ij}(a_{1i}, a_{2j})}{\sum w_{ij}}$ and maximum $max(P(I_1), P(I_2)) = max(\{w_{ij}sim_{ij}(a_{1i}, a_{2j})\})$. In the latter case the weights w_{ij} can only take values 0 or 1.

These genotypes are evaluated by applying the decision rule to the matching task and calculating the fitness function.

3.2 Fitness functions: pseudo-F-measure and neighbourhood growth

In the absence of labelled data it is not possible to estimate the quality of a set of mappings accurately. However, there are indirect indicators corresponding to "good characteristics" of sets of links which can be used to assess the fitness of a given decision rule. To establish such indicators, we rely on the assumptions we made about the matching task. Traditionally, the quality of the matching output is evaluated by comparing it with the set of true mappings M^t and calculating the precision p and recall r metrics. Precision is defined as $p = \frac{|tp|}{|tp|+|fp|}$, where tp is a set of true positives (mappings $m = (I_1, I_2)$ such that both $m \in M$ and $m \in M^t$) and fp is a set of false positives ($m \in M$, but $m \notin M^t$). Recall is calculated as $r = \frac{|tp|}{|tp|+|fn|}$, where fn is a set of false negatives ($m \notin M$, but $m \in M^t$). In the absence of gold standard mappings, we use Assumption 1 to formulate the pseudo-precision and pseudo-recall measures in the following way:

Definition 4: *Let M represent a set of mappings (I_i, I_j) between two sets of individuals $\mathcal{I}_1$, $\mathcal{I}_2$ such that $I_i \in \mathcal{I}_1$, $I_j \in \mathcal{I}_2$. Then,* ***pseudo-precision*** *is the value $p^{\sim} = \frac{|\{I_i | \exists I_j : (I_i, I_j) \in M\}|}{\sum_i |\{I_j | (I_i, I_j) \in M\}|}$, and* ***pseudo-recall*** *is the value $r^{\sim} = \frac{|M|}{min(|\mathcal{I}_1|, |\mathcal{I}_2|)}$.*

In an ideal case where $p = 1$, if Assumption 1 holds, then $p^{\sim} = 1$: of two mappings from the same individual one is necessarily an error. Similarly, in case where $r = 1$, the number of returned mappings will be equal to the size of the overlap between two instance sets $|M| = n_o = |\mathcal{I}_1 \cap \mathcal{I}_2|$, and the *pseudo-recall* value $r^{\sim} = \frac{|M|}{n_o} = 1$. However, estimating the true recall is problematic since n_o is not known in advance. From Assumption 1 it follows that $n_o \leq min(|\mathcal{I}_1|, |\mathcal{I}_2|)$, while $n_o = min(|\mathcal{I}_1|, |\mathcal{I}_2|)$ if one instance set is a subset of another. Incorrect estimation of n_o can be misleading for the genetic algorithm: it can result in "lenient" decision rules being favored and, in consequence, to many false positives in the resulting solution. To deal with such cases, we reduce the impact of incorrect recall estimations in the final fitness function.

A standard metric combining precision and recall is the F-measure $F_\beta = \frac{(1+\beta^2) \cdot p \cdot r}{\beta^2 \cdot p + r}$, where β characterizes the preference of recall over precision, and $\beta = 1$ means equal importance of both. To reduce the impact of recall, we used $\beta = 0.1$ and the pseudo-F-measure $F^{\sim}_{0.1} = \frac{1.01 p^{\sim} \cdot r^{\sim}}{0.01 \cdot p^{\sim} + r^{\sim}}$. In this way, solutions which increase precision are favored, while recall is only used to discriminate between solutions with similar estimated precision. This "cautious" approach is also consistent with the requirements of many real-world data linking scenarios, as the cost of an erroneous mapping is often higher than the cost of a missed correct mapping.

In order to incorporate Assumption 3, the final fitness function gives a preference to the solutions which accept mappings with similarity degrees close to 1: $F^{\sim}_{fit} = F^{\sim}_{0.1} \cdot (1 - (1 - sim_{avg})^2)$. In this way, the fitness function is able to discriminate between such decision rules as $avg(0.5 \cdot jaro(name, label), 0.5 \cdot edit(birthYear, yearOfBirth)) \geq 0.98$ and $avg(0.05 \cdot jaro(name, label), 0.05 \cdot edit(birthYear, yearOfBirth), 0.9 \cdot edit(name, yearOfBirth)) \geq 0.098$. While

these two rules would produce the same output in most cases, comparing irrelevant attributes (like *name* and $yearOfBirth$) is not desirable, because it increases a possibility of spurious mappings without adding any value.

While we used $F^{\sim}_{fit}$ as the main fitness criterion, to test the effect of the choice of a fitness function on the performance of the genetic algorithm, we implemented an alternative fitness function: the *neighbourhood growth* function F^{NG}_{fit}. While the pseudo F-Measure tries to estimate the quality of resulting mappings to guide the evolution of candidate solutions, F^{NG}_{fit} tries to exploit the desired property of a "good" similarity function: namely, that it should be able to discriminate well between different possible candidate mappings. To measure this property, we adapt the neighbourhood growth indicator defined in [2] to achieve an optimal clustering of instance matching results for a pre-defined similarity function, as an alternative to the threshold-based filtering criterion. We adapt this indicator as an alternative fitness criterion for selecting the most appropriate similarity functions.

Definition 5: *Let M_x represent a set of mappings (I_x, I_{xj}) between an individual $I_i \in \mathcal{I}_1$ and a set of individuals $I_{xj} \in \mathcal{I}_{\S|} \subseteq \mathcal{I}_{\in}$. Let $sim_{max} = max(sim(P(I_x), P(I_{xj})))$ Then,* ***neighbourhood growth*** *$NG(I_x)$ is defined as the number of mappings in M_x such that their similarity values are higher than $1 - c \cdot (1 - sim_{max})$, where c is a constant.*

Intuitively, high values of $NG(I_x)$ indicate that the neighbourhood of an instance is "cluttered", and the similarity measure cannot adequately distinguish between different matching candidates. Then the fitness function for a set of compared instance pairs M is defined as $F^{NG}_{fit} = 1/avg_x(NG(I_x))$. As this function does not require applying the filtering criterion, it only learns the similarity function, but not the optimal threshold. However, the threshold can be determined after the optimal similarity function has been derived: t is selected in such a way that it maximises the $F^{\sim}_{fit}$ function over a set of compared pairs.

3.3 Obtaining the optimal solution: genetic algorithm

The algorithm takes as input two instance sets $\mathcal{I}_1$ and $\mathcal{I}_2$ and two sets of potential attributes A_1 and A_2. Each set of attributes A_i includes all literal property values at a distance l from individuals in $\mathcal{I}_i$. In our experiments we used $l = 1$, however, also including the paths of length 2 if an individual was connected to a literal through a blank node. In order to filter out rarely defined properties, we also remove all attributes a_{ij} for which $\frac{|\{P(I_i)|a_{ij} \in P(I_i), I_i \in \mathcal{I}\}|}{|\mathcal{I}|} < 0.5$.

As the first step, the algorithm initializes the population of size N. For the initial population, all values of the genotype are set in the following way:

- A set of k pairs of attributes (a_{1i}, a_{2j}) is selected randomly from the corresponding sets A_1 and A_2.
- For these pairs of attributes the similarity functions sim_{ij} and the corresponding weights w_{ij} are assigned randomly while for all others are set to nil.

– The aggregation function and the threshold are initialized with random values, and the weights are normalized so that $\sum w_{ij} = 1$.

All initial solutions only compare a single pair of attributes ($k = 1$): this is done to identify highly discriminative pairs of attributes at the early iterations, and then improve these solutions incrementally.

Each iteration of the algorithm consists of two stages: selection and reproduction. At the selection stage, each candidate solution is applied to produce mappings between individuals from $\mathcal{I}_1$ and $\mathcal{I}_2$. In case of large-scale datasets, random sampling can be applied, so that the solutions are only applied to a subset $\mathcal{I}_1^S \subseteq \mathcal{I}_1$. The calculated F_{fit} fitness measure is used for the selection of candidate solutions for reproduction. Our algorithm uses the standard roulette wheel selection operator: the probability of a chromosome being selected is proportionate to its F_{fit} fitness. At the reproduction stage, a new population of chromosomes is generated by three different operators: elitist selection, crossover, and mutation. In the new population, the proportion of chromosomes produced by each operator is proportional to its rate: elitist selection rate r_{el}, crossover rate r_c, and mutation rate r_m ($r_{el} + r_c + r_m = 1$). Elitist selection copies the best subset of chromosomes from the previous population. The crossover operator takes two parent chromosomes and forms a pair of "children": each gene of the parent is passed to a randomly chosen child, while another child inherits a corresponding gene of the second parent. Finally, mutation modifies one of the genes of the original chromosome in one of the following ways:

– Adding or removing a comparison between attributes with a probability p_{att}^m. The operator either changes the similarity function for a pair of attributes to *nil* or selects a random similarity function and weight for a pair of attributes not compared in the original chromosome. The probability of adding a component (versus removing one) is calculated as $p_{add} = \frac{1}{n^+}$, where n^+ is the number of non-*nil* similarity comparisons in the original solution.
– Changing one of the weights w_{ij} for a pair of attributes where $sim_{ij} \neq nil$, with a probability p_{wgt}^m. The value of the change is calculated as $\frac{0.8 \cdot rnd + 0.2}{n^+}$, where rnd is a random number between 0 and 1.
– Changing a non-*nil* similarity function for a pair of attributes into a randomly selected one with a probability p_{sym}^m.
– Modifying the threshold value with the probability p_t^m: the algorithm decides whether the current threshold should be increased or decreased with the probability 0.5. The new threshold is set as $t_{new} = t_{old} \pm \Delta t$, where $\Delta t = rnd \cdot (1 - p^{\sim})(1 - t_{old})$ for increase and $rnd \cdot (1 - r^{\sim})t_{old}$ for decrease. The rationale behind this is to make bigger steps if precision/recall values are far from desired.
– Changing the aggregation function with p_{agg}^m.

At the new iteration, chromosomes in the updated population are again evaluated using the F_{fit} fitness function, and the process is repeated. The algorithm stops if the pre-defined number of iterations n_{iter} is reached or the algorithm

converges before this: i.e., the average fitness does not increase for n_{conv} generations. The phenotype with the best fitness in the final population is returned by the algorithm as its result.

4 Evaluation

To validate our method, we performed experiments with two types of datasets. First, we tested our approach on the benchmark datasets used in the instance matching tracks of the OAEI 2010 and OAEI 2011 ontology matching competitions[2], to compare our approach with state-of-the-art systems. Second, we used several datasets extracted from the linked data cloud to investigate the effect of different parameter settings on the results.

4.1 Settings

As discussed above, a genetic algorithm starts with an initial population of random solutions, and iteratively create new generations through selection, mutation and crossover. In our experiments, we used the following default parameters:

- rates for different recombination operators: $r_{el} = 0.1$, $r_m = 0.6$, and $r_c = 0.3$.
- rates for different mutation options: $p^m_{att} = 0.3$, $p^m_{wgt} = 0.15$, $p^m_{sym} = 0.15$, $p^m_t = 0.3$, $p^m_{agg} = 0.1$ (ensuring equivalent probabilities for modifying the list of compared properties, comparison parameters, and the threshold).
- termination criterion: $n_{iter} = 20$ (found to be sufficient for convergence in most cases).
- fitness function: $F^{\sim}_{fit}$, except when comparing $F^{\sim}_{fit}$ with F^{NG}_{fit}

The genetic algorithm is implemented as a method in the KnoFuss architecture [11]. Relevant subsets of two datasets are selected using SPARQL queries. Each candidate decision rule is used as an input of the KnoFuss tool to create the corresponding set of links. To reduce the computation time, an inverted Lucene[3] index was used to perform *blocking* and pre-select candidate pairs. Each individual in the larger dataset was indexed by all its literal properties. Each individual in the smaller dataset was only compared to individuals returned by the index when searching on all its literal properties, and pairs of compared individuals were cached in memory. Common pre-processing techniques (such as removing stopwords and unifying synonyms) were applied to the literal properties.

4.2 Benchmark test

The OAEI 2010 benchmark contains three test cases: *Person1* and *Person2*, which contain artificially distorted records of people, and *Restaurants*, which includes data about restaurants from the RIDDLE repository[4]. Two versions

[2] http://oaei.ontologymatching.org/
[3] http://lucene.apache.org
[4] http://www.cs.utexas.edu/users/ml/riddle/data.html

Table 1. Comparison of F1-measure with other tools on the OAEI 2010 benchmark [4].

Dataset	KnoFuss+GA	ObjectCoref	ASMOV	CODI	LN2R	RiMOM	FBEM
Person1	**1.00**	**1.00**	**1.00**	0.91	**1.00**	**1.00**	N/A
Person2	**0.99**	0.95	0.35	0.36	0.94	0.97	0.79
Restaurant (OAEI)	0.78	0.73	0.70	0.72	0.75	**0.81**	N/A
Restaurant (fixed)	**0.98**	0.89	N/A	N/A	N/A	N/A	0.96

of the *Restaurants* dataset exist: the version originally used in the OAEI 2010 evaluation which contained a bug (some individuals included in the gold standard were not present in the data), and the fixed version, which was used in other tests (e.g, [13], [7]). To be able to compare with systems which used both variants of the dataset, we also used both variants in our experiments. The OAEI 2011 benchmark includes seven test cases, which involve matching three subsets of the New York Times linked data (people, organisations, and locations) with DBpedia, Freebase, and Geonames datasets.

We compared our algorithm with the systems participating in the OAEI 2010 tracks as well as with the FBEM system [13], whose authors provided the benchmark datasets for the competition. We report in Table 1 on the performance of the KnowFuss system using decision rules learned through our genetic algorithm (noted KnowFuss+GA) as the average F1-Measure obtained over 5 runs of the algorithm with a population size $N = 1000$. The solution produced

Table 2. Example decision rules found by the algorithm with $N = 1000$.

Test case	Similarity function	Threshold
Person1	max(tokenized-jaro-winkler(soc_sec_id;soc_sec_id); monge-elkan(phone_number;phone_number))	$\geq$0.87
Person2	max(jaro(phone_number;phone_number); jaro-winkler(soc_sec_id;soc_sec_id))	$\geq$0.88
Restaurants (OAEI)	avg(0.22*tokenized-smith-waterman(phone_number;phone_number); 0.78*tokenized-smith-waterman(name;name))	$\geq$0.91
Restaurants (fixed)	avg(0.35*tokenized-monge-elkan(phone_number;phone_number); 0.65*tokenized-smith-waterman(name;name))	$\geq$0.88

by the genetic algorithm managed to achieve the highest F1-measure on 3 out of 4 datasets and the second highest F1-measure on 1 out of 4. Examples of produced decision rules are provided in Table 2. We observed that the algorithm took less time on identifying discriminative pairs of properties and the aggregation function and more on tuning weights and attribute similarity functions. To test the robustness of the results achieved by the algorithm with different settings, we performed tests on the benchmark datasets varying the crossover rates r_c, and mutation rate r_m. Surprisingly, varying the crossover rate and the mutation rate did not lead to significant changes in the results, except for extreme values. These parameters mostly affected the number of generations needed to

Table 3. Comparison of F1-measure with other tools on the OAEI 2011 benchmark [5].

Dataset	KnoFuss+GA	AgreementMaker	SERIMI	Zhishi.links
DBpedia (locations)	0.89	0.69	0.68	**0.92**
DBpedia (organisations)	**0.92**	0.74	0.88	0.91
DBpedia (people)	**0.97**	0.88	0.94	**0.97**
Freebase (locations)	**0.93**	0.85	0.91	0.88
Freebase (organisations)	**0.92**	0.80	0.91	0.87
Freebase (people)	0.95	**0.96**	0.92	0.93
Geonames	0.90	0.85	0.80	**0.91**
Average	**0.93**	0.85	0.89	0.92

converge to the optimal solution, and the algorithm usually converged well before 20 generations[5].

Given the larger scale of the OAEI 2011 benchmark, to speed up the algorithm we used random sampling with the sample size $s = 100$ and reduced the population size to $N = 100$. To improve the performance, a post-processing step was applied: the 1-to-1 rule was re-enforced, and for a source individual only 1 mapping was retained. As shown in Table 3, these settings were still sufficient to achieve high performance: the algorithm achieved the highest $F1$ measure on 4 test cases out of 7 and the highest average $F1$ measure. These results verify our original assumptions that (a) the fitness function based on the pseudo-F-measure can be used as an estimation of the actual accuracy of a decision rule and (b) the genetic algorithm provides a suitable search strategy for obtaining a decision rule for individual matching.

4.3 LOD datasets

To test the reusability of our method in different real-world scenarios, we have defined the following three matching tasks:

Music contributors. As a source dataset, we selected a list of music contributors from the LinkedMDB dataset[6]. This dataset of 3995 individuals was matched against the set of all people from DBpedia[7] (363751 individuals). The gold standard was constructed manually and included 1182 mappings.

Book authors. To construct this dataset, we extracted a set of 1000 individuals describing book authors from the BNB dataset[8] (from the first part of the dump, we selected 1000 authors with the highest number of published books). This dataset was also matched against the set of all people from DBpedia. The gold standard was constructed manually and included 219 correct mappings.

Research papers. To generate a matching task with a larger number of reliable gold standard mappings, we used a subset of 10000 research publications

[5] The datasets and test results are available for download from our website: http://kmi.open.ac.uk/technologies/knofuss/knofuss-GA-tests.zip

[6] http://www.linkedmdb.org/

[7] http://dbpedia.org

[8] http://www.archive.org/details/Bibliographica.orgBnbDataset

represented in the L3S-DBLP dataset[9] (out of the snapshot of 366113 publications included in the BTC 2010 dataset[10]). For these publications, we extracted their RDF descriptions from the DOI web-site[11]. We used equivalent DOI codes to create the gold standard and then removed corresponding properties from respective datasets to prevent the algorithm from using them as an easy solution.

On each of these datasets, we applied the algorithm with the same default settings as used in the benchmark tests. We performed the experiments using two different fitness functions: the unsupervised $F_{fit}^{\sim}$ fitness function and the actual $F1$-measure produced using the gold standard dataset. The latter case represents an ideal scenario, in which a complete set of labeled data is available in advance, and the algorithm only has to produce an optimal decision rule which would approximate this data. For *Music contributors* and *Book authors*, we varied the population size N in order to estimate the necessary number of candidate solutions which the algorithm has to test before achieving stable performance. The results for these datasets are summarised in Table 4, which shows average precision, recall, and $F1$-measure achieved using two different fitness functions, as well as the standard deviation of $F1$ measure $\sigma F1$ over 5 runs and the time of a single run for the unsupervised case[12]. In both cases, $F_{fit}^{\sim}$ allowed reaching

Table 4. Results with different population size.

Dataset	Pop. size N	F1-fitness (ideal case)			$F_{fit}^{\sim}$-fitness (unsupervised)				
		Precision	Recall	F1	Precision	Recall	F1	$\sigma F1$	Time (s)
Music contributors	50	0.92	0.92	0.92	0.90	0.90	0.90	0.021	520
	100	0.91	0.93	0.92	0.92	0.91	0.92	0.003	931
	500	0.91	0.93	0.92	0.92	0.92	0.92	0.003	4197
Book authors	50	0.90	0.93	0.91	0.66	0.69	0.68	0.022	753
	100	0.98	0.95	0.97	0.78	0.89	0.82	0.13	1222
	500	0.99	0.98	0.98	0.91	0.91	0.91	0.009	7281

high performance ($F1$ above 0.9), and increasing the population size N led to improvement in performance as well as more robust results (lower $\sigma F1$). In fact, for the *Music contributors* test case, the results produced using $F_{fit}^{\sim}$ and the ideal case $F1$ were almost equivalent. For the *Research papers* dataset (Table 5), we trained the algorithm on several samples taken from the DOI dataset and then applied the resulting decision rules to the complete test case (10000 individuals in the DOI dataset). This was done to emulate use cases involving large-scale repositories, in which running many iterations of the genetic algorithm over complete datasets is not feasible. From Table 5 we can see that starting from 100 sample individuals the algorithm achieved stable performance, which is consistent

[9] http://dblp.l3s.de/

[10] http://km.aifb.kit.edu/projects/btc-2010/

[11] http://dx.doi.org/

[12] Experiments were performed on a Linux desktop with two Intel Core 2 Duo processors and 3GB of RAM

Table 5. Results obtained for the *Research papers* dataset (for all sample sizes, population size $N = 100$ was used).

Sample size	F1-fitness (ideal case)			$F^{\sim}_{fit}$-fitness (unsupervised)					Complete set		
	Precision	Recall	F1	Precision	Recall	F1	$\sigma F1$	Time (s)	Precision	Recall	F1
50	0.50	0.76	0.60	0.58	0.36	0.44	0.063	162	0.68	0.22	0.33
100	0.95	0.88	0.91	0.998	0.72	0.83	0.068	255	0.995	0.68	0.81
500	0.96	0.85	0.90	0.99	0.73	0.84	0.046	842	0.98	0.75	0.85
1000	0.95	0.88	0.91	0.99	0.67	0.79	0.065	3667	0.997	0.71	0.83

with the results achieved for the OAEI 2011 benchmark. Applying the resulting decision rules to the complete dataset also produced results with precision and recall values similar to the ones achieved on the partial sample. Finally, to test

Table 6. Comparing the $F^{\sim}_{fit}$ and F^{NG}_{fit} fitness functions.

Dataset	$F^{\sim}_{fit}$-fitness			NG-fitness		
	Precision	Recall	F1	Precision	Recall	F1
Music contributors	0.92	0.91	0.92	0.90	0.91	0.91
Book authors	0.78	0.89	0.82	0.97	0.78	0.85
NYT-Geonames	0.88	0.82	0.84	0.87	0.92	0.89
NYT-Freebase (people)	0.60	0.97	0.74	0.47	0.66	0.55

the effect of the chosen fitness function on the performance, we compared the pseudo-F-measure $F^{\sim}_{fit}$ and neighbourhood growth F^{NG}_{fit} fitness functions. We applied the algorithm to the *Music contributors* and *Book authors* datasets, as well as to the NYT-Geonames and NYT-Freebase (people) test cases from the OAEI 2011 benchmark (without applying post-processing). The results reported in Table 6 show that both functions are able to achieve high accuracy with $F^{\sim}_{fit}$ providing more stable performance. This validates our initial choice of $F^{\sim}_{fit}$ as a suitable fitness criterion and reinforces our assumption that features of the similarity distribution can indirectly serve to estimate the actual fitness.

5 Conclusion and future work

In this paper, we proposed a method which exploits expected characteristics of "good" sets of mappings to estimate the quality of results of the individual matching task. We formalised these characteristics to propose a fitness function for a genetic algorithm, which derives a suitable decision rule for a given matching task. Experiments, which we performed with both benchmark and real-world datasets, have validated our initial assumptions and have shown that the method is able to achieve accuracy at the level of the top-performing state-of-the-art data linking systems without requiring user configuration, training data, or external knowledge sources.

We plan to use the results presented in this paper to pursue several promising research directions, in particular, combining our approach with more knowledge-

involving dataset matching methods. On the one hand, dataset matching systems have to rely on individual matching techniques to provide initial sets of mappings for refining. For such systems, using initial mappings of better quality can be beneficial. On the other hand, domain knowledge can be used to improve the unsupervised fitness functions, for example to reduce the fitness of decision rules whose results violate ontological restrictions.

6 Acknowledgements

Part of this research has been funded under the EC 7th Framework Programme, in the context of the SmartProducts project (231204).

References

1. de Carvalho, M.G., Laender, A.H.F., Goncalves, M.A., da Silva, A.S.: A genetic programming approach to record deduplication. IEEE Transactions on Knowledge and Data Engineering 99(PrePrints) (2010)
2. Chaudhuri, S., Ganti, V., Motwani, R.: Robust identification of fuzzy duplicates. In: ICDE 2005. pp. 865–876 (2005)
3. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
4. Euzenat, J., et al.: Results of the ontology alignment evaluation initiative 2010. In: Workshop on Ontology Matching (OM 2010), ISWC 2010 (2010)
5. Euzenat, J., et al.: Results of the ontology alignment evaluation initiative 2011. In: Workshop on Ontology Matching (OM 2011), ISWC 2011 (2011)
6. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of American Statistical Association 64(328), 1183–1210 (1969)
7. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: WWW 2011. pp. 87–96 (2011)
8. Isele, R., Bizer, C.: Learning linkage rules using genetic programming. In: Workshop on Ontology Matching (OM 2011), ISWC 2011. Bonn, Germany (2011)
9. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21(8), 1218–1232 (2009)
10. Ngonga Ngomo, A.C., Lehmann, J., Auer, S., Höffner, K.: RAVEN - active learning of link specifications. In: Workshop on Ontology Matching (OM 2011), ISWC 2011 (2011)
11. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: Integration of semantically annotated data by the KnoFuss architecture. In: EKAW 2008 (2008)
12. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging terminological structure for object reconciliation. In: ESWC 2010. pp. 334–348 (2010)
13. Stoermer, H., Rassadko, N., Vaidya, N.: Feature-based entity matching: The FBEM model, implementation, evaluation. In: CAISE 2010. pp. 180–193 (2010)
14. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the Web of Data. In: ISWC 2009. pp. 650–665. Washington, DC, USA (2009)
15. Zardetto, D., Scannapietro, M., Catarci, T.: Effective automated object matching. In: ICDE 2010. pp. 757–768 (2010)