KNOWLEDGE MEDIA INSTITUTE

Unsupervised data linking using a genetic algorithm

Technical Report kmi-11-2
September 2011

Andriy Nikolov and Mathieu d'Aquin and Enrico Motta

The Open University

Abstract. As commonly accepted identifiers for data instances in semantic datasets (such as ISBN codes or DOI identifiers) are often not available, discovering links between overlapping datasets on the Web is generally realised through the use of fuzzy similarity measures. Configuring such measures, i.e. deciding which similarity function to apply to which data properties with which parameters, 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 report, 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 training 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

Establishing links between overlapping but separately constituted dataset 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 need to be 're-conciliated' while not sharing commonly accepted identifiers (such as ISBN codes), not relying on the same schemas and ontologies (therefore using different properties to represent the same information) and often implementing different formatting conventions for attributes (e.g., using "Motta, Enrico" as the name of a person in one case, and "Enrico Motta" in the other).

When not realised manually (which is obviously not a scalable solution), data linking therefore 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. Configuring such a decision rule for a pair of datasets involves choosing the relevant properties to be compared, the similarity functions that should be used for each pair, the weight of each comparison, and the criterion determining whether a pair of individuals with a given similarity value should be linked (i.e. aggregation function and threshold).

Such choices are highly dependent on the domain of the data, the ontologies used to represent the data, as well as the specific formatting conventions used in the datasets. 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 report, we consider an approach based on a genetic algorithm to investigate the question: can a suitable decision rule for linking two datasets be learned in an unsupervised way, based only on the characteristics of the datasets and on the distribution of similarity values amongst their instances?

A genetic algorithm relies on evolving a set of initially random solutions to a problem, using selection and variation mechanisms that favor good solutions over bad ones, according to a fitness criterion. To address the question above using a genetic algorithm, we therefore rely on the assumption that such a fitness criterion can be established for a data linking decision rule without having to rely on a manually established gold standard. Following research in the area of record linkage in databases, we devise such a criterion by relying on the distribution of links and of similarity values generated by applying a particular decision rule.

To test our assumptions, we apply a genetic algorithm dedicated to establishing a data linking decision rule using an unsupervised fitness criterion as described above on the benchmark datasets from the OAEI 2010 ontology alignment contest. 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 report is structured as follows. In section 2, we provide an overview of the basic notions of the coreference resolution 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 report and discusses directions for future work.

2 Problem Definition and Related Work

In this section, we specify the tasks of coreference resolution (also called data linking) and of establishing the necessary decision rule, together with a brief description of the relevant existing work.

2.1 Coreference Resolution

The problem of coreference was originally studied in the database community where it is known as record linkage or object identification [1]. With the development of the linked data initiative, it gains importance in the Semantic Web

community where it is studied under the names of coreference resolution [2], reference reconciliation [3], and link discovery [4]. The coreference resolution task can be defined as follows.

Definition 1*: Let $\mathcal{D}_1$ and $\mathcal{D}_2$ represent two datasets, each one containing a set of individuals $\mathcal{I}_i$ and structured according to a schema $\mathcal{O}_i$. Each individual $I_{ij} \in \mathcal{I}_i$ describes some real world entity ω_j. Two individuals are said to be equivalent $I_j \equiv I_k$ if they describe the same entity $\omega_j = \omega_k$ according to a chosen identity criterion. The goal of the coreference resolution task is to discover all pairs of individuals $\{(I_{1i}, I_{2j}) | I_{1i} \in \mathcal{I}_1, I_{2j} \in \mathcal{I}_2\}$ such that $\omega_{1i} = \omega_{2j}$. In the context of linked data, datasets $\mathcal{D}_i$ are represented by RDF graphs. Individuals $I_i \in \mathcal{I}_i$ are identified by URIs and described using the classification schema and properties defined in the corresponding ontology $\mathcal{O}_i$.*

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 report. *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 [3], axioms defined in the ontological schema [5], [6], and mutual impact of different mappings [7].

The individual matching task can be defined as follows.

Definition 2*: 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 report by Fellegi & Sunter [8], 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 can take the form of comparing the confidence degree returned by the similarity function with a threshold t. A mapping (I_1, I_2) is returned if $sim(P(I_1), P(I_2)) \geq t$.

2.2 Establishing a Decision Rule for Individual Matching

As can be seen from the description above, the key component of a coreference resolution task based on individual matching is the decision rule. For a given pair of datasets to link, a decision rule has to be established that incorporate comparisons between relevant pairs of properties in the two datasets, using similarity functions appropriate to the forms of the values of these properties, as

well as weights and a thresholds to obtain an adequate discriminative ability for the decision rule.

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 coreference resolution (e.g., OKKAM [9]), generic ontology matching systems (e.g., RiMOM [10]), or systems which primarily rely on the dataset matching stage (e.g., CODI [5]).

However, in most other cases, a dedicated decision rule has to be established for each coreference resolution task (i.e., each pair of datasets to link). Existing coreference resolution systems in the Semantic Web area take two different approaches to realise this:

Manual configuration where the decision rule is specified by the user. This approach is taken by the popular SILK system [4]. 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 [2]. This alleviates the need for user input to establish the decision rule, but requires the availability of a substantial set of robust training 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 [11] individuals are clustered into matching and non-matching classes based on the structure of their neighborhood rather than on simple threshold filtering. Zardetto et al [12] 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.

The method described in this report proposes to use a genetic algorithm guided by a fitness criterion using similar characteristics to assess the expected quality of a decision rule, and of the derived set of links. Our method goes a step further as it uses distribution features to choose an appropriate similarity function for a given matching task as well as a suitable filtering criterion, rather then 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 [13]. 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 favor 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 Genetic algorithms: main concepts

Definition 3*: 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 can be used to evaluate, how close the candidate solution C_i is to the optimal one.*

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* and *mutation*:
  - *Crossover* recombines elements of several "parent" chromosomes to produce several new chromosomes (or "children")
  - *Mutation* produces a new chromosome by randomly tweaking the genes of the original one.

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

[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.

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.

3.2 Representing individual matching in terms of a genetic algorithm

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 as a comparison of an aggregated attribute similarity function with a threshold.

***Definition 4**: A **decision rule** for an individual matching task is defined as: $sim(P(I_1), P(I_2)) \geq t$ where $sim(P(I_1), P(I_2))$ is the **similarity function** comparing profiles of two individuals. This 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,*
- *t is a threshold ($0 \leq t \leq 1$)*

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. If a pair of attributes (a_{1i}, a_{2j}) is not compared, then sim_{ij} is set to nil. We included a number of character-based functions (edit distance, Jaro, Jaro-Winkler, Smith-Waterman, Monge-Elkan, and I-Sub) 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. The weighted average allows capturing "composite keys" where identity is determined based on a combination of properties, while the maximum is better suited for tasks where the profile includes several discriminative attributes (like both the social security ID and the phone number).

These genotypes are evaluated by applying the decision rule to the matching task and calculating the *pseudo-F-measure* $F^{\sim}$ metric.

3.3 Evaluating decision rules: pseudo-F-measure

The crucial component of the method is the fitness function which would allow estimating the quality of a set of mappings without possessing labeled data or involving the user. Under these conditions it is not possible to measure the quality of the results 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 following assumptions about the matching task and the distribution of similarity values returned by the function $sim(P(I_1), P(I_2))$:

1. While different URIs are often used to denote the same entity in different repositories, a URI can be expected to be a unique identifier within one dataset. If this condition holds for an instance set $\mathcal{I}$, any individual should be mapped to at most one individual in this dataset: $\forall I_i : |\{I_\mathcal{I} | (I_i, I_\mathcal{I}) \in M, I_\mathcal{I} \in \mathcal{I}\}| \leq 1$.
2. If Assumption 1 holds, then for two sets of individuals $\mathcal{I}_1$, $\mathcal{I}_2$ the maximum number of correct mappings is $|M| \leq min(|\mathcal{I}_1|, |\mathcal{I}_2|)$.
3. A meaningful similarity function produces results in the interval $\{0..1\}$ and returns values close to 1.0 for pairs of matching individuals.

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 the assumptions 1 and 2 to formulate the pseudo-precision and pseudo-recall measures in the following way:

Definition 5: *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. In accordance with Assumption 2, $n_o \leq min(|\mathcal{I}_1|, |\mathcal{I}_2|)$, 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 assume the ideal scenario where $n_o = min(|\mathcal{I}_1|, |\mathcal{I}_2|)$, but 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 $F^{\sim}_{0.1} = \frac{1.01p^{\sim}\cdot r^{\sim}}{0.01\cdot p^{\sim}+r^{\sim}}$. 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 $avg(0.05 \cdot jaro(name, label), 0.05 \cdot edit(birthYear, yearOfBirth), 0.9 \cdot edit(name, yearOfBirth)) \geq 0.098$. Although 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.

In the general case, high values of $p^{\sim}$ and $r^{\sim}$ do not strictly imply high values of p and r: e.g., we can imagine two lists of people where the social security ID of each person in the first list is equivalent to the phone number of exactly one person in the second list. Then, a function based on comparing these two attributes will result in both $p^{\sim} = 1$ and $r^{\sim} = 1$, while both $p = 0$ and $r = 0$. However, in practical tasks we can expect that (a) comparable attributes in two datasets have similar distribution of values, and (b) datasets selected for matching have a substantial degree of overlap.

3.4 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 $\mathcal{I}_2$, and the $F^{\sim}_{fit}$ fitness measure is calculated. This 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 $F^{\sim}_{fit}$ 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^m_{att}. 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^m_{wgt}. 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^m_{sym}.
- Modifying the threshold value with the probability p^m_t: 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^m_{agg}.

Only one action is selected for each mutated genotype, and $\sum p^m_i = 1$.

At the new iteration, chromosomes in the updated population are again evaluated using the $F^{\sim}_{fit}$ 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 and the assumptions on which it is based, we performed experiments with two types of datasets. First, we tested our approach on the PR benchmark datasets, which were used as a benchmark in the instance matching track of the OAEI 2010 ontology matching competition[2], to compare our method with state-of-the-art systems and study the effects of different parameter settings. Second, we used several datasets extracted from the linked data cloud to check whether the method can be reused in other real-world scenarios, and to investigate the effect of the characteristics of the datasets on the results. This section summarises the results obtained and the key points of discussion related to the use of our approach for unsupervised coreference resolution. We first summarise the default settings for these experiments.

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: $p^m_{att} = 0.3$, $p^m_{wgt} = 0.15$, $p^m_{sym} = 0.15$, $p^m_t = 0.3$, $p^m_{agg} = 0.1$.
- termination criterion: either the limit of iterations $n_{iter} = 20$ is reached or the average fitness of the population does not increase for $n_{conv} = 10$ iterations.

We implement our method on top of our KnoFuss architecture for data linking [14]. 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. We call the combined system X-KnoFuss.

4.2 Benchmark test

The benchmark contains three test cases:

- *Person1* and *Person2* test cases originally created in the FEBRL project[4]. Two pairs of datasets contain records of people, which were artificially distorted to create the test case for matching tools.

[2] http://oaei.ontologymatching.org/2010/im/index.html
[3] http://lucene.apache.org
[4] http://sourceforge.net/projects/febrl/

Table 1. Comparison of F1-measure with other tools on the benchmark datasets.

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

– *Restaurants*. The test case includes two datasets containing data about restaurants from two sources with 112 mappings between them. The benchmark is based on the test case included in the RIDDLE repository[5]. Two versions of this 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, [9], [2]). To be able to compare with systems which used both variants of the dataset, we also used both variants in our experiments.

We compared our algorithm with the systems participating in the OAEI 2010 instance matching track as well as with the FBEM system [9], whose authors provided the benchmark datasets for the competition. We report in Table 1[6] 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 $N = 1000$. As can be seen from Table 1, the solution produced 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. 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. 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 population size N, crossover rate 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 converge to the optimal solution. Figure 1 shows the average F1-measure achieved by the algorithm with different population sizes. As expected, increasing the size of the population also improves the average

[5] http://www.cs.utexas.edu/users/ml/riddle/data.html

[6] In [9] only rounded numbers for precision and recall are provided for the Restaurant dataset: 0.98 and 0.95 respectively.

Table 2. Example decision rules found by the algorithm with $N = 1000$ (weights and threshold are rounded).

Test case	Similarity function	Threshold
Person1	max(tokenized-jaro-winkler(person1:soc_sec_id;person2:soc_sec_id); monge-elkan(person1:phone_numer;person2:phone_numer))	≥0.87
Person2	max(jaro(person2:phone_numer;person1:phone_numer); jaro-winkler(person2:soc_sec_id;person:soc_sec_id))	≥0.88
Restaurants (OAEI)	avg(0.22*tokenized-smith-waterman(restaurant1:phone_number; restaurant2:phone_number); 0.78*tokenized-smith-waterman(restaurant1:name;restaurant2:name))	≥0.91
Restaurants (fixed)	avg(0.35*tokenized-monge-elkan(restaurant1:phone_number; restaurant2:phone_number); 0.65*tokenized-smith-waterman(restaurant1:name;restaurant2:name))	≥0.88

performance and robustness of the algorithm. Choosing a very small N can lead the algorithm to converge on sub-optimal solutions.

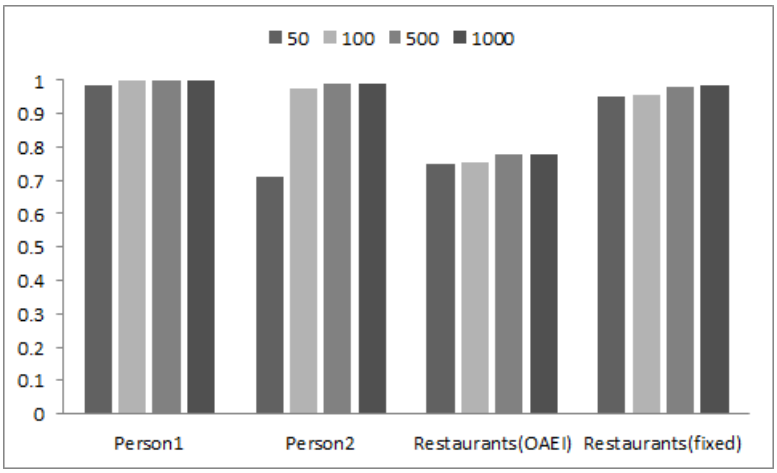

Fig. 1. Average F1-Measure achieved on OAEI datasets for different population sizes N.

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[7]. This dataset of 3995 individuals was

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

matched against the set of all people from DBPedia[8] (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[9]. 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 represented in the L3S-DBLP dataset[10] (out of the snapshot of 366113 publications included in the BTC 2010 dataset[11]). For these publications, we extracted their RDF descriptions from the DOI web-site[12]. 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. In order to test the degree to which the $F^{\sim}_{fit}$ fitness function can be used as an estimation of the actual quality of results, we performed the experiments using two different fitness functions: the unsupervised $F^{\sim}_{fit}$ 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 3, which shows average precision, recall, and F1-measure achieved using both supervised and unsupervised fitness functions. Moreover, it reports the standard deviation of F1 measure $\sigma F1$ over 5 runs and the time of a single run for the unsupervised case[13]. For the *Music contributors* test case, the results produced using $F^{\sim}_{fit}$ and $F1$ are very close. Moreover, the use of the $F^{\sim}_{fit}$ fitness function leads to higher precision than $F1$. This is a consequence of the bias for precision which is encoded in the $F^{\sim}_{fit}$ fitness function. In the case of *Book authors*, the effect of the population size is more substantial. Having $N = 50$ and $N = 100$ was not sufficient for the algorithm to achieve stable behaviour, i.e. out of several runs, the algorithm would sometimes converge on a sub-optimal solution. For $N = 500$, the algorithm has shown robust behaviour: the differ-

[8] http://dbpedia.org

[9] http://www.archive.org/details/Bibliographica.orgBnbDataset. From the first part of the dump, we selected 1000 authors with the highest number of published books, as they are more likely to be represented in DBPedia

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

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

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

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

Table 3. Results for the *Music contributors* and *Book authors* datasets.

Dataset	Pop. size N	F1-fitness (ideal case)			$F^{\sim}_{fit}$-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

ence between the best and the worst F1-measure was about 0.02, achieving high performance.

It can be noticed however, that in the case of *Book authors*, the performance obtained with the unsupervised $F^{\sim}_{fit}$ fitness function was lower than the one obtained with the F1-Measure. In this case, the $F^{\sim}_{fit}$ fitness function could not discriminate between several similar decision rules with the same characteristics of the output: all of them compared person's name (*foaf:name* and *rdfs:label* in BNB and DBPedia respectively) and the date of birth (property chain {*bio:event*, *bio:date*} and the property *dbpedia:birthDate*). These alternative decision rules did not violate the 1-to-1 mapping restriction and returned sets of results of similar sizes. However, the proportion of actually correct results in these sets varied. To overcome such issues, the use of domain knowledge can be beneficial: for instance, evaluating mappings with respect to domain-specific constraints can potentially increase the sensitivity of the fitness function and filter out sub-optimal solutions.

Table 4. 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

For the *Research papers* dataset (Table 4), 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 4 we can see that the performance of the unsupervised fitness function of $F^{\sim}_{fit}$ is slightly lower. The main reason for this is the preference for precision at the expense of recall, which is encoded in the fitness function. As a result, precision of the solutions produced by the unsupervised algorithm was always higher than the precision of the solutions optimal with respect to the F1-measure.

We can also see that very small sample sizes (50) lead to unstable performance in the same way as small population sizes. However, starting from 100 individuals the algorithm achieved stable performance. 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.

The datasets and test results are available for download from our website[14].

5 Conclusion and future work

In this report, 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 as a pseudo-F-measure metric and used it as 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 report to pursue several promising research directions. One of these directions involves 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.

The second potential direction involves combining individual matching with schema alignment. In order to minimise the number of incorrect links between individuals, it is important to select the datasets for matching appropriately: for example, in our *Music contributors* test case, matching instances of the class *linkedmdb:music_contributor* with the specific class *dbpedia:Artist* instead of generic *dbpedia:Person* would avoid producing false positive mappings. The problem of selecting appropriate subsets of data for matching was a topic of our previous work [15], where we proposed the use of a semantic index for this task. We plan to investigate to which extent our unsupervised method can benefit from utilizing schema mappings, and, on the other hand, whether distributions of results produced by unsupervised individual matching can be used to produce mappings between schema terms: classes and properties.

6 Acknowledgements

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

[14] http://kmi.open.ac.uk/technologies/knofuss/knofuss-GA-tests.zip

References

1. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering **19**(1) (2007) 1–16
2. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: Proceedings WWW 2011. (2011) 87–96
3. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. (2005)
4. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the Web of Data. In: Proceedings ISWC 2009, Washington, DC, USA (2009) 650–665
5. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging terminological structure for object reconciliation. In: Proceedings ESWC 2010, Heraklion, Crete, Greece 334–348
6. Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. Journal of Data Semantics **12** (2008)
7. Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., de Meer, H.: idMesh: Graph-based disambiguation of linked data. In: Proceedings WWW 2009, Madrid, Spain, ACM (2009) 591–600
8. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of American Statistical Association **64**(328) (1969) 1183–1210
9. Stoermer, H., Rassadko, N., Vaidya, N.: Feature-based entity matching: The FBEM model, implementation, evaluation. In: Proceedings CAISE 2010. (2010) 180–193
10. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering **21**(8) (2009) 1218–1232
11. Chaudhuri, S., Ganti, V., Motwani, R.: Robust identification of fuzzy duplicates. In: Proceedings ICDE 2005. (2005) 865–876
12. Zardetto, D., Scannapietro, M., Catarci, T.: Effective automated object matching. In: Proceedings ICDE 2010. (2010) 757–768
13. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)
14. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: Integration of semantically annotated data by the KnoFuss architecture. In: Proceedings EKAW 2008. (2008)
15. Nikolov, A., d'Aquin, M.: Identifying relevant sources for data linking using a Semantic Web index. In: Workshop on Linked Data on the Web (LDOW 2011), WWW 2011, Hyderabad, India (2011)