TU Delft

Delft University of Technology

Simulated Annealing-based Ontology Matching

Mohammadi, Majid; Hofman, Wout; Tan, Yao Hua

DOI
10.1145/3314948

Publication date
2019

Document Version
Final published version

Published in
ACM Transactions on Management Information Systems

Citation (APA)
Mohammadi, M., Hofman, W., & Tan, Y. H. (2019). Simulated Annealing-based Ontology Matching. *ACM Transactions on Management Information Systems*, *10*(1), Article 3. https://doi.org/10.1145/3314948

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Simulated Annealing-based Ontology Matching

MAJID MOHAMMADI, Delft University of Technology
WOUT HOFMAN, The Netherlands Institute of Applied Technology (TNO)
YAO-HUA TAN, Delft University of Technology

Ontology alignment is a fundamental task to reconcile the heterogeneity among various information systems using distinct information sources. The evolutionary algorithms (EAs) have been already considered as the primary strategy to develop an ontology alignment system. However, such systems have two significant drawbacks: they either need a ground truth that is often unavailable, or they utilize the population-based EAs in a way that they require massive computation and memory. This article presents a new ontology alignment system, called SANOM, which uses the well-known simulated annealing as the principal technique to find the mappings between two given ontologies while no ground truth is available. In contrast to population-based EAs, the simulated annealing need not generate populations, which makes it significantly swift and memory-efficient for the ontology alignment problem. This article models the ontology alignment problem as optimizing the fitness of a state whose optimum is obtained by using the simulated annealing. A complex fitness function is developed that takes advantage of various similarity metrics including string, linguistic, and structural similarities. A randomized warm initialization is specially tailored for the simulated annealing to expedite its convergence. The experiments illustrate that SANOM is competitive with the state-of-the-art and is significantly superior to other EA-based systems.

CCS Concepts: • **Information systems → Data cleaning**; **Mediators and data integration**; • **Theory of computation → Data integration**;

Additional Key Words and Phrases: Ontology alignment, simulated annealing, SANOM, OAEI

ACM Reference format:
Majid Mohammadi, Wout Hofman, and Yao-Hua Tan. 2019. Simulated Annealing-based Ontology Matching. *ACM Trans. Manage. Inf. Syst.* 10, 1, Article 3 (May 2019), 24 pages.
https://doi.org/10.1145/3314948

1 INTRODUCTION

Ontologies are the tools to formalize the objects and their corresponding relations in a domain. Due to their extraordinary power of expressiveness, ontologies have been used in diverse fields to model the underlying concepts in a formal manner (Baader et al. 2006; Bandrowski et al. 2016; Hoehndorf et al. 2015; Reitsma et al. 2009).

The ontology-based modeling is subjective and expert-dependent, hence the similar concepts in one particular domain can be constructed in entirely distinct ways. The discrepancy in models is referred to as heterogeneity, and it is seemingly inevitable despite the fact that data are coming from various sources in the era of information explosion.

Authors' addresses: M. Mohammadi, W. Hofman, and Y.-H. Tan, Jaffalaan 5, 2628BX, Delft, The Netherlands; emails: m.mohammadi@tudelft.nl, wout.hofman@tno.nl, y.tan@tudelft.nl.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
2158-656X/2019/05-ART3 $15.00
https://doi.org/10.1145/3314948

RIGHTSLINK

The heterogeneity among various data sources is a major impediment to the path of interoperability. This difference among information systems calls for the need to design an automatic solution to make them interact. Ontology matching, or alignment, is one approach to make the heterogeneous information systems interoperable by finding the semantically identical concepts of two ontologies that are stated in distinct ways. The ontology alignment systems usually take advantage of multiple similarity measures to find similar entities. However, the way to decide among various similarity measures is a fundamental issue to attack.

1.1 Related Works

The evolutionary algorithms (EAs) have long been used for ontology alignment. There are two different ways to apply EAs to the ontology alignment problem. The first approach is the so-called *meta-matching*, whose goal is to find heuristically the hyper-parameters of an alignment system. Generally, a set of similarity measures for each pair of entities is selected, and the goal is to achieve the optimal weights for the chosen similarity metrics. Another critical parameter usually computed by meta-matching techniques is the threshold according to which the final alignment will be obtained. The major shortcoming of the meta-matching is that they often need a reference alignment, or a part of it, to identify the hyper-parameters. In reality, however, the ground truth of given ontologies is often unavailable, and the applicability of such systems is thus restricted. Such a drawback is present in most of the meta-matching systems using EAs (Acampora et al. 2014; Martínez-Romero et al. 2013; Xue and Liu 2017).

To our knowledge, there is only one meta-matching system that is able to discover alignments of two given ontologies without the ground truth (Xue and Wang 2015). In their proposed system, X. Xue et al. have used two heuristic measures that are not reliant on the ground truth. The measures are *MatchFmeasure* and *Unanimous Improvement Ratio (UIR),* based on which memetic algorithm is applied to identify the alignment.

The second way of using EAs is to solve the ontology alignment problem directly. Similar to the meta-matching techniques, there are multiple systems that require a reference alignment. These systems optimize various objective functions such as F-measure (Gil et al. 2008; Xue et al. 2015a) and a weighted sum of similarity metrics (Xue et al. 2015b). Such systems also have narrow applicability in real-world situations, since no gold standard is available in reality.

In addition, there are several EA-based systems suitable for real-world situations. J. Wang et al. are arguably the first ones who used an evolutionary algorithm, i.e., genetic, to find the alignment between two given ontologies (Wang et al. 2006). Their proposed system, GAOM,[1] models a possible alignment as a population member (chromosome). They further define the intension of a concept as a set containing its name, properties, and instances, and the extension of a concept as its relation (i.e., object property) to some other entities at the same ontology. Based on the intensional and extensional features, the fitness of a chromosome is computed, and the optimal alignment is discovered using the genetic algorithm.

GAOM suffers from several drawbacks. First, it solely matches the classes, not the object or data properties, although they are used to measure the similarity of classes. On top of that, it is not clear how the structural similarity of concepts is considered.

A well-developed system, called MapPSO (Bock and Hettenhausen 2012), identifies the alignment based on the discrete particle swarm optimization (PSO). MapPSO is able to align classes and properties of two given ontologies without the requirement of a reference alignment. This system utilizes lexical, linguistic, and structural similarity metrics to determine the fitness of a particle.

[1]Stands for **G**enetic **A**lgorithm-based **O**ntology **M**atching.

RIGHTSLINK

Aside from its salient characteristics, MapPSO has several severe drawbacks as well. First, there is no pre-processing (e.g., tokenization and stemming) over the names of various entities. In their alignment algorithm, the Levenshtein string similarity (Yujian and Bo 2007) is directly applied to the names of two entities to gauge their similarity. This approach has low applicability in real-world ontologies, since the concepts are likely to be named as the combination of various tokens. As a result, the similarity computation of names merely based on the Levenshtein metric would lead to overall poor performance, since the recent studies have accented the role of string similarity metrics for ontology matching (Cheatham and Hitzler 2013). Such names cannot be discovered in WordNet (Miller 1995), neither would the linguistic similarity used in MapPSO lead to a significant mapping discovery when the linguistic heterogeneity is present.

Yet another subtle but essential drawback of MapPSO is that the same string similarity metric has been used for matching properties. Nonetheless, the sole consideration of the names of properties would increase simultaneously the false negative and false positive (Cheatham and Hitzler 2014).

There are also several pitfalls inherited from PSO. PSO is a population-based evolutionary algorithm, and MapPSO used it in such a way that it needs to generate a significant number of particles to transition to the next generation and to find the optimum of the given problem. Such populations need to be stored in the main memory so it requires a considerable amount of space. The computation of population fitness would also be time-consuming. Further, PSO is suffering from the so-called premature convergence, hence it is likely that it converges to the local optima. There are also other systems based on the population-based incremental learning (Xue and Chen 2018) and the non-dominated sorting genetic algorithm-II (NGSA-II) (Xue and Wang 2017), which solve the ontology alignment problem as well.

1.2 Contributions

In this article, the simulated annealing (SA) is used as the primary strategy to find the alignment between two given ontologies. SA has several salient features that make it practically more efficient than other evolutionary techniques. SA mimics the slow cooling in metallurgy in a way that it slowly decreases a temperature value that is high at the beginning of the process. When the temperature is high, it is more likely that the transition to a worse state (based on the fitness function) happens. As the temperature decreases, the odds of moving to a worse solution diminishes as well. Accepting the worst solution at the beginning would help explore the whole solution space so the chance of the premature convergence significantly falls. Along with its convergence, SA is more time- and memory-efficient than the population-based EAs, since it only operates on one single state based on it producing a *successor*. Therefore, it requires less memory to store the populations and less time for computing the fitness of multiple chromosomes in a population.

Aside from the inherent characteristics of SA, there are several other advantages of the system proposed in this article, SANOM.[2] Contrary to MapPSO, SANOM performs a complete pre-processing step that is proved to enhance significantly the performance of matching (Cheatham and Hitzler 2013). Further, it benefits from the so-called Soft TF-IDF (term-frequency and inverse document frequency) string metric (Cohen et al. 2003) and generalizes it with two base similarity metrics. One of the string similarity metrics is Jaro-Winkler, which only compares the names, and the second is a WordNet-based metric to gauge the linguistic proximity of tokens. The proposed Soft TF-IDF is able to detect the correspondences whose parts of names have been stated by different but synonymous tokens.

[2]Stands for **S**imulated **AN**nealing-based **O**ntology **M**atching.

RIGHTSLINK

For matching properties, SANOM will use the notion of the *core* concept, defined in Cheatham and Hitzler (Cheatham and Hitzler 2014) as an extra name for the given properties. This would increase the likelihood of mapping while the false positive decreases as well.

Among EA-based ontology alignment systems, MapPSO is the only one that participated in the OAEI, and its implementation is also freely available.[3] Thus, we compare in particular SANOM with MapPSO in terms of the execution time as well as efficiency gauged by various performance metrics.

The contributions of this article can be summarized as follows:

- An alignment is modeled as a state whose optimum based on a fitness function will solve the ontology matching problem;
- An intrinsic fitness function is developed by using various similarity measures. In this regard, the Soft TF-IDF metric is extended by using two base similarity metrics: one for the strings similarity of tokens and one for their linguistic relations;
- The simulated annealing is adjusted to find the alignment between two given ontologies. In this regard, a randomized greedy algorithm is developed for the initialization that expedites the convergence of the algorithm;
- The proposed system is evaluated by the OAEI anatomy, conference, and disease and phenotype tracks. An extensive statistical comparison is conducted as well to display the advantages and pitfalls of the proposed system.

The preliminary implementation of SANOM participated in the OAEI 2017 (Mohammadi et al. 2017a), and the current, enhanced implementation participates in the OAEI 2017.5 and 2018.

1.3 Structure

This article is structured as follows: Section 2 dedicates to the preliminary concepts of ontology alignment and simulated annealing required for the further sections. The computation of the alignment fitness using string and structural similarity metrics is discussed in Section 3. Section 4 contains the details of the proposed system, SANOM, and the experimental results are presented in Section 5. The article concludes in Section 6.

2 PRELIMINARIES

In this section, the fundamental concepts regarding the ontology and ontology alignment are first revised, and it follows by the explanation of the simulated annealing method.

2.1 Ontologies and Ontology Matching

Ontologies are keys to formally modeling a domain of interest. One of the salient advantages of doing so is the interoperability among various systems in one particular domain.

In this article, the ontologies can be simply considered as a 4-tuple of their different entities.

Definition 2.1 (Euzenat et al. 2007). An ontology O could be defined as

$$O = (C, DP, OP, I),$$

where

- C is the set of classes that are the principal concepts in a domain;
- DP is the set of data properties explaining the characteristics of the classes;
- OP is the set of object properties defining the relation of two classes;
- I is the set of individuals that instantiate the modeled concepts.

[3] https://sourceforge.net/projects/mappso/.

RIGHTSLINK

The ontologies are designed by humans and are not homogeneous by nature. This would prevent the systems from sharing data with each other, as their modeling would not concur. Ontology alignment would be a solution to enable the interoperability between two or multiple systems having different models. Several rudimentary concepts are first presented that are required for ontology alignment.

Definition 2.2 (Correspondence (Euzenat et al. 2007)). A correspondence between two given ontologies O and O' is defined as a set of 4-tuples

$$< e, e', r, c >,$$

where

- e is an entity, e.g., class, property, or instance, from the first ontology;
- e' is an entity form the second ontology;
- r is the type of relation between two entities, e.g., equivalence, subsumption;
- $c \in [0, 1]$ is the confidence of the matching.

Definition 2.3 (Alignment (Euzenat et al. 2007)). An alignment is the typical outcome of the ontology matching systems and consists of several correspondences between different entities of two given ontologies.

The goal of this article is to take advantage of several similarity measures and to propose a methodology based on the well-known simulated annealing to discover the alignment between two ontologies.

2.2 Simulated Annealing

Simulated annealing is a probabilistic approach to estimate the global optimum of problems that cannot be solved by the standard optimization techniques (Metropolis et al. 1953). As the name suggests, this technique simulates the annealing in metallurgy that slowly cools the materials to decrease their defects.

Such a controlled cooling is implemented in the simulated annealing as the probability to transition to a worse solution. The probability of a move to a worse solution is proportionate to the temperature: The higher the temperature, the more chance to move to a worse solution. Such a feature would enable SA to explore the whole search space, hence it does not converge prematurely, unlike the genetic and swarm intelligence algorithms.

In contrast to population-based EAs, SA only operates on one possible solution, called *state*, and tries to improve it to get a better solution. Such an enhancement is performed by creating a new successor in the neighborhood of the current state and then probabilistically transitioning to it. Let S be the current state and S' be the *successor* (or the neighbor) created based on the current state. The proposed move from S to S' happens based on a fitness function: If the fitness of S' is superior to S, then the transition *certainly* happens, and it *probably* occurs otherwise.

The probability of a move when the fitness of the successor is less than the current state is commensurate with the value of their fitness and the temperature. In more detail, let $f(S)$ and $f(S')$ be the fitness of the current and successor states, respectively. If the $f(S') > f(S)$, then transition to *successor* happens. Otherwise, let $\Delta E = f(S') - f(S)$, then the probability to move to the proposed successor P_{move} can be stated as

$$P_{move} = min\left(e^{\frac{\Delta E}{T}}, 1\right), \tag{1}$$

where T is the temperature.

RIGHTSLINK

It is evident from Equation (1) that if $f(S') > f(S)$, then $e^{\Delta E/T} > 1$ and $P_{move} = 1$. Thus, the proposed move to S' will certainly happen. Otherwise, the transition is reliant on ΔE and T: The greater ΔE or smaller T, the smaller chance to accept the move to a state with lower fitness.

Having generated the successor and having then computed the probability of accepting the move using Equation (1), the move is accepted or rejected in practice by sampling from a uniform distribution in the interval $[0, 1]$. If the sampled value is less than P_{move}, then the move to S' is accepted. Otherwise, the transition is rejected, and a new successor is produced based on the current state.

The temperature also plays a critical role in the transition to a worse state. The simulated annealing algorithm starts with a higher temperature so the transition to worse states are more likely at the beginning. However, the temperature becomes lower as time goes by, hence it is unlikely to move to a worse solution. This enables SA to explore the whole solution space at the beginning to find the global optima and to prevent premature convergence. The overall SA algorithm is summarized in Algorithm 1.

ALGORITHM 1: Simulated annealing

Input: $S = S_0$, maxIter
for iter=1:maxIter **do**
  $T = updateTemperature(iter, maxIter)$
  $S' = generateSuccessor(S)$
  Compute $P_{move} = Pr(S, S', T')$ by Equation (1)
  Sample r from the uniform distribution in the interval $[0, 1]$
  if $P_{move} > r$ **then**
    $S = S'$
  end if
end for
Output Final state S

3 ALIGNMENT FITNESS

At the heart of any evolutionary algorithm, there must be a way to measure the fitness of different solutions based on which the evolution happens. First, a precise definition is presented for the alignment fitness:

Definition 3.1 (Alignment Fitness). Given an alignment A between two ontologies O_1 and O_2, the fitness of A is computed by the function F: $A^* \rightarrow R$ (where A^* is the set of all possible alignments), and is defined as

$$F(A) = \sum_{c \in A} f(c),$$

where $f : A \rightarrow R$ computes the similarity of each correspondence in the given alignment A.

The alignment fitness definition reveals the need for computing the fitness of each correspondence for having the overall fitness of a given alignment. The function f calculates the similarity of two entities in the correspondence c.

The first way of the similarity calculation is to consider the names (e.g., URI, label, comments, etc.) of two given entities and to determine their sameness using either string similarity metrics or their linguistic relations using WordNet (Miller 1995). The way of finding the similarity between two classes and two properties are different in the proposed system.

RIGHTSLINK

It is also possible to consider the positions of two entities in their ontologies as a meter of similarity. For instance, if two classes match from two ontologies, then the likelihood of mapping their subclasses increases. Such metrics are referred to as the structural similarity measures. In a nutshell, $f(c) = f_{string}(c) + f_{strucutural}(c)$, where $f_{string}(c)$ and $f_{strucutural}(c)$ are the string and structural similarity measures, respectively.

In the remainder of this section, the appropriate similarity metrics that are utilized by SANOM are reviewed.

3.1 String Similarity Metric

In this section, the techniques for computing the similarity between the strings of two entities are revised. We take advantage of the Soft TF-IDF (term frequency-inverse document frequency) with two base similarity metrics. The reason for using this metric is that it can be generalized to accommodate multiple base similarity metrics. There are some other metrics such as Soft Jaccard that have the same capability, but Soft TF-IDF has shown better performance in terms of both precision and recall in recent studies (Cheatham and Hitzler 2013). The base metrics for Soft TF-IDF are Jaro-Winkler to deal with names as a sequence of characters, and the other one is Wu and Palmer (Wu and Palmer 1994), which computes the linguistic relatedness of two names using WordNet.

It is common that ontologies have several annotations that facilitate finding their peers in other ontologies. Further, the sole comparison of properties names would lead to poor results. Therefore, we consider a set of names for each entity as follows:

- For classes, the essential name is their uniform resource identifier (URI). Besides, there are some annotations that might help the process of matching. Among them are *label* and *comment* annotations that provide more information about the corresponding class. There are sometimes related synonyms for classes in the ontology that should also be considered. The OAEI anatomy track has several related synonyms that would enhance the alignment outcome.
- For the property alignment, considering solely the names would mislead the matching process. As recommended by Cheatham and Hitzler (2014), we consider the *core* concept as another name for each property. The core is the first verb, if it exists, in the property name whose length is higher than three, or otherwise the first noun along with its corresponding adjective. The Stanford part of speech tagger is utilized to extract the core of each property (Toutanova et al. 2003).

Evidently, it is likely that each concept has more than one name; therefore, we take the maximum similarity among various names as the string similarity among two corresponding entities. Let $S \in O$ and $T \in O'$ be a set of names pertaining to two entities e_1 and e_2, then

$$f_{string}(c) = Sim(S, T) = \max_{s \in S, t \in T} \text{Soft TF-IDF(s,t)}, \tag{2}$$

where c represents a correspondence containing the mapping e_1 to e_2, $Sim(S, T)$ is the similarity between the names of two entities S and T, and Soft TF-IDF denotes the string similarity measure. In further subsections, the Soft TF-IDF is explained. Prior to that, we need to use several pre-processing strategies to increase the chance of matching.

3.1.1 Pre-processing Strategies. The modification of strings before the similarity computation is essential to increase the chance of mapping entities. The primary pre-processing strategies utilized by SANOM are:

RIGHTSLINK

- **Tokenization.** The terminology of concepts is usually constructed from a sequence of words. The words are often concatenated by white space, the capital letter of first letters, and several punctuations such as "–" or "_". Therefore, the initial strings can be broken into a bag of words that is called *tokenization*.
- **Stop word removal.** Stop words refer to the common words that do not convey any particular meaning. The stop words can be distinguished by looking up the tokens (identified after tokenization) in a table storing the potential stop words. The Glasgow stop word list is utilized in the current implementation.[4]
- **Stemming.** The entities may refer to the same concept, but they appear differently due to various verb tense, plural/singular, and so forth. Therefore, we need to revert them to the normal state to be able to detect the similar concepts that have been altered for grammatical reasons. The Porter stemming method is used for this matter (Porter 1980).

3.1.2 Soft TF-IDF with Multiple Base Similarity Metrics. TF-IDF, or cosine similarity, is one of the most popular strategies in information retrieval (Cohen et al. 2003). To calculate TF-IDF, we need to compute the frequency of the word w in the bag of tokens S (e.g., $TF_{w,S}$) and the inverse fraction of strings that contain w (e.g., IDF_w). Then, TF-IDF of two given strings S and T are computed as

$$\text{TF-IDF}(S,T) = \sum_{w \in S \cap T} V(w,s)V(w,T), \tag{3}$$

where V is defined as

$$V(w,S) = \frac{log(TF_{w,S}+1).log(IDF_w)}{\sum_{w'} log(TF_{w',S}+1).log(IDF'_w)}. \tag{4}$$

Equation (3) only considers the words that are exactly *the same* in both bags of words. However, *the sameness* can be interpreted differently, especially for ontology alignment. Thus, TF-IDF can be extended by defining the sameness using a base string similarity measure. Given such a similarity metric and a threshold, the set C is defined as the set of triples (s,t,sim), where $s \in S$ and $t \in T$ are tokens whose similarity sim is computed by the base similarity metric (and is, of course, greater than a given threshold). Having the set C, the Soft TF-IDF is defined as

$$\text{Soft TF-IDF}(S,T) = \sum_{w \in C} V(w,s)V(w,T)D(w,T), \tag{5}$$

where $D(w,T) = max_{v \in C} sim(w,v)$, and $sim(w,v)$ is the similarity of w and v in the set C.

The base similarity metric gauges the similarity of tokens obtained from each name. In this study, we take advantage of two similarity metrics and take their maximum as the final similarity of two given tokens. The reason of considering the maximum similarity is that two tokens are assumed to be identical if their names are the same or they are linguistically related. For instance, *ConferenceDinner* and *ConferenceBanquet* are deemed the same, since the first token of two names is identical, and the second token is linguistically similar. The similarity metrics for measuring the strings similarity and lingual relatedness are:

- **Jaro-Winkler metric.** The combination of TF-IDF and Jaro-Winkler has shown promising performance in the name entity matching (Cohen et al. 2003), and also in ontology alignment (Cheatham and Hitzler 2013). By the same token, SANOM exploits Jaro-Winkler with the threshold 0.9 as one of the base similarity metrics. The value of the threshold is in line with recent studies (Cheatham and Hitzler 2013).

[4] http://ir.dcs.gla.ac.uk/resources/linguistic utils/stop words.

RIGHTSLINK

- **WordNet-based metric.** The linguistic heterogeneity is also rampant in various domains. Therefore, the existence of a similarity metric to measure the lingual closeness of two entities is absolutely essential. In this study, the relatedness of two given tokens is computed by the Wu and Palmer measure (Wu and Palmer 1994) and is used as a base similarity metric. The threshold should be high enough, since two distinct tokens might seem alike, since Wu and Palmer check the semantic relatedness of given tokens. Our investigation showed that any value less than 0.8 would practically state that most of the given tokens are claimed to be the same according to Wu and Palmer. Thus, one needs to specify a much higher threshold to avoid it. In the current implementation of SANOM, the threshold for this similarity metric is set to 0.95.

Using these two base similarity metrics, we can discover the concepts with synonymous changes in one or multiple tokens.

3.2 Structural Similarity

The preceding string similarity metric gives a high score to the entities that have lexical or linguistic proximity. Another similarity of two entities could be derived from their positions in the given ontologies.

We consider two structural similarity measures for the current implementation of SANOM:

- The first structural similarity is gauged by the subsumption relation of classes. If there are two classes c_1 and c_2 whose superclasses are s_1 and s_2 from two given ontologies O_1 and O_2, then the matching of classes s_1 and s_2 would increase the similarity of c_1 and c_2. Let s be a correspondence mapping s_1 to s_2, then the increased similarity of c_1 and c_2 is gauged by

$$f_{structural}(c_1, c_2) = f(s). \tag{6}$$

- Another structural similarity is derived from the properties of the given ontologies. The alignment of two properties would tell us that their corresponding domain and/or ranges are also identical. Similarly, if two properties have the analogous domain and/or range, then it is likely they are similar as well.

  The names of properties and even their corresponding core concepts are not a reliable meter based on which they are declared a correspondence. A recent study has shown that the mapping of properties solely based on their names would result in high false positive and false negative rates; e.g., there are properties with identical names that are not semantically related while there are semantically relevant properties with totally distinct names.

  The current implementation treats the object and data properties differently. For the object properties op_1 and op_2, their corresponding domains and ranges are computed as the concatenation of their set of ranges and domains, respectively. Then, the fitness of the names, domains, and ranges are computed by the Soft TF-IDF. The final mapping of two properties is the average of the top two fitness scores obtained by the Soft TF-IDF. For the data properties, the fitness is computed as the similarity average of names and their corresponding domain.

  On the other flow of alignment, it is possible to derive if two classes are identical based on the properties. Let e_1 and e_2 be classes, op_1 and op_2 be the object properties, and R_1 and R_2 are the corresponding ranges, then the correspondence $c = (e_1, e_2)$ is evaluated as

$$f_{structural}(c) = \frac{f_{string}(R_1, R_2) + f_{string}(op_1, op_2)}{2}. \tag{7}$$

RIGHTSLINK

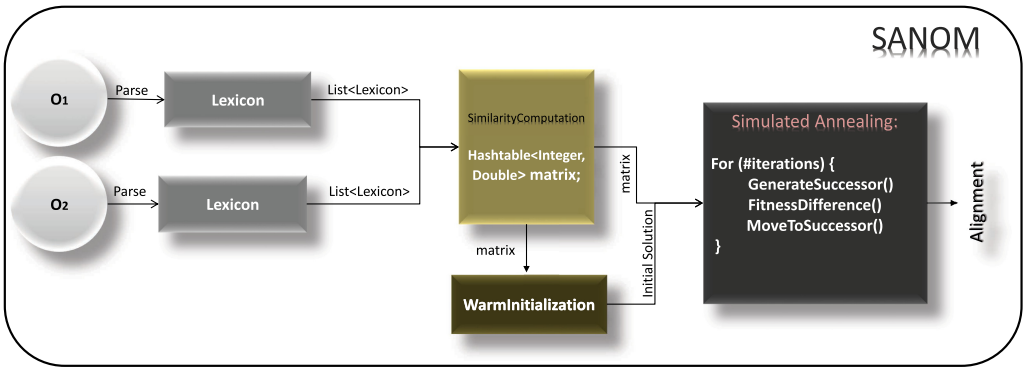

Fig. 1. The architecture of SANOM.

4 ONTOLOGY ALIGNMENT USING SIMULATED ANNEALING

Having computed the fitness function, the simulated annealing can be exploited to find the best possible alignment of two given ontologies. In this section, the necessary steps of the simulated annealing are described to solve the ontology alignment problem. Figure 1 displays the details of the SANOM implementation. In the following, we explicate different modules of SANOM depicted in Figure 1.

4.1 Ontology Parsing and Similarity Computation

According to Figure 1, ontologies are first parsed using OWL API (Horridge and Bechhofer 2011), and they are stored in a list of a data structure called *lexicon*. Lexicon is the main data structure of SANOM, which contains a list of hash sets for each concept in the ontology. This list contains all the names of a concept (e.g., URI and labels) that are tokenized and undergone the pre-processing strategies before being saved. Thus, a hash set in the list contains the tokens of a name of a concept form ontology. The overall concepts of each ontology are stored in a list of lexicon. The list index is considered as the index of the corresponding concept in the ontology. The index will be used to store the similarity of concepts.

After parsing the ontologies, we first need to compute the similarity of each concept from the first ontology to all concepts from the target. As is recently recommended (Faria et al. 2018), the similarity computations should efficiently be stored in a hash table. Further, we only store the similarities whose magnitude is bigger than a value. In the current implementation, this value is set to 0.5 (Faria et al. 2018). The keys of the hash table can be simply generated based on the index of the concept of the first ontology to that in the target. Using hash tables decreases significantly the amount of memory required for saving the similarities for large-scale ontologies.

Since the computation of similarity metrics is a nested loop, it is a time-consuming procedure. Thus, we used the Java Fork/Join framework (Lea 2000) to expedite this procedure. The Java Fork/Join framework uses the divide and conquer strategy so it divides the initial big tasks into several small ones (the fork step) and then solve the smaller tasks. At the end, the small tasks are aggregated together (the join step). For computing the similarity of a concept from the first ontology, we divide the concepts of the second ontology into different disjoint partitions, and compute the similarities of the concept of the first ontology with those concepts in the smaller tasks.

Having computed the similarities, we now look into other elements in SANOM. Prior to that, we need to model an alignment to be able to use the SA for ontology alignment.

RIGHTSLINK

4.2 Representation of an Alignment

The state in the simulated annealing would represent an alignment. Therefore, the optimal mappings between the given ontologies are obtained by optimizing the state fitness.

Let n and m be the number of concepts of two given ontologies, the state $S \in R^n$ is an integer vector whose values are between 1 and m, e.g, $1 \leq S_i \leq m$. Hence, if the i^{th} cell contains the number j, it indicates that (e_i, e_j) is the related correspondence.

Definition 4.1 (Alignment State). Any arbitrary state S during the simulated annealing represents an alignment between two given ontologies, and can be seen as a set containing pairs of concepts from two ontologies:

$$S = \{(e_1, e_{j_1}), (e_2, e_{j_2}), \ldots, (e_n, e_{j_n})\} \tag{8}$$

where (e_i, e_{j_i}) is a correspondence mapping e_i in the first ontology to e_{j_i} in the target, and $j_i \in \{1, 2, \ldots, m\}$.

Definition 4.1 formally describes the state in the simulated annealing as an alignment. From the implementation view, the set S can be defined as a vector whose entries would indicate a correspondence. More in detail, the element at the k^{th} position of this vector is the mapping (e_k, e_{j_k}).

In contrary to the population-based evolutionary strategies, the simulated annealing only operates on one single state and tries to evolve it to obtain a better solution. Therefore, it is more time- and memory-efficient. The length of S could be more optimal if we choose it as the minimum of m and n, especially if their difference is high. However, such an improvement is not significant and is ignored in the current implementation.

4.3 Warm Initialization with a Randomized Greedy Technique

To expedite the convergence of the simulated annealing, we use a randomized greedy technique for the initialization. An element from the alignment state is arbitrarily chosen by finding a random number r between 1 and n. Then, the entity e_r is mapped to entity e_{j_r}, which has the maximum similarity, e.g., $arg\max_{e \in O_2} f(e, e_r) = e_{j_r}$. The similarity of the correspondences $f(e, e_{j_r})$ is stored in a hash table that can be retrieved immediately so finding an initial solution is not time-consuming. It is evident that this way of mapping does not result in optimal alignment, but it is significantly better than using an arbitrary initial state.

The mapping is considered to be one-to-one, hence it must be fulfilled in the initialized state as well. Therefore, some auxiliary variables are required to check these constraints. We also need to compute the fitness of the initial solution, since it is required in the SA. The fitness can be simply calculated by adding the fitness of each correspondence we added to the alignment. Algorithm 2 summarizes the whole procedure of finding an efficient initial state.

4.4 Generating a Successor

The simulated annealing finds the optimal solution by the transition to a new state that usually has a higher fitness value. The prerequisite to such a move is to first generate the next state in the neighborhood of the current.

We swap the elements of the current state to produce a successor. The number of elements to be swapped can be a fraction of the state length. In the current implementation, we alter q elements of the current state where $q = \lceil 5\% * |S| \rceil$ and $|S|$ is the length of the current state. The alteration is by finding q distinct number between 1 and n, called k, and then exchanging the elements $s(k(i))$ and $s(k(i+1))$ where i is an index.

The fitness of the successor can also be computed while it is created based on the current state. When the values at the $k(i)$ position are replaced with the value of $k(i+1)$, it means the

RIGHTSLINK

ALGORITHM 2: Randomized greedy technique for initialization

Input: Set of entities of the source and target ontologies E_1 and E_2
$n = |E_1|$, $m = |E_2|$, $counter = 0$, $fit = 0$, S
while counter < n **do**
    r = $generate - random - number(1, n)$
    If(Chosen-Before(r)) **continue;**
    $e_{j_r} = arg\max_{e \in O_2} f(e, e_r)$
    fit += $f(e_r, e_{j_r})$
    S(r) = e_{j_r}
    Remove(e_{j_r}, E_2)
    ++counter;
end while
Output State S and its fitness fit

correspondence $(e_{k(i)}, e_{S(k(i))})$ is replaced with $(e_{k(i)}, e_{S(k(i+1))})$, and the correspondence $(e_{k(i+1)}, e_{S(k(i+1))})$ is substituted with $(e_{k(i+1)}, e_{S(k(i))})$. As a result, we need merely to subtract the fitness values of the previous correspondences and add those of the new ones. The fitness of these correspondences have already been stored in a hash table. Algorithm 4 summarizes the overall procedure for creating a successor and its fitness. Since the fitness of correspondences is computed in a hash table before running the SA, the creation of a successor and its fitness is quite swift. It only requires to swap $k/2$ elements and conduct $4k/2$ additions/subtractions.

ALGORITHM 3: Generating a successor and its fitness calculation

Input: State S and its fitness f(S)
$n = |S|$, $m = |E_2|$, $S' = S$, $f(S')$ = f(S)
$q = \lceil 5\%n \rceil$
k=generate-distinct-number(q,1,n); // generating q distinct number in the interval [1,n]
for i < length(k); k+2 **do**
    swap(S',k(i),k(i+1)); // replacing the elements of S in the positions $k(i)$ and $k(i+1)$
    $f(S') -= f(e_{k(i)}, e_{S(k(i))})$.
    $f(S') -= f(e_{k(i+1)}, e_{S(k(i+1))})$
    $f(S') += f(e_{k(i)}, e_{S(k(i+1))})$
    $f(S') += f(e_{k(i+1)}, e_{S(k(i))})$
end for
Output State S' and its fitness $f(S')$

4.5 SANOM in a Nutshell

SANOM first computes the similarity of each entity from the first ontology to the entities from the target. Then, the warm initialization would find a possibly good initial alignment state for the given ontologies. The initial alignment is then enhanced by the simulated annealing by generating a successor, computing its fitness, and then moving to it. Such an enhancement is recurrently repeated for some number of iterations.

The number of iterations is a parameter that can be tuned by the user. According to the number of iterations, the temperature in each iteration can be simply computed. Given the number of iterations $iter_{max}$ and the initial temperature t, the temperature at the iteration $iter$ can be

RIGHTSLINK

computed as

$$t_k = \left(1 - \frac{iter}{iter_{max}}\right) t.$$

Having computed the temperature, the overall ontology alignment algorithm is summarized in Algorithm 4.

ALGORITHM 4: SANOM

Input: Source and target ontologies O_1 and O_2, number of iteration $iter_{max}$, initial temperature t=1.
Finding the initial state S and its fitness $f(S)$ by Algorithm 2
while $iter < iter_{max}$ **do**
  $t_k = t - \frac{iter}{iter_{max}} t.$
  S' and its fitness $f(S')$ are generated by Algorithm 3.
  $\Delta E = f(S') - f(S)$.
  $P_{move} = min\left(e^{\frac{\Delta E}{t}}, 1\right)$.
  if P_{move} > random(0,1) **then**
    $S = S'$
    $f(S) = f(S')$
  end if
end while
Output State S

In terms of the time complexity, the randomized greedy initialization is of order $O(n)$ and it is only executed once. The successor generation is quite swift, and it only requires $k/2$ swap operations and $4k/2$ additions/subtractions, where k is the number of elements in the alignment to be swapped. Thus, the SA is very swift. However, the most time-consuming module is the similarity computation hash table. However, it is efficiently implemented using Java Fork/Join framework. As a result, the time complexity varies significantly with respect to n, since the string similarity is partly reliant on the length of the concept names (or the number of tokens) and structural similarity is dependent on the number of superclasses/subclasses. However, it is certain that both structural and string similarity metrics are required to be conducted for each element of two ontologies.

5 EXPERIMENTAL RESULTS

To evaluate the efficiency and efficacy of the proposed ontology alignment system, several standard datasets with a known ground truth are required. In this section, the proposed system is evaluated and compared with the state-of-the-art. We take advantage of the datasets from three tracks of the ontology alignment evaluation initiative (OAEI), i.e., anatomy, conference, and disease and phenotype tracks, to evaluate the performance of SANOM and to compare it with several other alignment systems.

The proper performance metrics are also required to gauge the fineness of an alignment. There are three common performance measures for this end. The first metric is *precision* whose magnitude shows how accurate the system is. Let R and A be the reference and an alignment identified by a system, respectively, the precision of the system A given R, e.g., Pr(A,R), is computed as

$$Pr(A, R) = \frac{|A \cap R|}{|A|},$$

where |.| is the cardinality operator.

RIGHTSLINK

Recall is another popular performance metric and is an indicator to show the completeness of a system over a mapping task. Given R and A, recall $Re(A, R)$ is calculated as

$$Re(A, R) = \frac{|A \cap R|}{|R|}.$$

The drawback of precision is that it does not consider how well the system could discover the underlying correspondences. Recall, however, only gauges the ability of a system in identifying the matches, but not its accuracy. These metrics might be useful in situations where the false correspondences are not tolerated, or the discovery of more true correspondences are appreciated even at the expense of having more false positives.

To consider both accuracy and completeness of a system, F-measure should be used, which is the harmonic mean of precision and recall, e.g.,

$$\text{F-measure}(A, R) = 2\frac{Pr(A, R) \times Re(A, R)}{Pr(A, R) + Re(A, R)}.$$

The above performance measures are the three popular metrics to gauge different facets of the performance of a given system. Along with reporting these metrics, the statistical techniques have been used as well for comparing our system with other competing ones. The McNemar's test is recommended to compare the systems when there is one mapping task (Mohammadi et al. 2018). This test is suitable to compare various systems over the OAEI anatomy track, and the corresponding results will be visualized by a directed graph.

For the conference track, the systems are compared based on the Friedman test and its related post hoc procedure (Mohammadi et al. 2017b). The results of such comparisons are visualized by the critical difference diagrams. The statistical comparison is discussed at the experiments in more details.

5.1 The Anatomy Track

The anatomy track is one of the earliest benchmarks in the OAEI. The task is about aligning the Adult Mouse anatomy and a part of NCI thesaurus containing the anatomy of humans. Each of the ontologies has approximately 3,000 classes, which are designed carefully and are annotated in technical terms. A simple string similarity measure can discover an acceptable portion of the similar concepts; however, there is a considerable share of non-trivial mappings that requires more in-depth analysis to be discovered.

SANOM is applied to this problem and then compared with AML (Faria et al. 2017), XMap (Achichi et al. 2016), LogMap and LogMapLite (Jiménez-Ruiz and Grau 2011), KEPLER (Kachroudi et al. 2017), Wiki3 (Hertling 2017), and ALIN (da Silva et al. 2017). The number of iteration was set to 1000 for this track. We applied MapPSO to the anatomy track, but its outcome was not acceptable at all with both precision and recall less than 0.05. Thus, we left it out for comparison on the anatomy track. Among the participating systems, LogMap and LogMapLite, SANOM, and ALIN are the systems that do not take advantage of any background knowledge such as UMLS Metathesaurus (Bodenreider 2004). Hence, it is evident that these systems have lower performance with respect to those with biomedical background knowledge.

The systems are first compared based on precision, recall, and F-measure, which are tabulated in Table 1. According to this table, AML is the system with the highest discovery, followed by XMap. Both of these systems have utilized biomedical background knowledge that led to the better performance.

Among the system without background knowledge, SANOM has the best performance with respect to recall, which means that the proposed system could discover more correspondences. LogMap, however, has better performance in terms of precision. The difference between these

RIGHTSLINK

Table 1. The Precision, Recall, and F-measure of Participatory Systems on the OAEI Anatomy Track

System	Precision	F-measure	Recall
AML	0.95	0.943	0.936
XMap	0.926	0.893	0.863
KEPLER	0.958	0.836	0.741
LogMap	0.918	0.88	0.846
LogMapLite	0.962	0.829	0.728
SANOM	0.888	0.870	0.853
WikiV2	0.883	0.802	0.734
ALIN	0.996	0.506	0.339

two systems is approximately 1% regarding recall and 3% with respect to precision. The overall difference between SANOM and LogMap is 1% regarding F-measure, which is the trade-off between precision and recall. Further, SANOM is interestingly quite competitive with XMap as well; their difference is nearly 1% in terms of precision and 2% regarding recall, in spite of the fact that XMap takes advantage of UMLS, dedicated background knowledge in the biomedical domain.

We further compare the systems in the anatomy track via the McNemar's mid-p test (Mohammadi et al. 2018). The McNemar's test applies to the experiments with dichotomous outcomes, hence it can be used for the alignment comparison, since their findings could be seen as dichotomous (true vs. false correspondence) as well. The McNemar's mid-p test requires two numbers from two systems for comparison: The number of correspondences that are correctly identified by the first system and not by the second, e.g., n_{10}, and the number of correctly discovered correspondences by the second system and not by the first, e.g., n_{01}.

The computation of n_{10} and n_{01} can be performed in two different ways. The first approach only considers the correct correspondences of both systems and neglects the false positive. Given the alignments A_1 and A_2 along with the reference R, the number n_{01} and n_{10} could be calculated as

$$\begin{cases} n_{01} = |(A_2 \cap R) - A_1| \\ n_{10} = |(A_1 \cap R) - A_2| \end{cases}. \tag{9}$$

By the same token, the numbers could be obtained while the false positive is considered, e.g.,

$$\begin{cases} n_{01} = |(A_2 \cap R) - A_1| + |A_1 - A_2 - R| \\ n_{10} = |(A_1 \cap R) - A_2| + |A_2 - A_1 - R| \end{cases}. \tag{10}$$

The numbers pertaining to Equations (9) and (10) are computed and displayed in Tables 2 and 3, respectively. To compare the i^{th} and j^{th} systems, the elements (i, j) and (j, i) of these tables are taken as n_{10} and n_{01}, and the McNemar's mid-p test is applied. The mid-p-value is computed as follows:

$$\text{mid-p-value} = 2 \sum_{x=n_{01}}^{n} \binom{n}{x} 0.5^n - \binom{n}{n_{01}} 0.5^n,$$

where $n = n_{01} + n_{10}$. If the null hypothesis of this test is rejected, then it is drawn that the system with higher positive discovery is the better systems. Otherwise, the systems are practically analogous.

The result of the McNemar's test over the foregoing systems are visualized by the directed graph in Figures 2 (ignoring false positive) and 3 (considering false positive). In these graphs, the nodes

RIGHTSLINK

Table 2. The Required Numbers for the Comparison of Systems via the McNemar's Test while the False Positive Is Neglected

	ALIN	AML	KEPLER	LogMap	LogMapLite	SANOM	WikiV3	XMap
Alin	0	2	0	4	3	10	14	0
AML	903	0	301	168	326	161	322	143
KEPLER	608	8	0	28	134	61	106	22
LogMap	766	29	182	0	180	73	213	53
LogMapLite	592	14	115	7	0	31	128	20
SANOM	782	32	225	83	214	0	239	77
WikiV3	610	17	94	47	135	63	0	37
XMap	788	30	202	79	219	93	229	0

The entries above the main diagonal can be viewed as n_{10}'s, and those below the diagonal would be n_{01}'s.

Table 3. The Required Numbers for the Comparison of Systems via the McNemar's Test while the False Positive Is also Considered

	ALIN	AML	KEPLER	LogMap	LogMapLite	SANOM	WikiV3	XMap
ALIN	0	74	48	121	47	166	160	103
AML	909	0	338	266	366	292	456	213
KEPLER	608	63	0	118	174	196	239	107
LogMap	766	76	203	0	184	176	346	127
LogMapLite	592	76	159	84	0	161	269	111
SANOM	783	74	253	148	233	0	370	140
WikiV3	616	77	135	157	180	209	0	121
XMap	794	69	238	173	257	214	356	0

The entries above the main diagonal can be viewed as n_{10}'s, and those below the diagonal would be n_{01}'s.

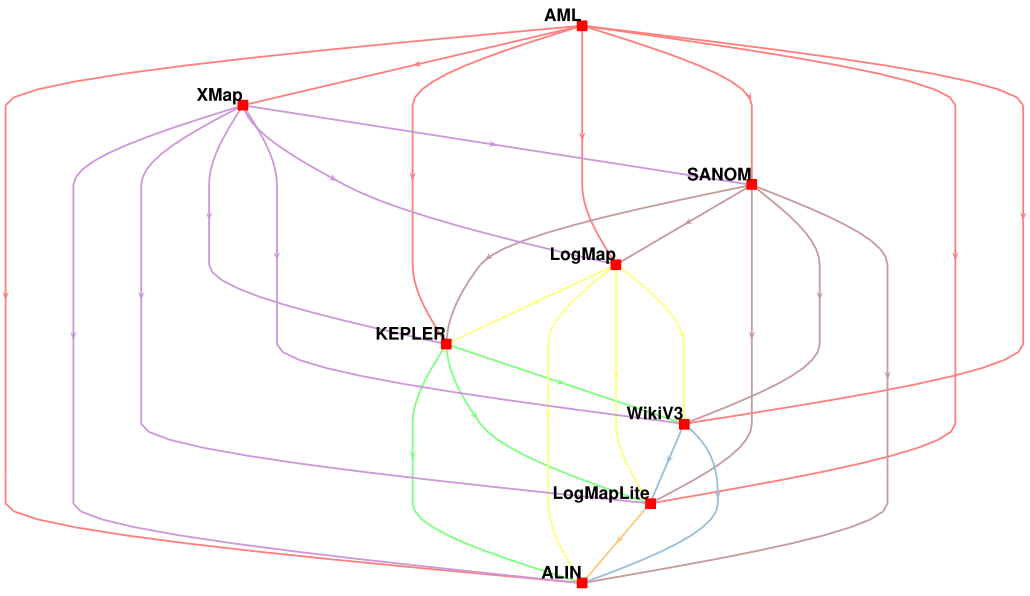

Fig. 2. The comparison of various systems via the McNemar's test while the false positive is ignored. The nodes in this graph are the systems, and the edge $A \rightarrow B$ displays the superiority of A with respect to B.

RIGHTSLINK

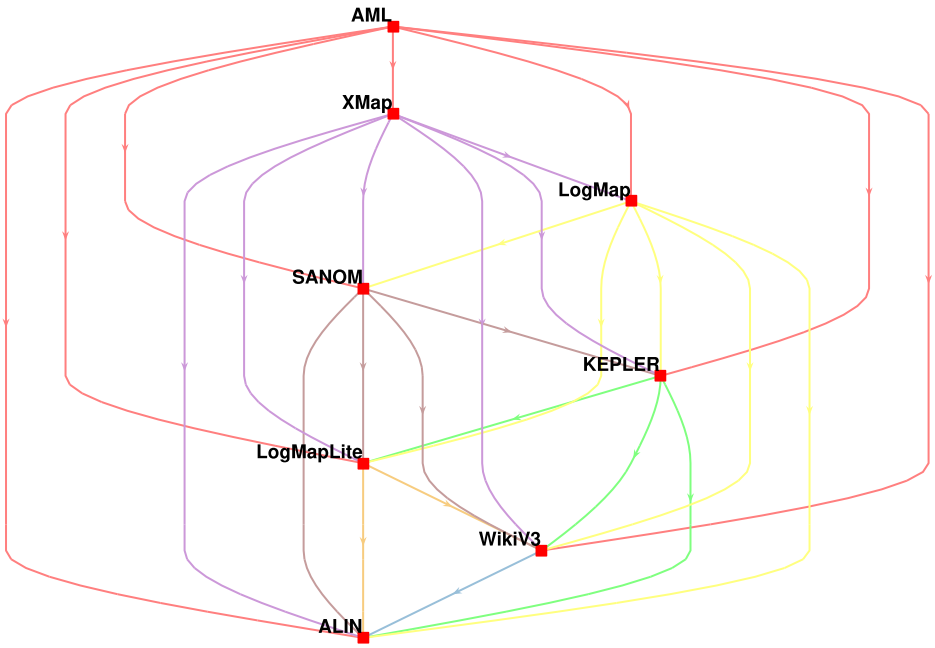

Fig. 3. The comparison of various systems via the McNemar's test while the false positive is ignored. The nodes in this graph are the systems, and the edge $A \rightarrow B$ displays the superiority of A with respect to B.

are the systems and the edge $A \rightarrow B$ indicates that A outperforms B. If there is no such edge, then the corresponding systems are identical.

As expected, SANOM outperforms LogMap when the false positive is ignored and is the third systems overall. The other two are the systems that use the background knowledge. If the false positive is considered as well, then LogMap has better performance than SANOM, which confirms that LogMap has fewer false correspondences.

In sum, SANOM consistently outperforms KEPLER, LogMapLite, WikiV3, and ALIN either the false positive is considered or ignored. In the opposite, AML and XMap have better performance than SANOM in both situations. The performance analysis of SANOM would call the need to use the biomedical background knowledge to enhance its performance in matching the biomedical ontologies.

Since the outcome of MapPSO is not acceptable, we did not compare it with SANOM in terms of precision, recall, and F-measure. However, we compare them concerning their execution times. SANOM completed the anatomy matching task in 118 seconds, while it took 806 seconds for MapPSO to complete it. The execution time on this track supports the efficiency of SANOM with respect to the population-based EAs.

5.2 The Conference Track

The experiments in this section include the alignment of seven ontologies from the conference track of the OAEI: cmt, conference, confOf, iasted, edas, ekaw, sigkdd. These ontologies describe the conference organization from different proceedings; therefore, they are heterogeneous by nature.

Pairing every two ontologies together, there are overall 21 mapping tasks. The tasks comprise not only the alignment of classes but the alignment of properties as well. Therefore, it is a suitable

RIGHTSLINK

challenge to gauge the goodness of systems for the alignment of properties. Our experience over this track showed that SANOM converges to the optimal solution with less than 100 iterations in all tracks. Thus, the number of iterations was set to 100.

Along with the systems used for comparison in the anatomy track, MapPSO is also considered for comparison, since it has a better performance in the conference track. The overall performance is also gauged by the micro and macro averaging. The macro averaging is the mean of the performance scores over all tasks. For the micro averaging, however, the false positive (FP), false negative (FN), and true positive (TP) in each task are first computed. Then, the micro averages of the precision and recall are

$$\hat{Precision} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FP_i}, \qquad \hat{Recall} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FN_i}, \tag{11}$$

where TP_i, FN_i, and FP_i are, respectively, the true positive, false negative, and false positive for the i^{th} task and $\hat{Precision}$ and $\hat{Recall}$ are, respectively, the precision and recall micro average.

Table 4 tabulates the precision, recall, and F-measure of various systems along with the micro and macro averages. In terms of recall, the proposed system has the best performance by the margin of 6% from AML, the second-best performing system, and by the margin of 14% from XMap and LogMap. Regarding precision, XMap, LogMap, and AML have better performance than SANOM. It is usually the case that the true positive increases at the expense of more false positives. Concerning F-measure, however, SANOM is superior to those of LogMap and XMap and is quite competitive with AML, which is the best-performing alignment system in this track.

The outcome of systems will be further analyzed statistically. To this end, the Friedman test and its corresponding post hoc test are used. The family-wise error rate, which happens when there are more than two systems for comparison, is controlled by the Bergmann's correction method (Bergmann and Hommel 1988). The statistical comparison over multiple datasets needs a performance metric; hence, we perform such an analysis with each of three common metrics. The outcome of the test is visualized by critical difference (CD) diagrams.

Figure 4 displays three critical diagrams regarding three performance scores, e.g., precision, recall, F-measure. The horizontal axis in these figures are the rank obtained by the Friedman test, and each red line connects multiple systems whose performances are not significantly different from each other.

Based on these figures, it is evident that SANOM is competitive with AML and is superior to other systems with respect to recall. Regarding precision, SANOM has a moderate performance but the overall performance gauged by F-measure indicates that SANOM is among the top-performing systems.

In the conference track, there is no particular background knowledge to be utilized by the systems. Thus, the comparison among participatory systems is entirely fair. Thus, the conclusion can be drawn that SANOM has a great performance in discovering new correspondences at the expense of having false discovery. One can conclude that SANOM is more suitable for the domains where the false positive is tolerable if more correspondences are to be identified.

We finally compare SANOM and MapPSO, two alignment systems based on the evolutionary algorithm, from the execution time view over a computer with CPU core-i5 and 4*GB* RAM. Table 5 shows the execution time in seconds of both systems over the mapping tasks in the conference track. It is evident that SANOM is significantly faster than MapPSO. The overall time required for MapPSO to complete all the tasks is approximately 747 seconds, while SANOM completes them in about 58 seconds. Therefore, SANOM is not only superior from precision and recall metrics but is also remarkably swift with respect to MapPSO.

RIGHTSLINK

Table 4. The Performance Measures of the Systems Over the Tasks of the Conference Track

	SANOM			AML			LogMap			XMap			LogMapLite			KEPLER			ALIN			Wiki3			MapPSO		
	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R
cmt-conference	0.61	0.74	0.93	0.67	0.59	0.53	0.73	0.62	0.53	0.73	0.62	0.53	0.56	0.42	0.33	0.53	0.56	0.60	1.00	0.42	0.27	0.38	0.36	0.33	0.05	0.09	0.27
cmt-confOf	0.80	0.62	0.50	0.90	0.69	0.56	0.83	0.45	0.31	0.86	0.52	0.38	0.67	0.48	0.38	0.55	0.44	0.38	1.00	0.22	0.13	0.63	0.42	0.31	0.07	0.10	0.19
cmt-edas	0.63	0.69	0.77	0.90	0.78	0.69	0.89	0.73	0.62	0.75	0.72	0.69	0.73	0.67	0.62	0.69	0.69	0.69	1.00	0.47	0.31	0.73	0.67	0.62	0.08	0.13	0.38
cmt-ekaw	0.54	0.58	0.64	0.75	0.63	0.55	0.75	0.63	0.55	1.00	0.71	0.55	0.56	0.50	0.45	0.55	0.55	0.55	1.00	0.43	0.27	0.71	0.56	0.45	0.09	0.15	0.45
cmt-iasted	0.67	0.80	1.00	0.80	0.89	1.00	0.80	0.89	1.00	0.80	0.89	1.00	0.80	0.89	1.00	0.50	0.67	1.00	1.00	0.67	0.50	0.67	0.80	1.00	0.04	0.07	0.50
cmt-sigkdd	0.85	0.88	0.92	0.92	0.92	0.92	1.00	0.91	0.83	1.00	0.91	0.83	0.89	0.76	0.67	0.77	0.80	0.83	1.00	0.50	0.33	0.80	0.73	0.67	0.19	0.31	0.75
conference-confOf	0.79	0.76	0.73	0.87	0.87	0.87	0.85	0.79	0.73	0.79	0.76	0.73	0.90	0.72	0.60	0.56	0.58	0.60	0.83	0.48	0.33	0.73	0.62	0.53	0.15	0.23	0.53
conference-edas	0.67	0.74	0.82	0.73	0.69	0.65	0.85	0.73	0.65	0.83	0.69	0.59	0.75	0.62	0.53	0.48	0.53	0.59	0.83	0.43	0.29	0.64	0.58	0.53	0.02	0.03	0.06
conference-ekaw	0.66	0.70	0.76	0.78	0.75	0.72	0.63	0.55	0.48	0.65	0.52	0.44	0.62	0.42	0.32	0.52	0.50	0.48	0.75	0.36	0.24	0.64	0.46	0.36	0.09	0.13	0.28
conference-iasted	0.88	0.64	0.50	0.83	0.50	0.36	0.88	0.64	0.50	0.83	0.50	0.36	0.80	0.42	0.29	0.63	0.45	0.36	1.00	0.35	0.21	0.67	0.40	0.29	0.03	0.06	0.21
conference-sigkdd	0.75	0.77	0.80	0.85	0.79	0.73	0.85	0.79	0.73	0.82	0.69	0.60	0.80	0.64	0.53	0.71	0.69	0.67	0.86	0.55	0.40	0.67	0.59	0.53	0.09	0.15	0.40
confOf-edas	0.82	0.78	0.74	0.92	0.71	0.58	0.77	0.63	0.53	0.91	0.67	0.53	0.58	0.58	0.58	0.45	0.49	0.53	0.83	0.40	0.26	0.50	0.49	0.47	0.10	0.15	0.32
confOf-ekaw	0.81	0.83	0.85	0.94	0.86	0.80	0.93	0.80	0.70	0.81	0.72	0.65	0.77	0.61	0.50	0.62	0.63	0.65	1.00	0.46	0.30	0.73	0.52	0.40	0.26	0.35	0.55
confOf-iasted	0.71	0.63	0.56	0.80	0.57	0.44	1.00	0.62	0.44	0.80	0.57	0.44	1.00	0.62	0.44	0.36	0.40	0.44	1.00	0.36	0.22	0.57	0.50	0.44	0.08	0.14	0.44
confOf-sigkdd	0.83	0.77	0.71	1.00	0.92	0.86	1.00	0.83	0.71	1.00	0.73	0.57	1.00	0.73	0.57	0.80	0.67	0.57	1.00	0.44	0.29	0.80	0.67	0.57	0.06	0.11	0.43
edas-ekaw	0.71	0.72	0.74	0.79	0.59	0.48	0.75	0.62	0.52	0.79	0.59	0.48	0.59	0.50	0.43	0.65	0.60	0.57	0.63	0.32	0.22	0.63	0.51	0.43	0.04	0.07	0.17
edas-iasted	0.69	0.56	0.47	0.82	0.60	0.47	0.88	0.52	0.37	0.88	0.52	0.37	0.78	0.50	0.37	0.64	0.47	0.37	0.75	0.26	0.16	0.80	0.55	0.42	0.03	0.05	0.16
edas-sigkdd	0.80	0.64	0.53	1.00	0.80	0.67	0.88	0.61	0.47	1.00	0.70	0.53	0.88	0.61	0.47	0.88	0.61	0.47	1.00	0.33	0.20	0.78	0.58	0.47	0.07	0.11	0.27
ekaw-iasted	0.70	0.70	0.70	0.88	0.78	0.70	0.75	0.67	0.60	1.00	0.82	0.70	0.60	0.60	0.60	0.54	0.61	0.70	1.00	0.46	0.30	0.75	0.67	0.60	0.01	0.02	0.10
ekaw-sigkdd	0.89	0.80	0.73	0.80	0.76	0.73	0.86	0.67	0.55	0.88	0.74	0.64	0.88	0.74	0.64	0.78	0.70	0.64	1.00	0.53	0.36	0.88	0.74	0.64	0.05	0.08	0.27
iasted-sigkdd	0.70	0.80	0.93	0.81	0.84	0.87	0.71	0.69	0.67	0.87	0.87	0.87	0.73	0.73	0.73	0.59	0.70	0.87	1.00	0.57	0.40	0.72	0.79	0.87	0.05	0.08	0.20
Macro-Averaging	0.74	0.72	0.73	0.84	0.74	0.67	0.84	0.68	0.59	0.86	0.69	0.59	0.76	0.61	0.53	0.61	0.59	0.60	0.93	0.43	0.29	0.69	0.58	0.52	0.08	0.12	0.33
Micro-Averaging	0.72	0.72	0.72	0.84	0.74	0.66	0.82	0.67	0.57	0.84	0.68	0.57	0.72	0.59	0.50	0.59	0.58	0.57	0.89	0.42	0.27	0.67	0.57	0.49	0.07	0.11	0.31

The metrics for comparison are precision (P), recall (R), and F-measure (F). The overall performance of each system over all tasks is gauged by the micro and macro averaging.

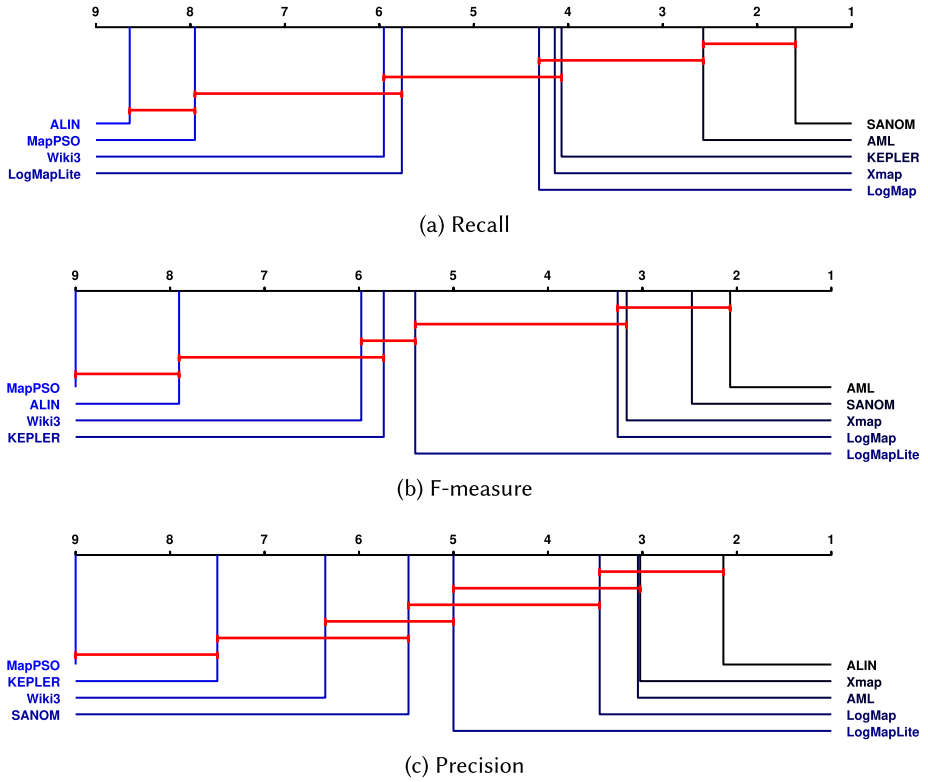

Fig. 4. The critical difference diagrams of systems on the OAEI conference tracks corresponding to three different performance measures: (a) Recall; (b) F-measure; (c) Precision. The x-axis in diagrams are the ranks obtained from the Friedman test, and red lines connect the systems whose performance are not significantly different.

Table 5. The Consumed Time for MapPSO (Bock and Hettenhausen 2012) and SANOM to Produce an Alignment for Each of the Tasks in the Conference Track

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
MapPSO	32.0	26.7	27.2	27.1	32.0	24.5	30.0	40.6	39.3	53.3	34.6	32.1	24.2	2.4	29.6	50.1	78.6	30.8	46.7	32.3	32.0
SANOM	9.4	0.9	1.8	1.2	1.5	1.7	1.7	3.2	2.2	3.4	1.6	2.0	2.3	2.3	1.1	3.0	4.9	2.8	4.3	2.3	4.3

The times are in seconds, and the number of each task corresponds to mapping two ontologies displayed in Table 4.

5.3 The Disease and Phenotype Track

SANOM is further applied to the OAEI disease and phenotype track (Harrow et al. 2017), which consists of matching various disease and phenotype ontologies. In particular, we consider the mapping of the human phenotype (HP) to the mammalian phenotype (MP), and aligning the human disease ontology (DOID) and the orphanet and rare diseases ontology (ORDO).

The ontologies in this track contain approximately 15,000 concepts; therefore, the alignment of these ontologies is challenging. Faria et al. (Faria et al. 2018) investigated the challenges of large biomedical ontologies, and they recommended several ways of dealing with these ontologies. Some of these recommendations have been used in SANOM, which makes it possible to align even ontologies with this size.

RIGHTSLINK

Table 6. The Precision, Recall, and F-measure of the Systems Participated in Aligning DOID and ORDO Ontologies from the Disease and Phenotype Track

	Precision	F-measure	Recall
LogMap	0.937	0.848	0.775
AML	0.514	0.646	0.870
LogMapLite	0.988	0.758	0.615
XMap	0.969	0.700	0.548
SANOM	0.975	0.747	0.605

Table 7. The Precision, Recall, and F-measure of the Systems Participated in Aligning HP and MP Ontologies from the Disease and Phenotype Track

	Precision	F-measure	Recall
LogMap	0.875	0.855	0.835
AML	0.889	0.843	0.801
LogMapLite	0.993	0.755	0.609
XMap	0.994	0.477	0.314
SANOM	0.995	0.728	0.574

MapPSO could not find the alignment of ontologies in this track, since it requires a massive amount of memory space. Thus, it cannot be compared with SANOM on this track.

For the reference, a *voted* reference alignment has been used that was created based on the outputs of the alignment systems participated in this track for the last three years. A reasoner was also used to validate the final alignment.

In comparison to the systems participated in other tracks of the OAEI, fewer systems can generate a reliable alignment for this track. All other participating systems in this track use a biomedical background knowledge. In particular, LogMap uses normalizations and spelling variants the SPECIALIST Lexicon,[5] XMAP uses a dictionary of synonyms extracted from the UMLS Metathesaurus (Bodenreider 2004), and AML has three background resources, one of which is selected automatically (Faria et al. 2014). The current version of SANOM, however, does not utilize any sort of background knowledge for the biomedical domain.

Table 6 tabulates the result of the various alignment systems for aligning DOID and ORDO ontologies. According to this table, the precision of SANOM is better than those of AML, LogMap, and XMap, and is competitive with LogMapLite. In terms of recall, however, AML and LogMap have better outcomes. SANOM is also better than XMap and is competitive with LogMapLite. Regarding F-measure, LogMap is the best system in this track, followed by LogMapLite and SANOM. Thus, SANOM outperformed XMap and AML in this track in spite of the fact that it does not use any background knowledge.

Table 7 displays the performance of systems on aligning HP and MP ontologies. According to this table, SANOM has excellent performance in terms of precision and outperforms all systems in this sense. Regarding recall, LogMap and AML are the top two systems, and SANOM is better than XMap and is competitive with LogMapLite. Concerning F-measure, LogMap and AML are the best systems, followed by LogMapLite and SANOM.

[5] http://wayback.archive-it.org/org-350/20180312141706/https://www.nlm.nih.gov/pubs/factsheets/umlslex.html.

RIGHTSLINK

The outcomes of SANOM on the disease and phenotype tracks are acceptable, but interestingly, its precision is high in contrast to other tracks. This gets back to the nature of the ontologies and the fact that SANOM has no use of biomedical background knowledge. Thus, SANOM can consider only the concepts as potential mappings that have a high string or structural similarity. In this case, mapping based solely on these similarity metrics has led to high precision and low recall. Another important topic is that the reference alignment has been created based on the alignments of other systems in previous years. Thus, the participating systems have contributions to the reference alignment, which means that it is more likely that they have much more correspondences in common with the reference. SANOM, however, has not participated in this track before and has therefore no impact on the creation of the reference. Nevertheless, the performance of SANOM is comparable and acceptable.

6 CONCLUSION AND DISCUSSION

This article presented a new ontology alignment system, called SANOM, which uses the simulated annealing to find the correspondences among two given ontologies. The problem of ontology matching was first revised as the minimization of a fitness function. Then, a compound fitness function was developed using several similarity metrics. The simulated annealing was then adapted to optimize the energy function and consequently derive the final alignment. SANOM has shown a great performance in discovering the correspondences of two given ontologies. Further, it is also fast and memory-efficient, especially in comparison to other alignment systems using evolutionary algorithms.

However, there are multiple avenues on which SANOM can be improved. SANOM has already acceptable performance in terms of recall, but its precision is not as good as its recall. This is probably due to the nature of the evolutionary algorithms that compute the overall performance of an alignment. Thus, the chances are that in an intermediate state with the superior fitness, there are multiple false correspondences that can be simply refuted. Rejecting the false correspondences could be done using an alignment repair technique. Thus, one avenue to enhance the precision of SANOM would be the use of an alignment repair technique.

Yet another way of improving SANOM is to use background knowledge such as UMLS. Most of the tracks of the OAEI lie within the biomedical realm, hence utilization of such background knowledge would increase the performance on those tracks and help us fairly compare it with competing ones over those tracks.

SANOM is both memory- and time-efficient. However, mapping the big ontologies (e.g., ontologies with more than 50,000 concepts, is another problem. Recently, there are several suggestions to enable the alignment systems matching these ontologies as well. For instance, one can store the concepts of one ontology in a hash table and then search the concepts of the second into this hash table. Since finding in hash tables is of the order one, the overall searching of all concepts is of the order m, where m is the number of concepts in the second ontology. It was shown that such a strategy finds many correspondences in the biomedical domain and decreases the consequent search space quadratically. Such suggestions are left for the future development of SANOM.

REFERENCES

Giovanni Acampora, Hisao Ishibuchi, and Autilia Vitiello. 2014. A comparison of multi-objective evolutionary algorithms for the ontology meta-matching problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'14)*. IEEE, 413–420.

Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, et al. 2016. Results of the ontology alignment evaluation initiative 2016. In *OM: Ontology Matching*. No commercial editor., 73–129.

RIGHTSLINK

Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. 2006. CEL—A polynomial-time reasoner for life science ontologies. In *Proceedings of the International Joint Conference on Automated Reasoning*. Springer, 287–291.

Anita Bandrowski, Ryan Brinkman, Mathias Brochhausen, Matthew H. Brush, Bill Bug, Marcus C. Chibucos, Kevin Clancy, Mélanie Courtot, Dirk Derom, Michel Dumontier, et al. 2016. The ontology for biomedical investigations. *PloS One* 11, 4 (2016), e0154556.

Beate Bergmann and Gerhard Hommel. 1988. Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple Hypothesenprüfung/Multiple Hypotheses Testing*. Springer, 100–115.

Jürgen Bock and Jan Hettenhausen. 2012. Discrete particle swarm optimisation for ontology alignment. *Inform. Sci.* 192 (2012), 152–173.

Olivier Bodenreider. 2004. The unified medical language system (UMLS): Integrating biomedical terminology. *Nucl. Acids Res.* 32, suppl_1 (2004), D267–D270.

Michelle Cheatham and Pascal Hitzler. 2013. String similarity metrics for ontology alignment. In *Proceedings of the International Semantic Web Conference*. Springer, 294–309.

Michelle Cheatham and Pascal Hitzler. 2014. The properties of property alignment. In *Proceedings of the 9th International Conference on Ontology Matching (OM'14)*, Vol. 1317, 13–24.

William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string metrics for matching names and records. In *Proceedings of the Kdd Workshop on Data Cleaning and Object Consolidation*, Vol. 3. 73–78.

Jomar da Silva, Fernanda Araujo Baiao, and Kate Revoredo. 2017. ALIN results for OAEI 2017. In *Proceedings of the 12th International Workshop on Ontology Matching (OM'17)*. 114.

Jérôme Euzenat, Pavel Shvaiko, et al. 2007. *Ontology Matching*. Vol. 18. Springer.

Daniel Faria, Booma Sowkarthiga Balasubramani, Vivek R. Shivaprabhu, Isabela Mott, Catia Pesquita, Francisco M. Couto, and Isabel F. Cruz. 2017. Results of AML in OAEI 2017. In *Proceedings of the 12th International Workshop on Ontology Matching (OM'17)*. 122.

Daniel Faria, Catia Pesquita, Isabela Mott, Catarina Martins, Francisco M. Couto, and Isabel F. Cruz. 2018. Tackling the challenges of matching biomedical ontologies. *J. Biomed. Sem.* 9, 1 (2018), 4.

Daniel Faria, Catia Pesquita, Emanuel Santos, Isabel F. Cruz, and Francisco M. Couto. 2014. Automatic background knowledge selection for matching biomedical ontologies. *PloS One* 9, 11 (2014), e111226.

Jorge Martinez-Gil, Enrique Alba, and José F. Aldana-Montes. 2008. Optimizing ontology alignments by using genetic algorithms. In *Proceedings of the Workshop on Nature Based Reasoning for the Semantic Web, Karlsruhe.*

Ian Harrow, Ernesto Jiménez-Ruiz, Andrea Splendiani, Martin Romacker, Peter Woollard, Scott Markel, Yasmin Alam-Faruque, Martin Koch, James Malone, and Arild Waaler. 2017. Matching disease and phenotype ontologies in the ontology alignment evaluation initiative. *J. Biomed. Sem.* 8, 1 (2017), 55.

Sven Hertling. 2017. WikiV3 results for OAEI 2017. In *Proceedings of the 12th International Workshop on Ontology Matching (OM'17)*. 190.

Robert Hoehndorf, Paul N. Schofield, and Georgios V. Gkoutos. 2015. The role of ontologies in biological and biomedical research: A functional perspective. *Brief. Bioinform.* 16, 6 (2015), 1069–1080.

Matthew Horridge and Sean Bechhofer. 2011. The OWL API: A Java API for OWL ontologies. *Sem. Web* 2, 1 (2011), 11–21.

Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. 2011. Logmap: Logic-based and scalable ontology matching. In *Proceedings of the International Semantic Web Conference*. Springer, 273–288.

Marouen Kachroudi, Gayo Diallo, and Sadok Ben Yahia. 2017. OAEI 2017 results of KEPLER. In *Proceedings of the 12th International Workshop on Ontology Matching*. 138.

Doug Lea. 2000. A Java fork/join framework. In *Proceedings of the ACM Conference on Java Grande*. ACM, 36–43.

Marcos Martínez-Romero, José Manuel Vázquez-Naya, Francisco Javier Nóvoa, Guillermo Vázquez, and Javier Pereira. 2013. A genetic algorithms-based approach for optimizing similarity aggregation in ontology matching. In *Proceedings of the International Work-Conference on Artificial Neural Networks*. Springer, 435–444.

Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys.* 21, 6 (1953), 1087–1092.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

Majid Mohammadi, Amir Atashin, Wout Hofman, and Yaohua Tan. 2017a. SANOM results for OAEI 2017. In *Proceedings of the 12th International Workshop on Ontology Matching*. 185.

Majid Mohammadi, Amir Ahooye Atashin, Wout Hofman, and Yaohua Tan. 2018. Comparison of ontology alignment systems across single matching task via the McNemar's test. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 4 (2018), 51.

Majid Mohammadi, Wout Hofman, and Yao-Hua Tan. 2019. A comparative study of ontology matching systems via inferential statistics. *IEEE Transactions on Knowledge and Data Engineering* 31, 4 (2019), 615–628.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

RIGHTSLINK

Femke Reitsma, John Laxton, Stuart Ballard, Werner Kuhn, and Alia Abdelmoty. 2009. Semantics, ontologies, and eScience for the geosciences. *Comput. Geosci.* 35, 4 (2009), 706–709.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Vol. 1. Association for Computational Linguistics, 173–180.

Junli Wang, Zhijun Ding, and Changjun Jiang. 2006. Gaom: Genetic algorithm-based ontology matching. In *Proceedings of the IEEE Asia-Pacific Conference on Services Computing*. IEEE, 617–620.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 133–138.

Xingsi Xue and Junfeng Chen. 2018. Optimizing ontology alignment through hybrid population-based incremental learning algorithm. *Memetic Comput.* (2018), 1–9.

Xingsi Xue, Jianhua Liu, Pei-Wei Tsai, Xianyin Zhan, and Aihong Ren. 2015a. Optimizing ontology alignment by using compact genetic algorithm. In *Proceedings of the 11th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 231–234.

Xingsi Xue and Shijian Liu. 2017. Compact evolutionary algorithm-based ontology meta-matching. In *Proceedings of the International Conference on Smart Vehicular Technology, Transportation, Communication and Applications*. Springer, 213–221.

Xingsi Xue and Yuping Wang. 2015. Optimizing ontology alignments through a memetic algorithm using both MatchFmeasure and unanimous improvement ratio. *Artific. Intell.* 223 (2015), 65–81.

Xingsi Xue and Yuping Wang. 2017. Improving the efficiency of NSGA-II based ontology aligning technology. *Data Knowl Eng.* 108 (2017), 1–14.

Xingsi Xue, Yuping Wang, and Weichen Hao. 2015b. Optimizing ontology alignments by using NSGA-II. *Int. Arab J. Inform. Technol.* 12, 2 (2015).

Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE Trans. Patt. Anal. Mach. Intell.* 29, 6 (2007), 1091–1095.

Received June 2018; revised January 2019; accepted February 2019

RIGHTSLINK