



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

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

An adaptive ontology mapping approach with neural network based constraint satisfaction[☆]

Ming Mao^{a,*,3}, Yefei Peng^{b,1,3}, Michael Spring^{c,2}^a SAP Labs, 3410 Hillview Ave, Palo Alto, CA 94306, USA^b Yahoo!, 701 First St, Sunnyvale, CA 94089, USA^c University of Pittsburgh, 135 N Bellefield Ave, Pittsburgh, PA 15260, USA

ARTICLE INFO

Article history:

Received 22 January 2009

Received in revised form 3 November 2009

Accepted 12 November 2009

Available online 17 November 2009

JEL classification:

D.2.12 [Software Engineering]:

Interoperability–Data mapping

H.3.3 [Information Systems]: Information Search and Retrieval–Retrieval models

I.2.6 [Artificial Intelligence]:

Learning–Connectionism and neural nets

Keywords:

Constraint satisfaction problem (CSP)

Harmony-based adaptive aggregation

Interactive activation and competition

(IAC) network

Ontology mapping

PRIOR+

ABSTRACT

Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies. It is a key challenge to achieve semantic interoperability in building the Semantic Web. This paper proposes a new generic and adaptive ontology mapping approach, called the PRIOR+, based on propagation theory, information retrieval techniques and artificial intelligence. The approach consists of three major modules, i.e., the IR-based similarity generator, the adaptive similarity filter and weighted similarity aggregator, and the neural network based constraint satisfaction solver. The approach first measures both linguistic and structural similarity of ontologies in a vector space model, and then aggregates them using an adaptive method based on their harmonies, which is defined as an estimator of performance of similarity. Finally to improve mapping accuracy the interactive activation and competition neural network is activated, if necessary, to search for a solution that can satisfy ontology constraints. The experimental results show that harmony is a good estimator of f-measure; the harmony based adaptive aggregation outperforms other aggregation methods; neural network approach significantly boosts the performance in most cases. Our approach is competitive with top-ranked systems on benchmark tests at OAEI campaign 2007, and performs the best on real cases in OAEI benchmark tests.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

The World Wide Web (WWW) is widely used as a universal medium for information exchange. Automatically exchanging data and reusing the exchanged data in the WWW is limited due to the heterogeneity problem existing in information resources, and the non-semantic nature of HTML and URLs. Information heterogeneity occurs at three levels, i.e., syntax, structure and semantics [38]. Syntactic heterogeneity is the simplest heterogeneity problem caused by the usage of different data formats. To solve the syntactic

heterogeneity, standardized formats such as XML,⁴ RDF/RDFS⁵ and OWL⁶ have been widely used to describe data in a uniform way that makes automatic processing of shared information easier. However standardization cannot overcome structural heterogeneity which occurs as a result of the way information is structured even in homogeneous syntactic environments. For example, one source might model trucks but only classify them into a few categories; while the other source might make very fine-grained distinctions between types of trucks based on their physical structure, weight, purpose, etc. Manually encoded transformation rules as well as some middleware components have been used to solve structural heterogeneity problems [42]. Though sophisticated solutions to syntactic and structural heterogeneity have been developed, the problem of semantic heterogeneity is still only partially solved. Semantic heterogeneity occurs whenever two contexts do not

[☆] This paper has been revised and extended from the authors' previous work [23–25].

* Corresponding author. Tel.: +1 650 852 3868.

E-mail addresses: ming.mao@sap.com (M. Mao),

ypeng@yahoo-inc.com (Y. Peng), spring@sis.pitt.edu (M. Spring).

¹ Tel.: +1 408 349 8446.

² Tel.: +1 412 624 9429.

³ This work was done when the authors had affiliation with University of Pittsburgh.

⁴ <http://www.w3.org/TR/2004/REC-xml-20040204>.

⁵ <http://www.w3.org/TR/rdf-schema/>.

⁶ <http://www.w3.org/TR/owl-features/>.

share the same interpretation of information (e.g. homonyms and synonyms). For example, the semantic search engine Swoogle returns 346 documents when searching for *spring*.⁷ The top-ranked results show that the same term has many different meanings, e.g. one *spring* means the *season*, the other *spring* means the *ground water*, etc. Though synonym sets, term networks, concept lattices, features and constraints have been proposed as solutions for solving semantic heterogeneity among different information systems, those approaches are not sufficient to solve the problem of semantic heterogeneity in the WWW environment [38].

The vision of the Semantic Web [1] provides many new perspectives and technologies to overcome the limitation of the WWW. Ontologies are a key component to solve the problem of semantic heterogeneity, and thus enable semantic interoperability between different web applications and services because of its ability to provide formal and explicit definitions of data and allow reasoning over related concepts. Though ontologies are being used more and more, no universal ontology exists for the WWW. Given the reality of multiple ontologies over many domains, ontology mapping that aims to find semantic correspondences between similar elements of different ontologies has been the subject of research in various domains and applications [32].

The following use cases show how ontology mapping can help to achieve semantic correspondences in different scenarios, and thus motivate our work in this area. First of all, ontology mapping is important to the success of the Semantic Web. The pervasive usage of agents or web services is a characteristic of the Semantic Web. However agents or web services may use different protocols that are independently designed, which means when agents or web services meet, there is little chance for them to understand each other without an “interpreter”. Therefore ontology mapping is “a necessary precondition to establish interoperability between agents or services using different ontologies” [7] (page 2). That is, the mapping between ontologies provides the means for agents and services to either translate their messages or integrate bridge axioms in their own models. Secondly, ontology mapping is also widely used to support data integration and information transformation. For example, a web marketplace such as Amazon⁸ may need to combine products from multiple vendors’ catalogs into its own. A web portal like NCSTRL⁹ may want to integrate documents from multiple library directories into its own. A company may want to merge its service taxonomy with its partners. A researcher may want to merge his/her bookmarks with those of his/her peers, etc. Moreover, from the perspective of information retrieval, ontology mapping can support semantic query processing across disparate sources by expanding or rewriting the query using the corresponding information in multiple ontologies. For example, a user is looking for the director of a movie, e.g., “Star War”, on the Web. In one movie website, the name of movie is identified as “*moviename*” and the name of its director is identified as “*director*” in its schema. However in another movie website, those two concepts might be identified as “*title*” and “*directorname*”, respectively in their schema. Therefore to enable a federated search on those two websites, a mapping between the schemas of those two websites will help us rewrite queries according to different schemas.

Ontology mapping can be done either by hand or using automated tools. Manual mapping becomes impractical as the number, size and complexity of ontologies increases. Automatic ontology mapping has shown its importance in practical applications including the Semantic Web [1], information transformation and data

integration [6], query processing across disparate sources [14], to name a few. Fully or semi-automated mapping approaches have been examined in various research studies, e.g., analyzing linguistic information of elements in ontologies [20,34,39], utilizing information retrieval techniques [21,22,36], treating ontologies as structural graphs [28,31,41], using heuristic rules to look for specific mapping patterns [15] and applying machine learning techniques [5,26]. Comprehensive surveys of ontology mapping systems and approaches can be found in [9,19,32].

Though the state of the art approaches have made significant progresses in ontology mapping, they suffer from two limitations. First, ontology mapping approaches that use multiple mapping strategies meet the problem of aggregating multiple similarities. Currently they either use some predefined experience numbers to weight different similarities or tentatively set parameters in aggregation functions (e.g. sigmoid). Manually setting parameters is impractical due to its inability to adapting to different ontology mapping tasks. A second limitation is that most ontology mapping approaches do not thoroughly deal with ontology constraints (e.g., the hierarchical relations in RDFS, the constraints and axioms in OWL, and the rules in SWRL¹⁰). Most approaches either ignore ontology constraints completely or deal with ontology constraints based on some heuristic rules [8,18,31]. To the best of our knowledge, exceptions are GLUE [5], which adopts relaxation labeling to optimize mapping configurations by considering ontology constraints, and RiMOM [39], which uses risk minimization to search for the optimal mappings from the results output by multiple strategies. To overcome the limitations, in this paper we propose a new weight assignment method to adaptively aggregate different similarities and then adopt a neural network based constraint satisfaction model to improve overall mapping performance from previously aggregated results. Actually our adaptive aggregation method can be integrated in most matching systems that estimate multiple similarities. Though the neural network model can be implemented independently from our adaptive aggregation approach, the parameter (i.e. harmony) provides a good guidance on where the neural network model should be activated or not.

Our contributions in this paper are twofold:

- We propose a measure *harmony* to estimate performance of similarities for ontology mapping, without given the ground truth. Based on the estimation we adaptively aggregate various similarities and adjust our mapping strategies.
- We propose to use the interactive activation and competition (IAC) neural network to search for a solution that best satisfies ontology constraints.

To evaluate our approach, we adopt the benchmark tests from OAEI ontology matching campaign 2007. We follow the evaluation criteria of OAEI, calculating the precision, recall and f-measure of each test case. Experimental results show that harmony has high correlation with f-measure of individual similarity; the harmony based adaptive aggregator outperforms all existing aggregation methods; the IAC neural network boosts mapping performance significantly. We also compare our approach with top-ranked systems that participated in the campaign.

The rest of the paper is organized as follows. Section 2 illustrates and defines ontology mapping problem with a simple example. Section 3 explains our approach in details. Section 4 describes our evaluation methodology and discusses experimental results. Section 5 reviews some related work followed by a summary and future work at the end.

⁷ Based on the search results returned from <http://swoogle.umbc.edu/> in July, 2007.

⁸ <http://www.amazon.com/>.

⁹ <http://www.ncstrl.org/>.

¹⁰ <http://www.daml.org/2003/11/swrl/>.

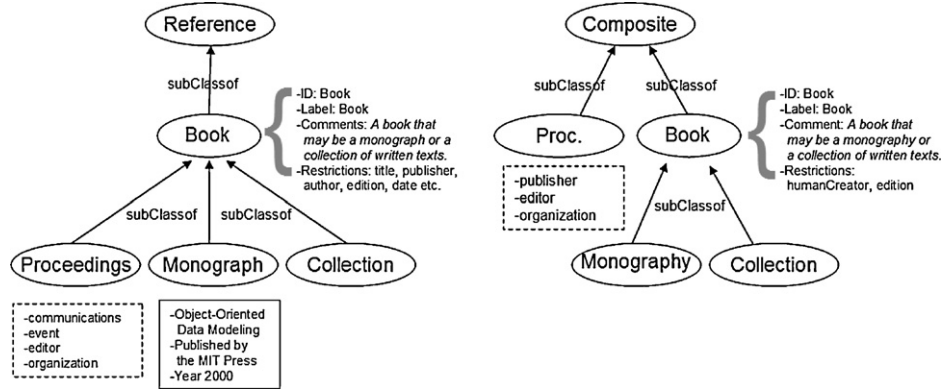


Fig. 1. Two bibliographic ontologies.

2. Problem statement

Ontology is a formal, explicit specification of a shared conceptualization in terms of concepts (i.e., classes), properties and relations [12]. Classes can be associated instances as well. Fig. 1 shows two sample ontologies in Bibliography area, in which the ellipses indicate classes (e.g., “Reference”, “Composite”, “Book” and “Proceedings”, etc.), the dashed rectangles indicate properties (e.g., “Publisher”, “Editor”, “Organization”, etc.), the lines with arrow-head indicate “subClassof” relation between two classes, and the solid rectangle indicates an instance that is associated with the class of “Monograph” (i.e., “object-oriented data modeling” published by the MIT Press at 2000). Each class and property has some information to describe and restrict it. For example, the descriptive information of Book in the left ontology includes its ID, label, comment and restrictions such as title, publisher, etc.

Ontology mapping aims to find semantic correspondences between similar elements in two ontologies. In this paper, semantic correspondence refers to “=” relationship and elements refer to classes and properties. The input of a mapping task is two homogeneous ontologies, O_1 and O_2 , expressed in formal ontology languages. The output is a mapping, also called the mapping result, between the input ontologies. Mapping can be represented in different ways such as queries [3], bridging axioms [6] or an instance in a mapping ontology [14]. We define mapping results as a statement 4-tuple (as written in Eq. (1)), where m is a mapping that specifies a specific element e_{1i} in O_1 corresponds to a certain element e_{2j} in O_2 with a relationship of r , and the mapping holds a confidence measure of s (also known as *similarity*), which is typically normalized in a range $[0..1]$.

$$m(e_{1i}, e_{2j}, r, s) \quad (1)$$

Candidate mappings in Fig. 1 include *Reference* and *Composite*, *Book* and *Book*, *Monograph* and *Monography*, *Collection* and *Collection*, *Proceedings* and *Proc*. They can be represented as: $m(\text{Reference}, \text{Composite} = .11)$, $m(\text{Book}, \text{Book} = 1)$, $m(\text{Monograph}, \text{Monography} = .9)$, $m(\text{Collection}, \text{Collection} = 1)$, $m(\text{Proceedings}, \text{Proc} = .36)$.

3. Our approach

Generally speaking, our approach, which we call PRIOR+, first measures the similarities of both linguistic and structural information of ontology in a vector space model using classic information retrieval technique. Next it adaptively aggregates different similarities according to their predicted performance and adjusts mapping strategies based on the performance of final similarity. Finally it utilizes the IAC neural network to solve the constraint satisfaction

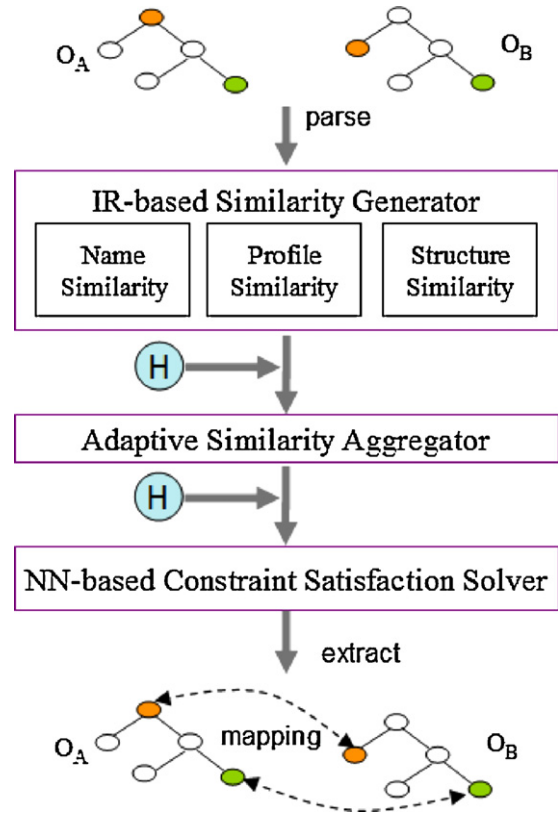


Fig. 2. The architecture of PRIOR+.

problem (CSP) in the context of ontology mapping. The architecture of our approach is shown in Fig. 2, where H denotes the harmony of similarities. The details of its three major modules, i.e., the IR-based Similarity Generator, the Adaptive Similarity Aggregator and the Neural Network (i.e., the IAC neural network) based Constraint Satisfaction Solver, are given in the following sections.

3.1. IR-based similarity generator

In this section we briefly describe the generation of different similarities and refer interested readers to our previous work [20–22] for details. The input of the similarity generator is two ontologies, which will be parsed by Jena¹¹ and each element of

¹¹ <http://jena.sourceforge.net/>.

which will be pre-processed by removing stop words, stemming, and tokenizing. The outputs are three similarity matrixes that contain similarity scores for each pair of elements in ontologies.

3.1.1. Name similarity

The name similarity is calculated based on the edit distance between the name (i.e., ID) of elements. The name-based similarity is defined as Eq. (2), where $EditDist(e_{1i}, e_{2j})$ is Levenshtein distance between elements e_{1i} and e_{2j} , $l(e_{1i})$ and $l(e_{2j})$ are the string length of the name of e_{1i} and e_{2j} , respectively.

$$NameSim(e_{1i}, e_{2j}) = 1 - \frac{EditDist(e_{1i}, e_{2j})}{\max(l(e_{1i}), l(e_{2j}))} \quad (2)$$

3.1.2. Profile similarity

The profile similarity is generated in three steps:

- **Profile Enrichment.** For each element in the ontology, we generate a *profile* (i.e., a combination of the element's descriptive information) to represent it and thus enrich its information. In particular, the profile of a class = the class's ID + label + comments + other restriction + its properties' profiles + its instances' profiles. The profile of a property = the property's ID + label + its domain + its range. The profile of an instance = the instance's ID + label + other descriptive information. Then the *tf-idf* weight is assigned for each profile based on the whole collection of all profiles in the ontology.
- **Profile Propagation.** To exploit the neighboring information of each element, the profile of the element's ancestors, descendants and siblings will be passed to that of the element with different weights.
- **Profile Mapping.** Finally the cosine similarity between the profiles of two elements of e_{1i} and e_{2j} is calculated in a vector space model using Eq. (3), where $V_{e_{1i}}$ and $V_{e_{2j}}$ are two vectors representing the profile of element e_{1i} and e_{2j} , respectively, n is the dimension of the profile vectors, $V_{ke_{1i}}$ and $V_{ke_{2j}}$ are k th element in the profile vector of element e_{1i} and e_{2j} , respectively, $|V_{e_{1i}}|$ and $|V_{e_{2j}}|$ are the lengths of the two vectors, respectively.

$$ProfileSim(e_{1i}, e_{2j}) = \frac{\vec{V}_{e_{1i}} \cdot \vec{V}_{e_{2j}}}{|V_{e_{1i}}| |V_{e_{2j}}|} = \frac{\sum_{k=1}^n (V_{ke_{1i}} * V_{ke_{2j}})}{\sqrt{\sum_{k=1}^n (V_{ke_{1i}}^2)} \sqrt{\sum_{k=1}^n (V_{ke_{2j}}^2)}} \quad (3)$$

3.1.3. Structural similarity

The structural similarity between two elements comes from their structural features (e.g., the number of direct property of a class). Structural similarity is considered for classes only. No structural similarity will be given to property and instance due to the lack of hierarchical information. The structural similarity of the classes in two ontologies is defined by Eq. (4), where e_{1i} and e_{2j} are two class elements in ontology O_1 and O_2 , respectively, n is the total number of structure features, $diff_k(e_{1i}, e_{2j})$ denotes the difference for feature k .

$$StructSim(e_{1i}, e_{2j}) = \frac{\sum_{k=1}^n (1 - diff_k(e_{1i}, e_{2j}))}{n} \quad (4)$$

The $diff_k(e_{1i}, e_{2j})$ is defined as Eq. (5), where $sf(e_{1i})$ and $sf(e_{2j})$ denote the value of structure features of e_{1i} and e_{2j} , respectively. In our approach $sf(e_{1i})$ and $sf(e_{2j})$ are one of the following values, i.e., the number of the class' direct properties, the number of the class' instances, the number of the class' children, and the normalized

depth of the class from the root.

$$diff(e_{1i}, e_{2j}) = \frac{|sf(e_{1i}) - sf(e_{2j})|}{\max(sf(e_{1i}), sf(e_{2j}))} \quad (5)$$

Here we give an example of depth difference. Assume the max depth of ontology O_1 is 5, the max depth of ontology O_2 is 6. $depth(e_{1i}) = 3$, $depth(e_{2j}) = 4$. We first normalize the depth as $sf(e_{1i}) = 3/5 = .6$, $sf(e_{2j}) = 4/6 = .67$. Then calculate depth difference: $diff(e_{1i}, e_{2j}) = |.6 - .67| / \max(.6, .67) = .07 / .67 = .10$.

3.2. Harmony

In this section we introduce the term *harmony* to estimate the importance and reliability of different similarities. Totally five individual harmonies (i.e., class name harmony, class profile harmony, class structural harmony, property name harmony, and property profile harmony) and two final harmony (i.e. class harmony, property harmony) will be estimated on the corresponding similarities.

3.2.1. The definition of harmony

As we stated in Section 1 that manually setting parameters to aggregate different similarities is impractical due to its inability to adapting to different ontology mapping tasks; meanwhile, the fact that different similarities work well in different situations motivates us to investigate a new measure that can estimate the quality of each similarity so that we can aggregate them according to their individual characteristic (see the comparison of individual similarity in Section 4.2.1). Therefore we are looking for a measurement that (1) can tell us which similarity is more reliable and trustful so that we can give it a higher weight during aggregation and filter out false mappings that have low similarity; (2) can assist us to adaptively adjust our mapping strategy, i.e., when to activate or inactivate NN-based constraint satisfaction solver.

Ideally, for 1-1 mapping, the similarity score of two truly mapped elements should be larger than that of all other pairs of elements that share the same row/column with the two elements in the similarity matrix, which implies that the two elements of this pair mutually prefer each other. Given the rationale, we define the *harmony* of the similarity matrix as Eq. (6), where $\#s_max$ denotes the number of the pair of elements that has the highest and the only highest similarity in its corresponding row and column in the similarity matrix, and $\#e_i$ denotes the number of elements in ontology O_i .

$$h = \frac{\#s_max}{\min(\#e_1, \#e_2)} \quad (6)$$

3.2.2. A simple example of harmony estimation

Table 1 is the name similarity matrix for the example illustrated in Fig. 1. The upper table lists the original similarity score between each pair of elements. The lower table illustrates how the harmony is calculated in this case, where “x” denotes the cell that has the highest similarity score in each row, “O” denotes the cell that has the highest similarity score in each column, and “⊗” denotes the overlapped cell that has the highest similarity in both the row and the column. The cells that have highest similarity score in each row or column are indicated in bold in Table 1. Therefore the $\#s_max$ in Eq. (6) is the number of “⊗” in Table 1. In this case, $\#s_max$ is 4 and thus the harmony of the name similarity matrix is $4/5 = .8$.

3.3. Adaptive similarity aggregator

3.3.1. Similarity aggregation in the state-of-art ontology mapping approaches

Aggregating different similarity is pervasive in ontology mapping systems that contain multiple individual matchers, for

Table 1
A sample of harmony calculation.

	Composite	Book	Proc.	Monography	Collection
Reference	.11	0	.22	.1	.1
Book	.22	1	.2	.2	.2
Proceedings	.18	.09	.36	.09	.18
Monograph	.11	.22	.11	.9	.1
Collection	.3	.2	.1	.1	1

$$Harmony = 4/5 = 0.8$$

		×		
	⊗			
		⊗		
			⊗	
○				⊗

example, COMA [4], Falcon-AO [34], RiMOM [39], and QOM [8], etc. Many strategies, e.g., *Max*, *Min*, *Weighted*, *Average* and *SIGMOD*, have been proposed to aggregate different similarities in the approaches. The *Max/Min* strategy returns the maximal/minimal similarity of individual matchers. The *Weighted* strategy determines a weighted sum of similarity of individual matchers. The *Average* strategy is one special case of the *Weighted* strategy and returns the average similarity over all individual matchers. The *SIGMOD* strategy combines multiple results using a sigmoid function, which is essentially a smoothed threshold function.

Among the strategies, the *Max/Min* strategy selects one extreme end of various similarities to be the representative of the final similarity, which is either too optimistic (e.g. *Max*) especially in case of contradicting similarities or too pessimistic (e.g. *Min*). The *Average* strategy considers the individual similarities equally important and cannot distinguish differences between them. The *Weighted* strategy overcomes the drawbacks of the *Average* strategy by assigning relative weights to individual matchers. The *SIGMOD* strategy emphasizes high individual predicting values and deemphasizes low individual predicting values.

Currently the systems that adopt the *Weighted* strategy or the *SIGMOD* strategy to aggregate similarities need to manually set aggregation weights based on experience for different similarities or tentatively set center position and steepness factor in the sigmoid function. However, manually predefined parameters cannot be generalized to adapt to different mapping situations. To our best knowledge, one exception to aggregate different similarity without manually setting weights is to use machine learning method proposed in [17]. However, this approach needs training data, which is not available in most cases. Therefore how to select appropriate parameters that can truly reflect the reliability of different similarities without knowing the ground truth in advance deserves further research.

3.3.2. Adaptive similarity aggregation

In this section we propose a new weight assignment method to adaptively aggregate different similarities. That is, we use the *harmony* of different similarities as *weight* to aggregate various similarities. Therefore the final similarity of the pair of elements (e_{1i}, e_{2j}) can be defined by Eq. (7), where h_k denotes the harmony of different similarities as defined in Eq. (6), n denotes the number of different types of similarity, and $Sim_k(e_{1i}, e_{2j})$ denotes the similarity of each

pair of elements.

$$FinalSim(e_{1i}, e_{2j}) = \frac{\sum_k h_k \times Sim_k(e_{1i}, e_{2j})}{n} \quad (7)$$

The input of the adaptive similarity aggregator is a set of individual similarity matrixes as described in Section 3.1. The output is an aggregated similarity matrix, which we call final similarity matrix. The comparison of the harmony-based adaptive similarity aggregation and other aggregation methods is given in Section 4.2.3.

There are noises in similarity matrix, especially in the case of name similarity. Even two names are not related at all, we may still get a positive edit distance based similarity if there is one shared letter. Harmony provides a way to filter out noises from similarity matrix. If a matrix has high harmony, we have high confidence that the lowest ranked similarities in each row and column are noises. Therefore, before aggregation we filter out a proportion of lower ranked similarities in each row and column based on harmony. The number of similarities filtered out is shown in Eq. (8), where p is number of lowest ranked similarities filtered out in each row or column, L is length of the row or column, h is harmony of the similarity matrix. The purpose of $(L - 1)$ in this equation is to make sure we have one similarity left for each row and column.

$$p = \min(L - 1, (h \times L)) \quad (8)$$

3.4. NN-based constraint satisfaction solver

3.4.1. The constraint satisfaction problem

A constraint satisfaction problem (CSP) is “a problem composed of a finite set of **variables**, each of which is associated with a finite **domain**, and a set of **constraints** that restricts the values the variables can simultaneously take. The task is to assign a value to each variable satisfying all the constraints.” [40] Classic examples of CSPs include the map coloring problem, sudoku, eight queens puzzle, etc.

CSP arises as an intriguing research problem in ontology mapping due to the fact that the characteristics of an ontology and its representations result in many kinds of constraints. For example, the hierarchical relations in RDFS do not allow crisscross mappings, the axioms such as owl:sameAs and owl:equivalentClass in OWL indicate an equivalent relation between different elements, and the rules in SWRL would be to imply or assert some properties that are not directly available. To improve the quality of ontology mapping, we need to find a configuration that can best satisfy those constraints.

CSPs are typically solved by a form of search, e.g. backtracking, constraint propagation or local search [40]. In [30] McClelland and Rumelhart described how to use the IAC neural network to solve CSPs in detail. Next we briefly introduce the IAC neural network, and then explain how to use it in ontology mapping.

3.4.2. The IAC network

In general, an IAC neural network consists of a number of competitive *nodes* connected to each other. Each node represents a *hypothesis*. The *connection* between two nodes represents constraint between their hypotheses. Each connection is associated with a *weight*. The activation of a node is determined locally by the nodes adjacent to it and the weights connecting to it. Furthermore, if two hypotheses support each other, the connection between them is *positive* (i.e., *excitatory*); whereas if two hypotheses are against each other, the connection between them is *negative* (i.e., *inhibitory*). The weight of the connection is proportional to the strength of the constraint. The stronger a constraint is, the larger the corresponding weight is.

The mechanism of the IAC neural network can be illustrated using the following simple example. Suppose we have two grids, 1 and 2, and two constraints:

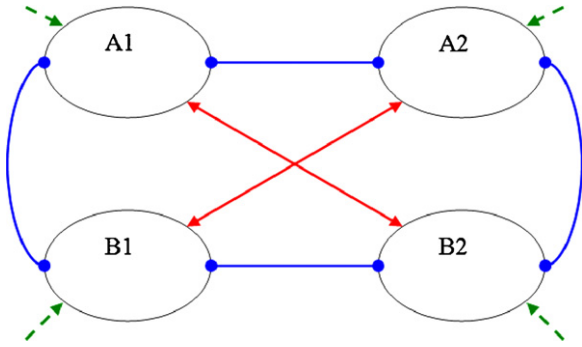


Fig. 3. A simple constraint network.

1. Each grid can have one value, either A or B.
2. The values of the two grids are different.

We have four hypotheses: A in grid 1 (H_{A1}); B in grid 1 (H_{B1}); A in grid 2 (H_{A2}); B in grid 2 (H_{B2}). Based on the two constraints we know there are two negative connections and one positive connection for each hypothesis:

1. H_{Ai} is against H_{Bi} , and vice versa ($i = 1$ or 2).
2. H_{x1} is against H_{x2} , and vice versa ($x = A$ or B).
3. H_{Ai} supports H_{Bj} , and vice versa ($i, j = 1$ or 2 , and $i \neq j$).

Fig. 3 illustrates the simple example, where each node represents a hypothesis, the line with rounded head and arrowhead represents negative connection and positive connection between hypothesis respectively and the dashed line with arrowhead represents a small stimulus on each node from outside.

Assume the negative weight is half the positive weight and all nodes are inactive at start. Though the input from three neighbors of the node will cancel out, the small excitatory input from outside will activate the node. All nodes will update their states asynchronously. That is, nodes are chosen to be updated sequentially in random order. Finally, either A1 and B2 or B1 and A2 will be active, and the network will reach a *stable state*, called *settled* or *relaxed solution*.

3.4.3. The IAC neural network in the context of ontology mapping

In the context of ontology mapping, a node in the IAC neural network represents a hypothesis that indicates element e_{1i} in ontology O_1 can be mapped to element e_{2j} in ontology O_2 . The connections between nodes in the network represent constraints between hypotheses. In Fig. 4 we can see the input of the network includes the initial activation of each node (i.e., the priori probability of a hypothesis), its bias, external inputs and a weight matrix responding to the connections between different hypotheses. In

our approach the initial activation of each node is set to the final similarity of (e_{1i}, e_{2j}) output from the adaptive similarity aggregator. The activation of the node can be updated using the following simple rule, where a_i denotes the activation of node i , written as n_i , net_i denotes the net input of the node.

$$a_i(t+1) = \begin{cases} a_i(t) + net_i(1 - a_i(t)), & net_i > 0 \\ a_i(t) + net_i a_i(t), & net_i < 0 \end{cases}$$

The net_i comes from three sources, i.e. its neighbors, its bias, and its external inputs. The net_i is defined as Eq. (9), where w_{ij} denotes the connection weight between n_i and n_j , a_j denotes the activation of node n_j , $bias_i$ denotes the bias of n_i , and ei_i denotes the external input of n_i , which is a function of the confidence of a mapping. Note that the weight matrix is symmetric and the nodes may not connect to themselves, i.e., $w_{ij} = w_{ji}$, $w_{ii} = 0$.

$$net_i = \sum_j w_{ij} a_j + bias_i + ei_i \quad (9)$$

The network can be stopped after running n cycles or at some *goodness* point. It is obviously that forcing the network to stop at a predefined cycle number usually is not optimal. In the rest of the section, we introduce how we stop the network according to its goodness.

McClelland and Rumelhart [30] defined the *goodness* (short for *goodness of fit*) as the degree to which the desired constraints are satisfied. They pointed out that the goodness depends on three things. First, it depends on the extent to which each node satisfies the constraints imposed upon it by other nodes. Thus, if a connection between two nodes is positive, the constraint is satisfied to the degree that both nodes are active. Otherwise if the connection is negative, the constraint is violated to the degree that both nodes are active. A simple way to express this is the product of the activation of two nodes times the weight connecting them, i.e., $w_{ij} a_i a_j$. Note that for positive weights the more active the two units are, the better the constraint is satisfied; whereas for negative weights the more active the two units are, the less the constraint is satisfied. Second, the priori probability of a hypothesis is captured by adding the bias to the goodness measure. Finally the goodness of a node when direct evidence is available is given by the product of the input value times the activation value of the node. The bigger this product, the better the system is satisfying this external constraint. Overall the *goodness* of node i can be defined as in Eq. (10), where the symbols share the same definition as Eq. (9).

$$goodness_i = \sum_j w_{ij} a_i a_j + bias_i a_i + ei_i a_i = net_i a_i \quad (10)$$

Since we are concerned with the degree to which the entire pattern of values assigned to all of the hypotheses are consistent with the entire body of constraints, the *overall goodness* is defined as

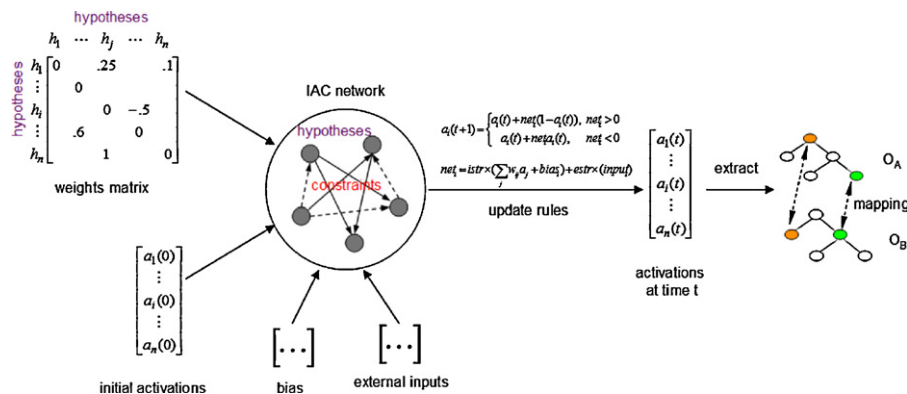


Fig. 4. The IAC neural network in the context of ontology mapping.

Table 2

The constraints used in our approach.

#	Constraints	Connection
1	Only 1-1 mapping is allowed.	Negative
2	No crisscross mapping is allowed.	Negative
3	If children elements match, then their parent elements match.	Positive
4	If parent elements match, then their children elements match.	Positive
5	If e_{1i} match e_{2j} , then e_{1s} match e_{2t} , where e_{1i} and e_{1s} , e_{2j} and e_{2t} are siblings in ontologies.	Positive
6	If property elements match, then their domain elements match.	Positive
7	If property elements match, then their range elements match.	Positive
8	If class elements match, then their direct property elements match.	Positive
9	If property elements match, then their mother-class elements match.	Positive
10	If class elements match, then their individual elements match.	Positive
11	If individual elements match, then their mother-class elements match.	Positive
12	Two elements match if their owl:sameAs or owl:equivalentClass or owl:equivalentProperty elements match.	Positive

the sum of all individual goodnesses. Now the constraint satisfaction problem is converted to the problem of maximizing the overall goodness. In practice we look for the $\delta_{goodness}$ between time t and $t-1$ (see Eq. (11)) less than a threshold to be a stop condition. To be noted, it cannot guarantee to find a global optimal solution. It could reach a local maximal or stops too early.

$$\delta_{goodness} = \sum_i goodness_i(t) - \sum_i goodness_i(t-1) \quad (11)$$

3.4.4. The implementation of the IAC neural network

Many constraints, e.g. the cardinality of a property, have been used to restrict ontologies. Different constraints result in different connections between nodes in the IAC neural network. For example, the constraint that “only 1-to-1 mapping is allowed” results in a negative connection between nodes (e_{1i}, e_{2j}) and (e_{1i}, e_{2k}) , where $k \neq j$. Moreover, “two elements match if their children match”, results in a positive connection between nodes (e_{1i}, e_{2j}) and (e_{1k}, e_{2t}) , where e_{1k} and e_{2t} are the children of e_{1i} and e_{2j} , respectively.

Table 2 lists the constraints that have been implemented in our approach. Although the number of negative constraints is much less than that of positive constraints, the ratio of negative connection and positive connection is not small. This is because each node in the network will have a huge amount of negative connections introduced by the 1-1 mapping of constraints. Though the weights of different constraints should be set to a function of its confidence, currently we just set the weight of positive constraints as 1 and the weight of negative constraints as -1.

4. Evaluation

To evaluate our approach we have applied the benchmark tests from OAEI ontology matching campaign 2007.¹² The reasons why we choose OAEI benchmark tests are: 1. The annual campaign has become an authoritative contest in the area of ontology mapping, and thus attracts many participants including both well-known ontology mapping systems and new entrants. 2. The campaign provides uniform test cases for all participants so that the analysis and comparison between different approaches is practical. 3. The ground truth of benchmark tests is open. Thus we can use it to comprehensively evaluate different components of our approach.

4.1. Data sets and evaluation criteria

The OAEI benchmark tests include 1 reference ontology O_R , dedicated to the very narrow domain of bibliography, and multiple test ontologies, O_T , that discard various information from the reference ontology in order to evaluate how algorithms behave when

Table 3

The overview of OAEI benchmark tests.

Tests	Description
#101–104	O_R and O_T have exactly the same or totally different names
#201–210	O_R and O_T have the same structure but different linguistics in some level
#221–247	O_R and O_T have the same linguistics but different structure
#248–266	Both structure and linguistics are different between O_R and O_T
#301–304	O_T are real world cases

information is lacking, except real cases. More specifically, benchmark tests can be divided into 5 groups as shown in Table 3. All benchmark tests can be downloaded here.¹³

We adopt the evaluation criteria used by the campaign. That is, standard evaluation measures *precision*, *recall* and *f-measure* will be computed against the reference alignments. The *precision*, *recall* and *f-measure* are defined as Eqs. (12), (13) and (14). For the matter of aggregation of the measures weighted harmonic means [10] will be computed.

$$precision \quad p = \frac{\#correct_found_mappings}{\#all_found_mappings} \quad (12)$$

$$recall \quad r = \frac{\#correct_found_mappings}{\#all_possible_mappings} \quad (13)$$

$$f-measure \quad f = \frac{2 \times p \times r}{p + r} \quad (14)$$

4.2. Experimental design and results

To evaluate our approach, we designed four experiments. Each answers one of the questions we are interested in:

1. What is the performance of each individual similarity, i.e., edit distance based name similarity, profile similarity, and structural similarity? Could we estimate the quality of each similarity in different situation so that we can aggregate them according to their individual characteristic?
2. Can the measure of *harmony* reflect the reliability of different similarities? That is, does it correlate to the performance of a similarity?
3. Is the harmony-based adaptive aggregation method better than other aggregation methods discussed in Section 3.3.1?
4. Does the IAC neural network work in the context of ontology mapping? If it does, how much improvement can it make? Furthermore, can the harmony assist the IAC neural network to adjust its mapping strategy for a better performance?
5. How does our approach perform compared with others?

¹² <http://oei.ontologymatching.org/2007/>.

¹³ <http://oei.ontologymatching.org/2007/benchmarks/>.

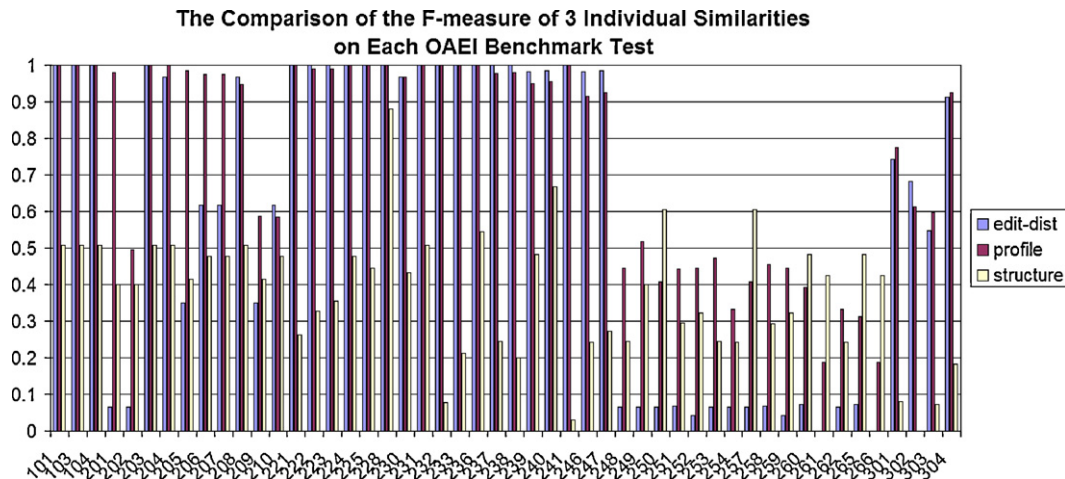


Fig. 5. The comparison of the f-measure of 3 individual similarities on each OAEI benchmark test.

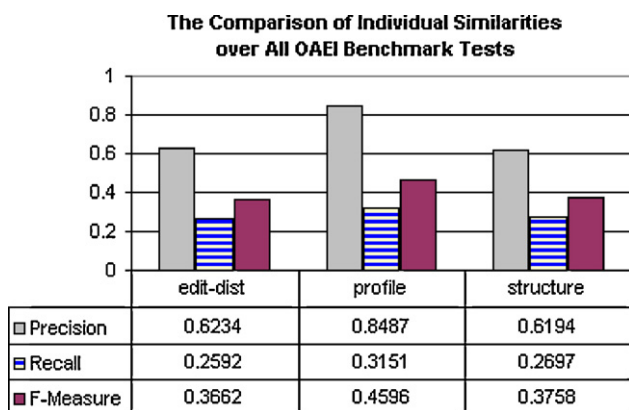


Fig. 6. The comparison of individual similarities over all OAEI benchmark tests.

4.2.1. The comparison of each individual similarity

We proposed three kinds of similarities, i.e., edit distance based name similarity, profile similarity, and structure similarity. Each similarity measures the correspondence between two elements from a different perspective. Fig. 5 compares the performance (i.e., f-measure) of 3 individual similarities on each OAEI benchmark test. Fig. 6 compares the performance (i.e., f-measure) of each individual similarity over all OAEI benchmark tests.

The observations from Figs. 5 and 6 are:

1. The edit distance based similarity is intuitive. It works very well on the cases that have high similarity between the names of elements in ontologies. For example, Test #101–104, #203, #208, #221–247, #301, #302, #304, etc. However such similarity is more lexical-oriented than semantic-oriented, which encounters trouble where synonyms exist. In the cases that have very low linguistic similarity, e.g., #201, #202 and #248–266, the performance of the edit distance based similarity is very poor. One solution to overcome the limitation of edit distance based similarity is to check auxiliary information in a thesaurus, e.g. WordNet.¹⁴ However integrating WordNet will cost much more time in finding synonymous relations between words, and thus decrease the efficiency of the whole approach.
2. The structure similarity explores structural features in two ontologies. It is extremely useful in pure graphic mapping tasks,

for example in #248–266 where meaningful linguistic information has been removed or replaced by some randomly generated information. However structure similarity contributes very little in the cases where linguistic information is adequate, e.g., #221–247, or in the case where structural information is limited, e.g., #301, #303–304, or does not exist at all, e.g. #302, in which the hierarchy of the ontology is absolutely flat. Finally, the overall recall of structure similarity is as low as .27, which indicates relying on this similarity only cannot help us to find most mappings.

3. The profile similarity utilizes all kinds of descriptive information to generate a profile for each element, and then compares the cosine similarity of two profiles in a vector space model. The profile similarity works very well when linguistic information is adequate, e.g., #101–104, #221–247, #301, #302, #304. Meanwhile, since the profile similarity explores the structural information of an element by integrating its property information, instance information and neighboring information, it also works well in the cases where linguistic information is limited, e.g., #201, #202, #205–207, #209, #248–266. Generally speaking, the profile similarity takes advantage of both edit distance comparison and structure analysis, and thus it outperforms edit distance based similarity and structure similarity in most cases except #250, #257, #261, #265, #266, where no or very little lexical information is available. Therefore, the mapping totally relies on the structure information. The precision, recall and f-measure of the profile similarity over all OAEI benchmark tests are .85, .31 and .46, respectively.
4. The fact that different similarities work well in different situations motivates us to investigate a new measure that can estimate the quality of each similarity so that we can aggregate them according to their individual characteristic.

4.2.2. The correlations between harmony and the characteristic of similarities

As stated in Section 3.2 that the harmony is defined to estimate the importance and reliability of a similarity, which can be reflected by its performance, e.g. f-measure. Thus we conduct this experiment to evaluate the correlation between harmony and f-measure.

The experiment methodology is: for each test, we calculate 5 similarities, i.e., class name similarity, class profile similarity, class structural similarity, property name similarity and property profile similarity. For each similarity matrix, we extract mapping results using naive descendant extraction algorithm [27]. After that we evaluate the results against the reference alignment and get the f-measure of each similarity. Meanwhile, we estimate 5 harmonies

¹⁴ <http://wordnet.princeton.edu/>.

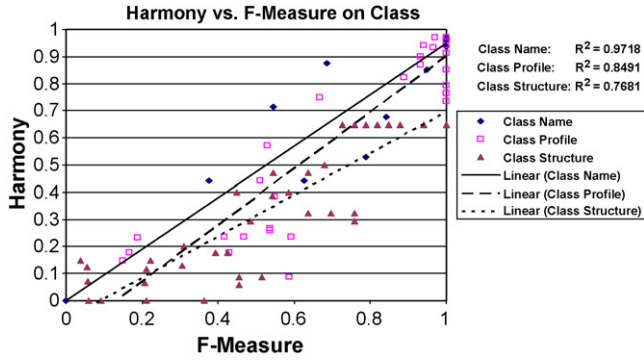


Fig. 7. The correlation of harmony vs. f-measure on class.

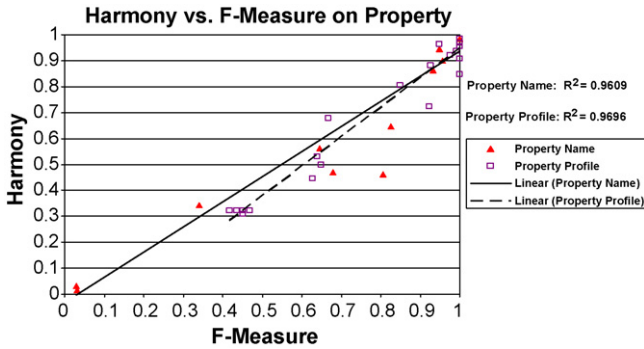


Fig. 8. The correlation of harmony vs. f-measure on property.

upon its corresponding matrix. Finally we compare the f-measure with the harmony on each similarity.

Fig. 7 and Fig. 8 demonstrate that harmony does linearly correlate with f-measure of different similarities. The R^2 for name similarity, profile similarity, and structural similarity on class are .97, .85 and .77, respectively in Fig. 7. The R^2 for the name similarity and profile similarity on property are .96 and .97, respectively in Fig. 8. The data show that the harmony is indeed a good estimator of f-measure for each individual similarity, especially on estimating the performance of class' ID, property's ID and property's profile. The observations make us confident on using harmony to adaptively aggregate similarities and adjust mapping strategies.

4.2.3. The comparison of different aggregation methods

Similarity aggregation has been researched in many ontology mapping approaches as discussed in Section 3.3.1. Data aggregation, called *data fusion*, has also been widely investigated in information retrieval area [13]. To evaluate the harmony-based adaptive weighted aggregation method, short as HADAPT, we conduct the experiment to compare the performance of HADAPT with six other aggregation methods selected from both ontology mapping and information retrieval area. Table 4 lists the name and brief

Table 4
Aggregation functions used in experiment.

Method	Description	Equation
HADAPT	Harmony-based adaptive aggregation	$fs = (h_i \times f(s_i)) / N$
MIN	Minimum of individual similarities	$fs = \min(s_i)$
MAX	Maximum of individual similarities	$fs = \max(s_i)$
AVG	Average of individual similarities	$fs = \text{sum}(s_i) / N$
ANZ	AVG/number of non-zero similarities	$fs = (\text{sum}(s_i) / N) / Nz$
MNZ	AVG \times number of non-zero similarities	$fs = (\text{sum}(s_i) / N) * Nz$
SIGMOID	Average of individual similarities smoothed by sigmoid	$fs = \text{sum}(\text{sigmoid}(s_i)) / N$

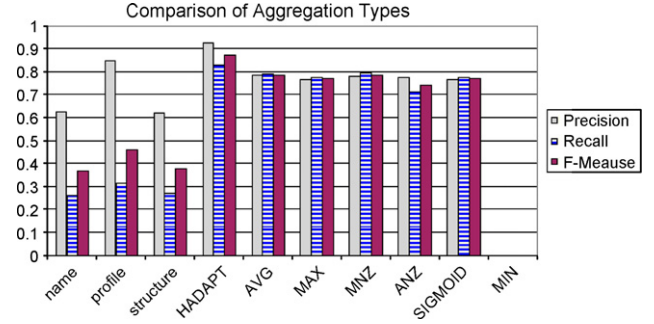


Fig. 9. Comparison of aggregation types.

description of seven aggregation methods, where s_i denotes the i th similarity, f denotes the adaptive filter function, fs denotes the final aggregated similarity, h_i denotes the harmony of i th similarity, N denotes the number of individual similarity, Nz denotes the number of non-zero similarities.

The experiment methodology is: for each test, we first calculate three individual similarities (i.e. name similarity, profile similarity and structural similarity) as described in Section 3.1. Then we aggregate the individual similarities using different aggregation methods as listed in Table 4. After aggregation, we apply the naive descendant extraction algorithm [27] to extract final mappings. Precision, recall and f-measure of final results on each test are calculated. Finally the overall precision, recall and f-measure are calculated over all benchmarks tests, which are shown in Fig. 9.

The first observation is the performance of individual similarity is very poor. Their recalls are lower than .32; f-measures are lower than .5. On the contrary the performance is dramatically boosted by all aggregation methods except MIN. They achieve recalls higher than .7, f-measure higher than .7. The data demonstrate that aggregation methods are very effective in improving the performance of mapping approaches that rely on measuring multiple similarities. Finally Fig. 9 shows the harmony-based adaptive similarity aggregation method (i.e., HADAPT) outperforms all other methods when aggregating different similarities. It holds the highest precision, recall, and f-measure at .92, .83 and .87, respectively.

4.2.4. The improvement of NN-based constraint satisfaction solver

If the performance of the aggregated similarity is good, the NN-based constrain satisfaction solver won't be activated. This judgment is based on harmony of aggregated similarity. When the harmony of final similarity is less than a threshold (.8 in our experiment) the network will start. The effect of the harmony threshold

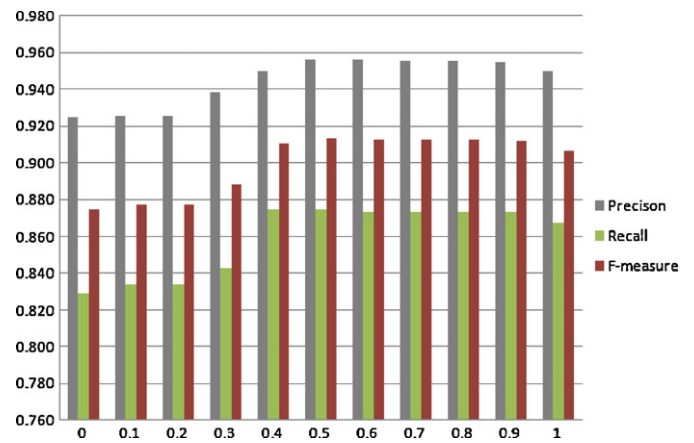


Fig. 10. The impact of different harmony thresholds on the performance of mapping.

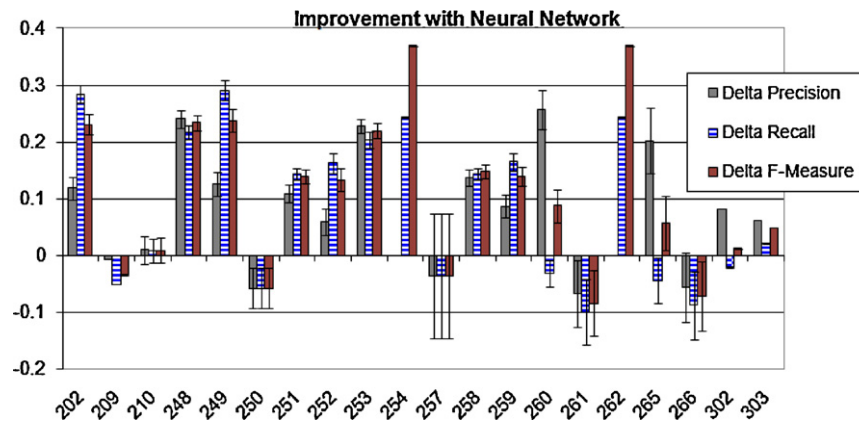


Fig. 11. The improvement with neural network.

Table 5

The number of possible mappings when NN-based CSP solver is applied.

Test #	# of possible mappings
202	5185
209	5121
210	5121
248	5185
249	5218
250	1089
251	5053
252	6338
253	5218
254	1089
257	1089
258	5086
259	6371
260	1021
261	2242
262	1089
265	1021
266	2242
302	2349
303	6390

was evaluated, and the result is shown in Fig. 10. Overall performance is almost the same when the threshold is between 0.5 and 0.9. There is slight drop of performance when the NN solver is applied to all tests (i.e., threshold = 1). It shows that when there is a perfect match, NN solver could introduce a little noise.

Currently, the NN-based constraint satisfaction solver will be activated on 20 tests, i.e., 202, 209, 210, 248–266, 302 and 303. The number of possible mappings for these tests is shown in Table 5.

Fig. 11 shows the change of the performance of these tests after applying the IAC neural network. 10 runs of NN-based constraint satisfaction solver were performed; mean and standard deviation of the 10 runs are reported in Fig. 11. Among 20 tests, 15 get improved on their f-measure, only 5 get lower f-measures. The biggest improvement of f-measure is .37. The overall improvement on these 20 tests is shown in Table 6. The NN-based CSP solver dramatically improves both precision and recall by more than .1. If we calculate percentage improvement, that is 13%, 24%, and 19% for precision, recall, and f-measure, respectively. To take a further

Table 6

Overall improvement with neural network.

H-mean	Precision	Recall	F-measure
Before NN	.76	.54	.63
After NN	.88	.67	.76
NN improvement	13%	24%	19%

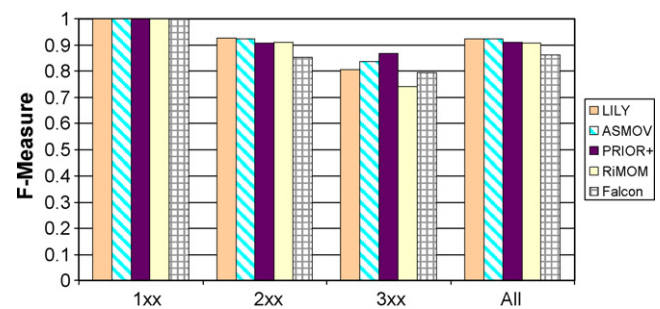


Fig. 12. The comparison of PRIOR+ with top-ranked systems on benchmark tests in OAEI campaign 2007.

look, the reason why the performance of test 209, 250, 257, 261 and 266 drop is: first of all, all of them have very limited linguistic information and some of them (e.g. 257, 261, 266) even do not have any linguistic information available at all. For example, all entity names and all comments have been removed in 209; and all the classes in 261 starting with capital letter are added, which will not have any mappings in 101. Therefore the original mapping performance is very poor for all these tests and further the rules that make use of previous mapping results in NN cannot help in this way. Meanwhile most of them (except 209) change the structure of the test ontology. For example, some add new classes in a new layer and the others flatten the hierarchy of the ontology. Therefore some rules in NN (e.g. children–parents rule) are not correct anymore. This is a drawback of NN approach. However if we take a look at real cases (e.g. 302, 303) that usually have linguistic information available, we will see the impact of structure difference decreases. On the other hand, for all the other tests that have been improved by NN approach, they have both some linguistic information and some consistent structure information that can help the rules implemented in NN to improve the overall performance.

4.2.5. The comparison between the PRIOR+ and top-ranked systems in OAEI campaign 2007

Fig. 12 compares the performance of PRIOR+ and 4 top-ranked ontology mapping systems (i.e., ASMOV [18], Lily [41], RiMOM [39], and Falcon-AO [34]) on the benchmark tests at OAEI campaign 2007. The evaluation data of these 4 systems can be downloaded here.¹⁵ The data of PRIOR+ can be downloaded here.¹⁶

¹⁵ <http://oaei.ontologymatching.org/2007/results/zip/>.

¹⁶ <http://www.sis.pitt.edu/~mingmao/om07/PRIORPLUS-result.zip> This data is slightly different from what we submitted to the OAEI campaign after improving the PRIOR+ approach.

The results in Fig. 12 shows the overall f-measure of PRIOR+ (.912) is competitive to that of RiMOM (.907), Falcon-AO (.890), LILY (.925) and ASMOV (.924). Moreover, all systems perform perfectly on test 1xx. Though the f-measure of PRIOR+ is not as good as that of LILY, ASMOV and RiMOM on test 2xx, PRIOR+ performs the best on real world cases, i.e., 3xx. To get a more meaningful comparison, Wilcoxon test was performed to compare PRIOR+ with the other four methods on precision, recall and f-measure. Except PRIOR+'s f-measure is significantly better than Falcon's ($p = 0.007$), there is no significant difference between PRIOR+ and the other four methods (LILY, ASMOV, RiMOM, Falcon), in all cases, p -value > 0.1 .

5. Related work

Different approaches have been proposed to solve the ontology mapping problem. The comprehensive survey of some famous ontology mapping systems, such as GLUE [5], QOM [8], Similarity Flooding [28], PROMPT [31], can be found in [9,19,32]. Here we only review 4 top-ranked systems that participated in OAEI campaign 2007, i.e., Falcon-AO [34], RiMOM [39], LILY [41], ASMOV [18], and the GLUE system [5]. The reason of reviewing 4 OAEI campaign participants is: 1. The techniques that the 4 systems used in their approach are diverse and based on the state-of-art approaches. In reviewing these systems, we are reviewing the latest developments in this area. 2. Like ours, all the systems are based on multiple similarities, and thus face the problem of aggregating different similarities in an effective way. Therefore, reviewing the approaches is critical to evaluate our approach by comparing each other. 3. The OAEI campaign provides uniform test cases for all participants so that the quantitative comparison between different approaches is practical. The reason of reviewing GLUE is due to its usage of relaxation labeling to optimize mapping configuration, which shares the same insight with us that satisfying ontology constraints as much as possible is critical to improve the accuracy of ontology mapping.

Falcon-AO [34] is a similarity-based generic ontology mapping system. It consists of three elementary matchers, i.e., V-Doc, I-Sub [37], and GMO, and one ontology partitioner, PBM. V-Doc constructs a virtual document for each URIref, and then measures their similarity in a vector space model. I-Sub compares the similarity of strings by considering their similarity along with their differences. GMO explores structural similarity based on a bipartite graph. PBM partitions large ontologies into small clusters, and then matches between and within clusters. The *profile* used in our approach is similar as the *virtual document* constructed in Falcon-AO. The difference is the virtual document only exploits neighboring information based on RDF model; whereas our profile does not have any limitation of information type, and thus can integrate any information including instance. From the aggregation view, though Falcon-AO measures both linguistic comparability and structural comparability of ontologies to estimate the reliability of matched entity pairs, it only uses them to form three heuristic rules to integrate results generated by GMO and LMO. In LMO Falcon-AO also linearly combines two linguistic similarities with some experiential number. Unfortunately neither experiential number nor heuristic rules can automatically adapt to different test cases, as we argued in Section 3.3.1. Furthermore, when estimating linguistic comparability Falcon-AO does not distinguish the difference between class and property; whereas our approach estimates harmony for class and property separately. Finally Falcon-AO does not have solutions to optimize final results so that they can satisfy various ontology constraints.

RiMOM [39] is a general ontology mapping system based on Bayesian decision theory. It utilizes normalization and NLP techniques and integrates multiple strategies for ontology map-

ping. Afterwards RiMOM uses risk minimization to search for optimal mappings from the results of multiple strategies. Both RiMOM and our approach do propagation based on propagation theory [11]. However RiMOM propagates the similarity of two entities to entity pairs associated with some kinds of relationship (e.g. superClassOf, siblingClassOf, domain, etc.); whereas our approach propagates original information of an element instead of its similarity to its neighboring elements, and then compares their similarity based on the propagated profiles. Another difference is when integrating multiple strategies RiMOM adopts a sigmoid function with tentatively set parameters, which has been demonstrated not good as harmony-based adaptive similarity aggregation (see Fig. 9). Furthermore, though RiMOM calculates two similarity factors to estimate the characteristics of ontologies, their estimation is suitable to some special situations only. For example, their linguistic similarity factor only concerns elements that have the same label. Meanwhile, the harmony in our approach is more general. Finally we explore different approaches to find the optimal mappings for final results extraction. RiMOM uses risk minimization approach; while we try neural network approach.

LILY [41] is a generic ontology mapping system based on the extraction of semantic subgraph. It exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. Then a subsequent similarity propagation strategy is applied to produce more alignments if necessary. Finally LILY uses classic image threshold selection algorithm to automatically select threshold, and extract final results based on the stable marriage strategy. One limitation of LILY is that it needs to manually set the size of subgraph according to different mapping tasks and the efficiency of semantic subgraph is very low in large-scale ontologies. Furthermore, as with most mapping approaches, LILY combines all separate similarities with experiential weights, and it does not consider ontology constraints directly.

ASMOV [18] is an automated ontology mapping tool that iteratively calculates the similarity between concepts in ontologies by analyzing four features such as textual description. It then combines the measures of these four features using a weighted sum. The weights are adjusted based on some static rules. At the end of each iteration, a pruning process eliminates the invalid mappings by analyzing two semantic inconsistencies: crisscross mappings and many-to-one mappings. Due to the limited literature available we are unable to compare our approach with ASMOV in detail. What we can say is the aggregation method in ASMOV is heuristic rule based *Weighted* aggregation and only two constraints are validated for their final results.

GLUE [5] is an instance-based ontology (specifically taxonomy) mapping system using machine learning techniques. GLUE first applies statistical analysis to the available data (i.e., joint probability distribution computation), and then implements multiple learners to exploit information in concept instances and taxonomic structure of ontologies. Next, GLUE uses a probabilistic model to combine the results of different learners. Finally, GLUE adopts relaxation labeling approach to search for the mapping configuration that best satisfies the domain constraints and the common knowledge, taking into account the observed similarities. GLUE and our approach are similar in that both of us try to look for a global optimal mapping configuration when considering ontology constraints. However, we explore different approaches and different kinds of constraints. GLUE adopts relaxation labeling approach that has been applied successfully in computer vision, natural language processing and hypertext classification [16,33,2]; whereas our approach integrates the IAC neural network that has been used to model visual word recognition [29] and information retrieval tasks [35]. Furthermore, the constraints implemented in our approach are more complex than that in GLUE.

6. Summary and future work

In this paper we proposed a new adaptive ontology mapping approach, the PRIOR+. First the PRIOR+ measures three similarities of ontologies in a vector space model. Meanwhile it estimates the harmony of each similarity upon its corresponding matrix. After that the PRIOR+ adaptively filters out false mappings and aggregates multiple similarities by weighting them with their harmonies. Finally the IAC neural network will be selectively activated to find a solution that best satisfies ontology constraints. Final results will be extracted using naive extraction algorithm from the optimized mappings.

The experimental results show: harmony is a good measure to estimate the reliability of different similarities. It is good at assisting adaptively aggregating similarities and adjusting mapping strategies as well. The harmony-based adaptive aggregation method outperforms other aggregation methods. Using the IAC neural network to solve constraint satisfaction problem in ontology mapping can dramatically improve the performance of mapping results. The PRIOR+ is competitive with all top-ranked systems on the benchmark tests at OAEI campaign 2007. Notably it outperforms all systems on benchmark real cases.

Future work may include: Explore and implement more constraints such as complex axioms in OWL. Investigate which constraint is more useful than others. Assign each constraint a different weight according to its priori probability.

Acknowledgments

Our thanks go to Dr. Paul Munro for his helpful discussion on the design and implementation of the IAC neural network and Dr. Daqing He, Sergey Sosnovsky and anonymous reviewers for their useful comments and suggestion on the draft of the paper.

References

- [1] T. Berners-Lee, J. Hendler, et al., The semantic web, *Scientific American* 284 (5) (2001) 34–43.
- [2] S. Chakrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlinks, in: *Proceedings of the ACM SIGMOD Conference*, 1998.
- [3] D. Calvanese, G.D. Giacomo, et al., Ontology of integration and integration of ontologies, *Description Logics* (2001).
- [4] H.H. Do, E. Rahm, COMA—a system for flexible combination of schema matching approaches, in: *Proceedings of VLDB 2002*, 2002, pp. 610–621.
- [5] A. Doan, J. Madhavan, et al., Learning to match ontologies on the semantic web, *VLDB Journal* 12 (4) (2003) 303–319.
- [6] D. Dou, D. McDermott, et al., Ontology translation on the semantic web, *Journal on Data Semantics II* (2005) 35–57.
- [7] M. Ehrig, *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*, Springer, 2006, ISBN-038732805X.
- [8] M. Ehrig, S. Staab, QOM—quick ontology mapping, in: *Proceedings of International Semantic Web Conference 2004*, 2004, pp. 683–697.
- [9] J. Euzenat, T. Bach, et al., State of the art on ontology alignment, *Knowledge web NoE* (2004).
- [10] J. Euzenat, et al., Results of the ontology alignment evaluation initiative 2006, in: *Proceedings of ISWC 2006 Ontology Matching Workshop*, Atlanta, GA, 2006.
- [11] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient belief propagation for early vision, *International Journal of Computer Vision* 70 (1) (2006).
- [12] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag, New York, 2003.
- [13] E.A. Fox, J.A. Shaw, Combination of multiple searches, in: *Proceedings of the 2nd Text Retrieval Conference (TREC-2)*, 1994, pp. 243–252.
- [14] D. Gasevic, M. Hatala, Ontology mappings to improve learning resource search, *British Journal of Educational Technology* (2005).
- [15] E. Hovy, Combining and standardizing large-scale, practical ontologies for machine translation and other uses, in: *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain, 1998.
- [16] R. Hummel, S. Zucker, On the foundations of relaxation labeling processes, *PAMI* 5 (May (3)) (1983) 267–287.
- [17] R. Ichise, Machine learning approach for ontology mapping using multiple concept similarity measures, *ACIS-ICIS*, IEEE Computer Society (2008) 340–346.
- [18] Y. Jean-Mary, M. Kabuka, ASMOV results for OAEI 2007, in: *Proceedings of ISWC 2007 Ontology Matching Workshop*, Busan, Korea, 2007.
- [19] Y. Kalfoglou, M. Schorlemmer, Ontology mapping: the state of the art, *Knowledge Engineering Review* 18 (1) (2003) 1–31.
- [20] M. Mao, Y. Peng, The PRIOR: results for OAEI 2006, in: *Proceedings of ISWC 2006 Ontology Matching Workshop*, Atlanta, GA, 2006.
- [21] M. Mao, Y. Peng, M. Spring, A profile propagation and information retrieval based ontology mapping approach, in: *Proceedings of the 2nd SKG Conference*, 2007.
- [22] M. Mao, Ontology mapping: an information retrieval and interactive activation network based approach, in: *Proceedings of 6th International Semantic Web Conference (ISWC/ASWC 2007)*, LNCS 4825, 2007, pp. 931–935.
- [23] M. Mao, Y. Peng, M. Spring, Integrating the IAC neural network in ontology mapping, in: *Proceedings of WWW 2008*, Beijing, China, 2008, pp. 1223–1224.
- [24] M. Mao, Y. Peng, M. Spring, Neural network based constraint satisfaction in ontology mapping, in: *Proceedings of AAAI 2008*, Chicago, USA, 2008, pp. 1207–1212.
- [25] M. Mao, Y. Peng, M. Spring, A harmony based adaptive ontology mapping approach, in: *Proceeding of the 2008 International Conference on Semantic Web and Web Services (SWWS'08)*, Las Vegas, USA, 2008, pp. 336–342.
- [26] M. Mao, Y. Peng, M. Spring, Ontology mapping: as a binary classification problem, in: *Proceedings of the 3rd SKG Conference*, 2008.
- [27] C. Meilicke, H. Stuckenschmidt, Analyzing mapping extraction approaches, in: *Proceedings of ISWC 2007 Ontology Matching Workshop*, Busan, Korea, 2007.
- [28] S. Melnik, H. Garcia-Molina, et al., Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: *Proc. 18th International Conference on Data Engineering (ICDE)*, 2002.
- [29] J. McClelland, D. Rumelhart, An interactive activation model of context effects in letter perception. Part 1. An account of basic findings, *Psychological Review* 88 (1981) 375–407.
- [30] J. McClelland, D. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, The MIT Press, 1988.
- [31] N. Noy, M. Musen, The PROMPT Suite: interactive tools for ontology merging and mapping, *International Journal of Human–Computer Studies* 59 (2003) 983–1024.
- [32] N. Noy, Semantic integration: a survey of ontology-based approaches, *SIGMOD Record* 33 (4) (2004) 65–70.
- [33] L. Padro, A Hybrid Environment for Syntax-Semantic Tagging (1998).
- [34] Y. Qu, W. Hu, G. Cheng, Constructing virtual documents for ontology matching, in: *Proceedings of the 15th International Conference on World Wide Web*, 2006.
- [35] W. Ross, H. Philip, Using the cosine measure in a neural network for document retrieval, in: *Proceedings of the 14th SIGIR Conference*, Chicago, IL, 1991.
- [36] B. Schopman, S. Wang, S. Schlobach, Deriving concept mappings through instance mappings, in: *Proceedings of ASWC 2008*, LNCS 5367, Springer, 2008, pp. 122–136.
- [37] G. Stoilos, G. Stamou, S. Kollias, A string metric for ontology alignment, in: *Proceedings of the International Semantic Web Conference 2005*, 2005, pp. 623–637.
- [38] H. Stuckenschmidt, F. Harmelen, *Information Sharing on the Semantic Web*, Springer, 2005.
- [39] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, K. Wang, Using Bayesian decision for ontology mapping, *Web Semantics: Science, Services and Agents on the World Wide Web* 4 (December (4)) (2006) 243–262.
- [40] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, 1993.
- [41] P. Wang, B. Xu, LILY: the results for the ontology alignment contest OAEI 2007, in: *Proceedings of ISWC 2007 Ontology Matching Workshop*, Busan, Korea, 2007.
- [42] G. Wiederhold, Mediators in the architecture of future information systems, *IEEE Computer* 25 (3) (1992).