ERSOM: A Structural Ontology Matching Approach Using Automatically Learned Entity Representation

Chuncheng Xiang, Tingsong Jiang, Baobao Chang, Zhifang Sui
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{ccxiang,tingsong,chbb,szf}@pku.edu.cn

Abstract

As a key representation model of knowledge, ontology has been widely used in a lot of NLP related tasks, such as semantic parsing, information extraction and text mining etc. In this paper, we study the task of ontology matching, which concentrates on finding semantically related entities between different ontologies that describe the same domain, to solve the semantic heterogeneity problem. Previous works exploit different kinds of descriptions of an entity in ontology directly and separately to find the correspondences without considering the higher level correlations between the descriptions. Besides, the structural information of ontology haven't been utilized adequately for ontology matching. We propose in this paper an ontology matching approach, named ERSOM, which mainly includes an unsupervised representation learning method based on the deep neural networks to learn the general representation of the entities and an iterative similarity propagation method that takes advantage of more abundant structure information of the ontology to discover more mappings. The experimental results on the datasets from Ontology Alignment Evaluation Initiative (OAEI[1]) show that ERSOM achieves a competitive performance compared to the state-of-the-art ontology matching systems.

[1]The OAEI is an international initiative organizing annual campaigns for evaluating ontology matching systems. All of the ontologies provided by OAEI are described in OWL-DL language, and like most of the other participates our ERSOM also manages the OWL ontology in its current version. OAEI: `http://oaei.ontologymatching.org/`

1 Introductions

In the recent years, it becomes evident that one of the most important directions of improvement in natural language processing (NLP) tasks, like word sense disambiguation, coreference resolution, relation extraction, and other tasks related to knowledge extraction, is by exploiting semantics resources (Bryl et al., 2010). Nowadays, the Semantic Web made available a large amount of logically encoded information (e.g. ontologies, RDF(S)-data, linked data, etc.), which constitutes a valuable source of semantics. However, extending the state-of-the-art natural language applications to use these resources is not a trivial task mainly due to the heterogeneity and the ambiguity of the schemes adopted by the different resources of the Semantic Web. How to utilize these resources in NLP tasks comprehensively rather than choose just one of them has attracted much attention in recent years.

An effective solution to the ontology heterogeneity problem is ontology matching (Euzenat et al., 2007; Shvaiko and Euzenat, 2013), whose main task is to establish semantic correspondences between entities (i.e., classes, properties or instances) from different ontologies.

Ontology matching is usually done by measuring the similarity between two entities from two different ontologies. To effectively calculate the similarities, almost all types of descriptions of an entity should be used. In previous works, given the different nature of different kinds of descriptions, similarities are normally measured separately with different methods and then aggregated with some kind of combination strategy to compute the final similarity score. For example, Mao et al. (2010) defined three single similarities (i.e., *Name similarity*, *Profile similarity* and *Structural similarity*) based on the descriptions of an entity, then they employed a harmony-based method to aggregate

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2419–2429,
Lisbon, Portugal, 17-21 September 2015.

the single similarities to get a final similarity for extracting the final mappings. However, treating different kinds of descriptions of an entity separately suffers from two limitations. First, it limits the capacity of modeling the interactions between different descriptions. For example, entity's label is always a specific substitution of its ID; entity's comment is a semantic definition for its ID; a class can be characterized with its related properties, and a property is usually restricted by its domain and range. These potential correlations of the descriptions are very important to measure the similarity between entities since they can be treated as some potential features describing an entity. Second, it is difficult to estimate how many and which kind of single similarities are needed for an aggregation method to get a satisfactory result.

On the other hand, in order to find more mappings, many structural ontology matching methods are proposed. To the best of our knowledge, previous structural methods are either local methods (Le et al., 2004; Sunna and Cruz, 2007) or global (i.e. iterative) methods but only use part of the structure information of the ontology (Li et al., 2009; Ngo and Bellahsene, 2012). For example, the ontology matching system YAM++ (Ngo and Bellahsene, 2012) utilizes a global structural method but it only uses the structure information of classes and properties to create the propagation graph to find mappings between classes and properties. A large amount of instances and their relations to classes and properties in the ontology haven't been exploited in this system.

To overcome the existing limitations, we propose in this paper a representation learning method to capture the interactions among entity's descriptions; then we present our global structural method which exploits more abundant structure information of the ontology. We summarize our contributions as follows.

- We propose to use the deep neural network model to learn the high-level abstract representations of classes and properties from their descriptions to acquire the potential correlations for the computing of the similarities between classes and properties. Moreover, there is no need to select and aggregate different single similarities in the similarity computation.
- We propose a global similarity propagation method that utilizes more abundant structure information including all kinds of entities and their relations in the ontology, to find more mappings.

To evaluate the effectiveness of our approach, we conduct experiments on the public datasets from OAEI campaign (We select the OAEI data sets mainly because evaluation metrics have been well defined on these data sets and comparision can be easily made). The experimental results show that our matching approach can achieve a competitive matching performance compared to the state-of-the-art systems.

2 Problem Statement

Ontology is a formal, explicit specification of a shared conceptualization in terms of classes, properties and relations (Euzenat et al., 2004). The process of ontology matching is to find mappings (or correspondences) between entities (classes, properties or individuals) from two ontologies. A mapping is defined as a four-tuple as written in Eq.(1), where e^1 and e^2 represent the entity in ontology O^1 and O^2 respectively, r is a kind of matching relation (e.g., equivalent, subsume) and $k \rightarrow [0, 1]$ is the degree of confidence of matching relation between e^1 and e^2 (Mao et al., 2010).

$$m =< e^1, e^2, r, k > \quad (1)$$

Similar with most of the OAEI systems (Li et al., 2009; Ngo and Bellahsene, 2012; Cheatham and Hitzler, 2013b), we focus on discovering only equivalent mappings between classes and properties with cardinality 1:1. That is, one class (property) in ontology O^1 can be matched to at most one class (property) in ontology O^2 and vise versa.

3 ERSOM: Entity Representation and Structure based Ontology Matching

In this paper, we propose a structural ontology matching approach using automatically learned entity representation, which we call ERSOM. Fig.1 shows the architecture of our approach. The details of its major modules are given in the following sections.

3.1 Learning the representation of entity

In this section, we present how to learn the high-level abstract representations for ontology entities. The motivations are: 1) we regard different kinds of the descriptions of an entity as a whole to avoid

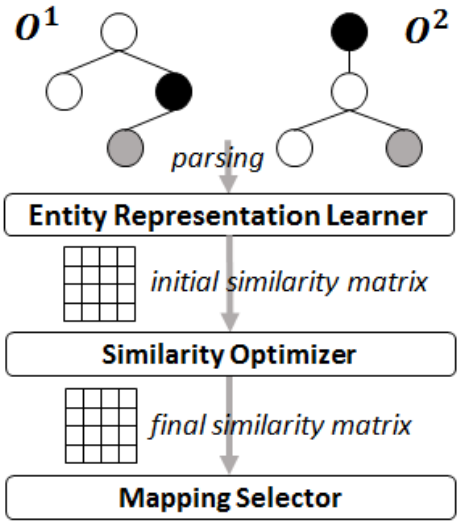

Figure 1: The architecture of ERSOM. Given the two to-be-matched ontologies, we first extract the descriptive information for each entity, then learn the entity's abstract representation based on its descriptions, and finally utilize the representations to compute the similarities between entities to initialize the similarity propagation method to find final mappings.

separatively calculating the similarities and aggregating them later with a combination method; 2) the learned representation can not only express the meaning of the original descriptions of an entity but also captures the interactions among different descriptions.

3.1.1 Creating term vector for entity

We first generate a combination of the entity's descriptions (CDs for short) and then create a term vector for each entity. In particular, the CDs of a class = the class's ID + label + comments + its properties' descriptions + its instances' descriptions. The CDs of a property = the property's ID + label + its domain + its range (or its textual value when the property is a datatype property). And the CDs of an instance = the instance's ID + label + its properties' values. A binary term vector is created for each entity with the pre-processing that consists of tokenizing, removing stop words, stemming and deleting superfluous words. In the binary term vectors, element 1 and 0 refer to the existence and inexistence of a specific word, respectively.

3.1.2 Learning entity representations

In the ontology matching area, training data usually refers to a pair of ontologies with correct mappings created by domain experts between their entities. The acquisition of such dataset is time-consuming and laborious. We state in this section how to learn the abstract representations for entities from their binary term vectors with an unsupervised way. The deep neural network (DNN) (Hinton et al., 2006; Bengio et al., 2007) is a multilayer learning model. It is mainly used for learning the high-level abstract representations of original input data. Given the generalization and the abstraction introduced in the representation learning procedure, DNN allows us to better model the interactions among different kinds of input features, and measure the similarity at a more general level. Inspired by the work in (Hinton, 2007; Bengio et al., 2012; He et al., 2013; Cui et al., 2014), we use auto-encoder (Bourlard and Kamp, 1988; Hinton and Zemel, 1994) to learn the representations for classes and properties. The auto-encoder is one of the neural network variants that can automatically discover interesting abstractions to represent an unlabeled dataset.

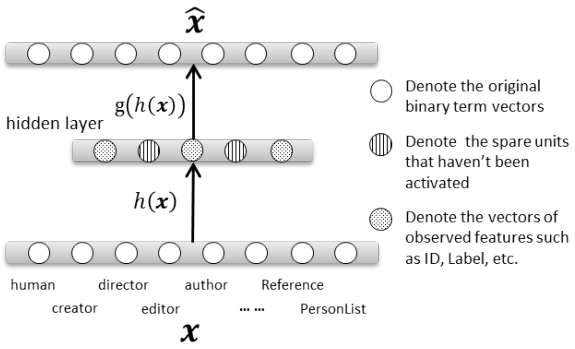

Figure 2: Unsupervised representation learning.

As shown in Fig.2, the input to the auto-encoder is denoted as x, which indicates a binary term vector of a class or a property. Auto-encoder tries to learn an approximation to the identity function $h(x)$, so as to output $\hat{x}$ that is similar to x. More specifically, the auto-encoder consists of an encoding process $h(x) = f(Wx + b_1)$ and a decoding process $g(h(x)) = f(W^T h(x) + b_2)$, where f is a activation function like the sigmoid function $f(x) = 1/(1 + exp(-x))$, W is the weight matrix and b is the bias term. The goal of the auto-encoder is to minimize the reconstruction error $L(x, g(h(x))) = |x - g(h(x))|^2$, thus retaining

maximum information. Through the combination and transformation, auto-encoder learns the abstract representation $h(x)$ of the input binary term vector. The representation is a real vector with values between 0 and 1.

In consideration of the large number of units in the hidden layer (as marked in Fig.2), a sparsity constraint is imposed on the hidden units to hold the capacity to discover interesting structure in the data. We use sparse auto-encoder (Coates et al., 2011) to learn the correlations between descriptions from their binary term vectors. The sparse auto-encoder attempts to minimize the following loss function:

$$J(W,b) = \sum_{i=1}^{m} \left\| x^{(i)} - \widehat{x}^{(i)} \right\|^2 + \lambda \left(\|W_1\|_F + \|W_2\|_F \right) + \beta \sum_{h \in H} KL\left(\rho \| \widehat{\rho}_h\right) \tag{2}$$

where $x^{(i)}$ is the binary term vector of the ith entity (a class or a property), $\widehat{x}^{(i)}$ is the reconstruction of $x^{(i)}$, λ is a regularization parameter, original $\| \cdot \|_F$ is the Frobenius norm, β controls the weight of the sparsity penalty term, H is the set of hidden units, and ρ is the sparsity parameter. Formally, $KL\left(\rho \| \widehat{\rho}_h\right) = \rho \log \frac{\rho}{\widehat{y}_h} + (1-\rho) \log \frac{1-\rho}{1-\widehat{y}_h}$ is the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\widehat{y}_h$.

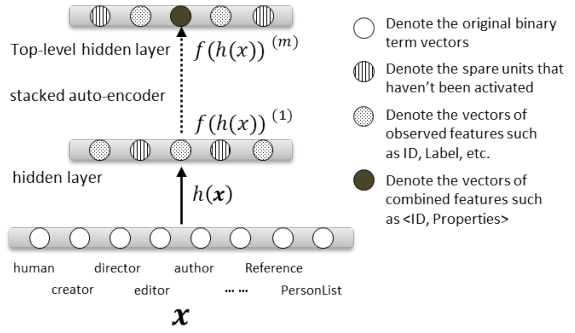

Figure 3: Learning higher level representations.

3.1.3 Learning higher level representations

The auto-encoder which only has one hidden layer may not be enough to learn the complex interactions between input features. Inspired by the work of Vincent et al. (2010) and He et al. (2013), we build multi-layer model to learn more abstract entity representations. To achieve this, we repeatedly stack new sparse auto-encoder on top of the previously learned $h(x)$ (i.e., the higher level representations are formed by combination of lower level representations). This model is called Stacked Auto-Encoder (SAE) by Bengio et al. (2007). In this way, when we input the binary term vector to the network, we can get its abstract representations in different levels. In other words, with the layer-by-layer learning, we obtain different levels of representations. The top-level representation, which models the final interactions of the original descriptions, can be used to measure the similarity between classes and properties.

The prototype of Stacked Auto-Encoder (SAE) is shown in Fig.3, where $f(h(x))^{(m)}$ denotes the final representation learned by the top-level hidden layer and superscript m means the SAE consists of m sparse auto-encoders.

3.2 Optimizing with the ontology structure

The above method can only consider the local descriptions (such as ID, label and comments etc.) of entities in ontology. According to the study in (Melnik et al., 2002), we present our structural method or called Similarity Propagation (SP) method, which exploits more abundant structure information of the ontology to discover more mappings globally. The intuition behind the propagation idea is that the similarity between two entities is not only about their neighbor entities, but it is about all other entities (neighbor entities' neighbor entities) in the ontologies. This idea has also been used in the ontology matching systems RiMOM(Li et al., 2009) and YAM++(Ngo and Bellahsene, 2012) in order to find mappings between classes and properties. But the nodes in their propagation graph are just limited to class pairs and property pairs, and the propagation edges are transformed from relations between two classes, two properties or a class and a property. The difference of our SP method is that we consider the instances and its relations with classes and properties when creating the propagation graph even if we also only find mappings between classes and properties. This is because (1) the similar degree of two classes will be increased if they have some of similar instances; (2) the similar degree of two properties will be increased if the instances that own these properties are similar. The propagation graph in our SP method will be much more complete compared with the previous ones.

Algorithm 1 presents the procedures of our SP method. In the first two steps of it, we represent each to-be-matched ontology to a Directed Labeled Graph (DLG). Each edge in the DLG has format (s, p, o), where s and o are the source and target nodes (each node represents a class, a property or an instance), and the edge's label p comes from one of the seven ontology relations including *HasSubClass*, *HasSubProperty*, *HasObjectProperty*, *HasDataProperty*, *HasInstance*, *HasDomain*, *HasRange*. Then we create a Pairwise Connectivity Graph (PCG) from two DLGs by merging edges having the same labels.

Algorithm 1: Our SP Algorithm

Input: The to-be-matched ontologies, O_R and O_T; The initial similarity matrix, M_0; The edges' weight matrix, W;

Output: The updated similarity matrix, M_1;

$DLG_1 \leftarrow Transform(O_R)$;
$DLG_2 \leftarrow Transform(O_T)$;
$PCG \leftarrow Merge(DLG_1, DLG_2)$;
$IPG \leftarrow Initiate(PCG, M_0, W)$;
$M_1 \leftarrow Propagation(IPG, Normalized)$;

In the fourth step of Algorithm 1, for a PCG, we assign weight values to edges as the inverse of the number of out-linking relationships of its source node (Melnik et al., 2002). For the nodes that consist of two classes or two properties, we assign them values calculated with the cosine similarity between their representations learned in section 2.1.3. For the node consisting of two instances, the similarity value assigned to it is measured with the ScaledLevenstein[2] between the IDs of instances. In this way, we construct an Induced Propagation Graph (IPG) on which the propagation algorithm will run iteratively. Let $\sigma(x, y)$ denotes the similarity score between entities x and y for node (x, y) in the IPG. At the $(i+1)$th iteration, the similarity score is updated as follows:

$$\sigma^{i+1} = \frac{1}{z}\left(\sigma^0 + \sigma^i + \varphi(\sigma^0 + \sigma^i)\right), \quad (3)$$

$$\varphi\left(\sigma^0 + \sigma^i\right) = \sum_{j=1}^{m}\left(\sigma^0 + \sigma_j^i\right) w_j \quad (4)$$

[2] `http://sourceforge.net/projects/secondstring/`. ScaledLevenstein is a good method for computing string similarity (Cheatham and Hitzler, 2013a), of course, it can be replaced by other methods.

where z is the normalization factor defined as $z = \max_{x' \in IPG}\left(\sigma^{i+1}\right)$. σ^0 and σ^i are similarities at the initial time and ith iterations, respectively. $\varphi()$ is the function to compute the similarities propagated from the adjacent node σ_j^i connected to node (x, y) in the $(i+1)$th iteration. And w_j is the weight of edge between the node (x, y) and its jth neighboring node.

During each iteration in the final step of Algorithm 1, only the similarity value between two entities in the node will be updated. At the end of each iteration, all similarity values are normalized by a *Normalized* function to all in range $[0, 1]$. The iteration stops after a predefined number of steps.

3.3 Mapping selection

Similar with the work in (Wang and Xu, 2007; Huang et al., 2007; Ngo and Bellahsene, 2012), we use the Stable Marriage (SM) algorithm (Melnik et al., 2002) to choose the 1:1 mappings from the M rows and N columns similarity matrix, where M and N is the number of classes and properties in ontologies O_1 and O_2, respectively. In addition, before we run the SM algorithm we set the value of cell $[i, j]$ of the similarity matrix to zero if i and j correspond to different types of entities. Thus, we remove lots of redundant data and only find the mappings between classes or properties.

4 Experiments

4.1 Data sets and evaluation criteria

The annual OAEI campaign is an authoritative contest in the area of ontology matching, we choose the data from OAEI in our experiments, because the evaluation metrics have been well defined and the comparision can be easily made. We observe strong structure similarities lies between OAEI ontologies and ontologies used in NLP tasks, such as WordNet and HowNet for WSD (Li et al., 1995; Agirre et al., 2009), and Freebase, YAGO, and knowledge graph for IE, text mining and QA (Yao and Van Durme, 2014; Yao et al., 2014), both describe entities and their relations with class, properties and instances.

Development dataset: the Standard Benchmark 2012 dataset that OAEI provides for developers to test their system before participating in the competition is used as the development dataset in our experiments. This dataset contains one reference ontology and 109 target ontologies. We use

this dataset to test various values for the parameters in our ERSOM and apply the best ones to the experiments on the testing datasets.

Testing dataset: (1) the Benchmark-Biblio 2012 dataset which contains one reference ontology and 94 target ontologies; (2) the Benchmark-Biblioc 2013 dataset which has five sub-datasets and there are one reference ontology and 93 target ontologies in each sub-dataset. We use these two datasets to evaluate the performance of our ERSOM approach.

In the matching scenario, each target ontology should be mapped to the reference ontology. We followed the standard evaluation criteria from the OAEI, calculating the precision, recall and f-measure over each test. The version computed here is the harmonic mean of precision and recall.

4.2 Experimental design and results

4.2.1 Evaluation for representation learning

We first use Jena[3] parsing the ontologies and extract descriptions for entities according to the description in section 2.1.1, then we create a vocabulary based on the dataset and denote each class and property as a binary term vector. We apply the L-BFGS algorithm (Ngiam et al., 2011) to train the stacked auto-encoder described in section 2.1.3. The size of the input layer is equals to the length of the vocabulary created from the dataset. We fix the parameters $\lambda = 1e-4$, $\beta = 3$ and $\rho = 0.25$ in Eq.2, and set the size of the first and second hidden layer of the stacked auto-encoder to 200 and 100, respectively, by experience. The number of iterations of the L-BFGS algorithm is set to 500. We use the learned representations to measure the similarities between classes and properties and apply the strategy presented in section 2.3 to extract final mappings. The matching results of our Unsupervised Representation Learning (URL) method on the development dataset and testing datasets are shown in Table 1 and Table 2, respectively.

Method	Prec.	Rec.	F-m.
TV	0.841	0.715	0.748
URL(1)	0.819	0.822	0.820
URL(2)	0.843	0.846	0.844

Table 1: Representation learning on dev. dataset.

In Table 1, TV denotes the matcher in which the similarities are calculated between binary term vectors of classes and properties by using cosine measure. URL(i), where $i \in \{1, 2\}$, represents that we use the representations learned by the ith hidden layer of the stacked auto-encoder to measure the similarities between classes and properties to find mappings. Table 1 shows that on the development dataset, the F-measure of TV is 0.748 and it is improved 9.6% and 12.8% when we use the representations learned by the single-layer and double-layer auto-encoder to find the mappings, respectively. It illustrates that we have learned some useful information from the term vectors, which can be explained as the interactions between descriptions of entities. From the last two rows in Table 1, we can find that the F-measure improved by 2.9% when we use the representations learned by the second hidden layer (i.e., URL(2)) to measure the similarities.

	Benchmark-Biblio 2012			Benchmark-Biblioc 2013		
	Prec.	***Rec.***	***F-m.***	***Prec.***	***Rec.***	***F-m.***
TV	0.870	0.719	0.761	0.865	0.715	0.757
URL(1)	0.805	0.808	0.806	0.780	0.783	0.786
URL(2)	0.814	0.817	0.815	0.787	0.790	0.793

Table 2: Representation learning on test datasets.

From Table 2 we can see that the F-measures are increased on both of the testing datasets when we use the learned representations to measure similarities between classes and properties compared with using term vectors, but the amount of improvements are less than that on the development dataset. This is because we estimate the parameters of the representation learning model on the development dataset and then apply them on the test tasks directly. The precision is reduced when we use URL method, this may be due to the learned representations of entities are too general. In addition, in the parameter adjustment process, we try to make the F value maximization, but not to care about mapping precision. This is because we usually compare the performance of the systems based on their matching F values.

4.2.2 Comparison with aggregation methods

Aggregating different similarities is pervasive in ontology matching systems that contain multiple single matchers, for example, Falcon-AO(Qu et al., 2006), RiMOM(Li et al., 2009), YAM++(Ngo and Bellahsene, 2012), etc. Since our representation learning method also combines all descriptions of an entity together in an unsu-

[3] https://jena.apache.org/

pervised way, we compare it with previous unsupervised aggregation strategies, that is, Max, Average, Sigmoid, Weighted(Cruz et al., 2010) and Harmony(Mao et al., 2008, 2010). As the work in (Mao et al., 2010; Ngo and Bellahsene, 2012), we first define three context profiles including individual profile, semantic profile and external profile for each class and property (this equivalent to divide the collection of descriptions of a class or a property into three different parts). Then we apply a vector space model with TFIDF weighting scheme and cosine similarity measure to compute similarity scores between profiles. And finally, we aggregate these three single similarities using different aggregation methods.

	Dev.	**Tes.1**	**Tes.2**
Individual Profile	0.668	0.612	0.611
Semantic Profile	0.434	0.472	0.477
External Profile	0.222	0.224	0.224
MAX	0.739	0.705	0.712
Average	0.792	0.786	0.786
Sigmoid	0.763	0.753	0.757
Weighted	0.755	0.716	0.728
Harmony	0.794	0.789	0.785
URL	**0.844**	**0.815**	**0.793**

Table 3: Comparison with aggregation methods.

Table 3 shows the F-measure of the single matchers and aggregation methods on the development dataset (Dev. for short) and two testing datasets (i.e., Tes.1 and Tes.2, which refer to the Benchmark-Biblio 2012 dataset and the Benchmark-Biblioc 2013 dataset, respectively). First, the performance of single matcher is poor, the highest F-measures are 0.668, 0.612 and 0.611 on the datasets Dev., Tes.1 and Tes.2, respectively. And when we use external profile to calculate the similarities, the F-measures are reduced to 22%. Second, the performance is dramatically boosted by aggregation methods and they all achieve F-measures higher than 0.7, so the aggregation methods are very effective in improving the performance of mapping approaches that rely on measuring multiple similarities. And finally, our Unsupervised Representation Learning (URL) method holds the highest F-measure both on the development dataset and on the testing datasets.

4.2.3 Evaluation for our structural method

In this experiment, we compare our Similarity Propagation (SP) method to other structure based methods, that is, ADJACENTS and ASCOPATH in (Le et al., 2004); DSI and SSC in (Sunna and Cruz, 2007); Li's SP (Li et al., 2009) and Ngo's SP (Ngo and Bellahsene, 2012). We first use entity's ID to compute the similarity between classes and properties to provide an unified initial similarity matrix as input (or initialization) for our SP and other structural methods. Then, a new similarity matrix will be created and updated by considering the initial similarities and different structure information. And finally, we extract the mappings from the newly created similarity matrix with the strategy described in section 2.3.

	Dev.	**Tes.1**	**Tes.2**
Initial Matcher	0.616	0.524	0.523
ADJACENTS	0.622	0.569	0.570
ASCOPATH	0.604	0.540	0.552
DSI	0.641	0.576	0.575
SSC	0.642	0.569	0.568
Li's SP	0.747	0.769	0.772
Ngo's SP	0.751	0.768	0.764
Our SP	**0.810**	**0.834**	**0.839**
Our SP with URL	**0.903**	**0.865**	**0.866**

Table 4: Comparison with structural methods.

In ADJACENTS method, the parameter W_k, where $k \in \{1, 2, 3\}$, is set to 1/3. The parameter MCP in the methods DSI and SSC is set to 0.75 as reported in their work. The iterative times to SP algorithm are fixed to 50. Table 4 reports the matching F-measures of these structure based methods on the development dataset (Dev. for short) and testing datasets (Tes.1, Tes.2 for short).

From table 4 we can see that the local-structure based methods (i.e., ADJACENTS, ASCOPATH, DSI and SSC) provide low matching quality. It means that these methods did not discover enough additional correct mappings or even find some incorrect mappings. For example, the F-measure even reduced on the development dataset when use ASCOPATH method. This is because if two entities don't have any common entity in their ancestors, their similarity is equal to 0. Whereas, Li's SP and Ngo's SP are global-structure based methods, and they seem to work well. The F-measure has even improved by 21.9% when using the Ngo's SP compared with the initial matcher. This

is because in the SP method, the similarity score of a pair of entities depends on not only their current status but also on the status of the other pairs. That explains why SP outperforms all other local based structural methods. In our SP, all instances and their relations to other entities in ontology are exploited to help find the mappings between classes and properties, therefore the matching quality is distinctly improved.

The last two rows of Table 4 shows that when we use the learned representations to create the initial similarity matrix to initialize our SP method, the matching quality is significantly improved. For example, the F-measure is improved from 0.810 to 0.903 on the development dataset. This illustrates that the initialization step is very important to the SP method.

4.2.4 Comparison with other ontology matching systems

We compare our ontology matching approach, called ERSOM, with other multi-strategy matching systems on the testing datasets. Fig.4 lists the results of top five matching systems according to their F-measures on the Benchmark-Biblio 2012 dataset and Benchmark-Biblioc 2013 dataset.

As shown in Fig.4, ERSOM outperforms most of the participates except the systems YAM++ and CroMatcher whose F-measures are 0.89 and 0.88 in 2013, respectively. CroMatcher achieves the same level of recall as YAM++ but with consistently lower precision (Grau et al., 2014). Unlike MapSSS, our approach does not use any external resources such as Google queries in its current version. In YAM++ approach, the gold standard datasets that taken from Benchmark dataset published in OAEI 2009 are used to generate training data to train a decision tree classifier. And in the classifying phase, each pair of elements from two to-be-matched ontologies is predicted as matched or not according to its attributes. However, ERSOM is an unsupervised approach, but it does not exclude using external resources and training data to help learning the representations of entities and provide the initial similarity matrix for the SP method to further improve the performance.

5 Related work

There are many studies on Ontology Matching (Euzenat et al., 2007; Shvaiko and Euzenat, 2013). Currently, almost all ontology matching systems exploit various kinds of information provided in

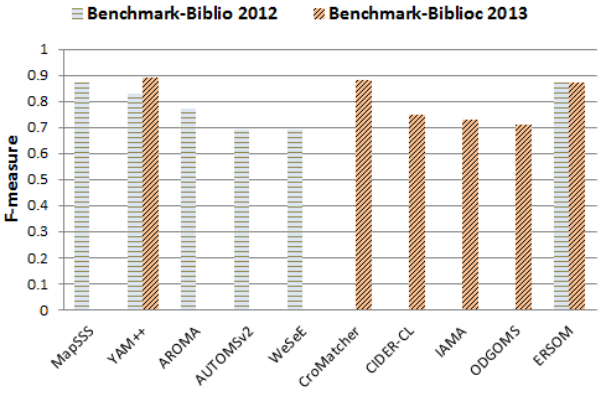

Figure 4: Comparison with other OAEI systems.

ontologies to get better performance. To aggregate the different similarity matrixes, various approaches have been proposed.

Pirró and Talia (2010) is a generic schema matching system. It exploits *Max*, *Min*, *Average* and *Weighted strategies* for the combination. The weighted method assigns a relative weight to each similarity matrix, and calculates a weighted sum of similarity for all similarity matrixes. The *Average method* is a special case of *Weighted*, which considers each similarity matrix equally important in the combination. *Max* and *Min* are two extreme cases that return the highest and lowest similarities in all similarity matrixes respectively. Ji et al. (2011) use the Ordered Weighted Average (OWA) to combine different matchers. It is a kind of ontology-independent combination method which can assign weights to the entity level, i.e., it use a specific ordered position rather than a weight associated with a specific similarity matrix to aggregate multiple matchers. Jean-Mary et al. (2009) combines different matchers by using a weighted sum strategy that adjusts weights empirically, or based on some static rules. This approach cannot automatically combine different matchers in various matching tasks.

There are several works which exploit the supervised machine learning techniques for ontology matching. Eckert et al. (2009), string-based, linguistic and structural measures (in total 23 features) were used as input to train a SVM classifier to align ontologies. CSR (Classification-based learning of Subsumption Relations) is a generic method for automatic ontology matching between concepts based on supervised machine learning (Spiliopoulos et al., 2010). It specifically focusses on discovering subsumption correspon-

dences. SMB (Schema Matcher Boosting) is an approach to combining matchers into ensembles (Gal, 2011). It is based on a machine learning technique called boosting, that is able to select (presumably the most appropriate) matchers that participate in an ensemble.

The difference of our work is that the textual descriptions are not been directly used to measure the similarities between entities. We learn a representation for each ontology entity in an unsupervised way to capture the interactions among the descriptions, which avoid the problem of selecting and aggregating different individual similarities.

6 Conclusions

The successful ontology matching is very important to link heterogeneous ontologies for NLP. In this paper, we have proposed an ontology matching approach, ERSOM, which describes the classes and properties in ontology with abstract representations learned from their descriptions and improves the overall matching quality using an iterative Similarity Propagation (SP) method based on more abundant structure information. Experimental results on the datasets from OAEI demonstrate that our approach performs better than most of the participants and achieves a competitive performance. In our future work, we will consider to use the ontology matching approach to the matching between different NLP-oriented ontologies such as wordnet, Freebase, YAGO, etc.

Acknowledgments

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The corresponding authors of this paper are Baobao Chang and Zhifang Sui.

References

Agirre, E., De Lacalle, O. L., Soroa, A., and Fakultatea, I. (2009). Knowledge-based wsd and specific domains: Performing better than generic supervised wsd. In *IJCAI*, pages 1501–1506. Citeseer.

Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR abs/1206.5538*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.

Bryl, V., Giuliano, C., Serafini, L., and Tymoshenko, K. (2010). Supporting natural language processing with background knowledge: Coreference resolution case. In *The Semantic Web–ISWC 2010*, pages 80–95. Springer.

Cheatham, M. and Hitzler, P. (2013a). String similarity metrics for ontology alignment. In *The Semantic Web–ISWC 2013*, pages 294–309. Springer.

Cheatham, M. and Hitzler, P. (2013b). Stringsauto and mapsss results for oaei 2013. *Ontology Matching*, page 146.

Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223.

Cruz, I. F., Stroe, C., Caci, M., Caimi, F., Palmonari, M., Antonelli, F. P., and Keles, U. C. (2010). Using agreementmaker to align ontologies for oaei 2010. In *ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings*, volume 689, pages 118–125.

Cui, L., Zhang, D., Liu, S., Chen, Q., Li, M., Zhou, M., and Yang, M. (2014). Learning topic representation for smt with neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 133–143.

Eckert, K., Meilicke, C., and Stuckenschmidt, H. (2009). Improving ontology matching using meta-level learning. In *The Semantic Web: Research and Applications*, pages 158–172. Springer.

Euzenat, J., Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*. Springer.

Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., De Bo, J., Dieng-Kuntz, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., et al.

(2004). State of the art on ontology alignment. *Knowledge Web Deliverable D*, 2:2–3.

Gal, A. (2011). Uncertain schema matching. *Synthesis Lectures on Data Management*, 3(1):1–97.

Grau, B. C., Dragisic, Z., Eckert, K., Euzenat, J., Ferrara, A., Granada, R., Ivanova, V., Jiménez-Ruiz, E., Kempf, A., Lambrix, P., et al. (2014). Results of the ontology alignment evaluation initiative 2013. In *International Workshop on Ontology Matching, collocated with the 12th International Semantic Web Conference-ISWC 2013*, pages pp–61.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *ACL (2)*, pages 30–34.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3.

Huang, J., Dang, J., Vidal, J. M., and Huhns, M. N. (2007). Ontology matching using an artificial neural network to learn weights. In *Proc. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa-07), India*.

Jean-Mary, Y. R., Shironoshita, E. P., and Kabuka, M. R. (2009). Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235–251.

Ji, Q., Haase, P., and Qi, G. (2011). Combination of similarity measures in ontology matching using the owa operator. In *Recent Developments in the Ordered Weighted Averaging Operators: Theory and Practice*, pages 281–295. Springer.

Le, B. T., Dieng-Kuntz, R., and Gandon, F. (2004). On ontology matching problems. *ICEIS (4)*, pages 236–243.

Li, J., Tang, J., Li, Y., and Luo, Q. (2009). Rimom: A dynamic multistrategy ontology alignment framework. *Knowledge and Data Engineering, IEEE Transactions on*, 21(8):1218–1232.

Li, X., Szpakowicz, S., and Matwin, S. (1995). A wordnet-based algorithm for word sense disambiguation. In *IJCAI*, volume 95, pages 1368–1374. Citeseer.

Mao, M., Peng, Y., and Spring, M. (2008). A harmony based adaptive ontology mapping approach. In *SWWS*, pages 336–342.

Mao, M., Peng, Y., and Spring, M. (2010). An adaptive ontology mapping approach with neural network based constraint satisfaction. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):14–25.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE.

Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q. V., and Ng, A. Y. (2011). On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272.

Ngo, D. and Bellahsene, Z. (2012). Yam++: a multi-strategy based approach for ontology matching task. In *Knowledge Engineering and Knowledge Management*, pages 421–425. Springer.

Pirró, G. and Talia, D. (2010). Ufome: An ontology mapping system with strategy prediction capabilities. *Data & Knowledge Engineering*, 69(5):444–471.

Qu, Y., Hu, W., and Cheng, G. (2006). Constructing virtual documents for ontology matching. In *Proceedings of the 15th international conference on World Wide Web*, pages 23–31. ACM.

Shvaiko, P. and Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176.

Spiliopoulos, V., Vouros, G. A., and Karkaletsis, V. (2010). On the discovery of subsumption relations for the alignment of ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(1):69–88.

Sunna, W. and Cruz, I. F. (2007). Structure-based methods to enhance geospatial ontology alignment. In *GeoSpatial Semantics*, pages 82–97. Springer.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising

autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Wang, P. and Xu, B. (2007). Lily: the results for the ontology alignment contest oaei 2007. In *Proceedings of the Second International Workshop on Ontology Matching*, pages 179–187. Citeseer.

Yao, X., Berant, J., and Van Durme, B. (2014). Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82.

Yao, X. and Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.