Ontology Alignment using Machine Learning Techniques

Bita Shadgar[a,*], Azadeh Haratian Nejhadi[a] and Alireza Osareh[a]
[a]*Computer Engineering Department, Shahid Chamran University, Ahvaz, Iran.*

Abstract. In semantic web, ontology plays an important role to provide formal definitions of concepts and relationships. Therefore, communicating similar ontologies becomes essential to provide ontologies interpretability and extendibility. Thus, it is inevitable to have similar but not the same ontologies in a particular domain, since there might be several definitions for a given concept. This paper presents a method to combine similarity measures of different categories without having ontology instances or any user feedbacks towards aligning two given ontologies. To align different ontologies efficiently, K Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT) and AdaBoost classifiers are investigated. Each classifier is optimized based on the lower cost and better classification rate. Experimental results demonstrate that the F-measure criterion improves to 99% using feature selection and combination of AdaBoost and DT classifiers, which is highly comparable, and outperforms the previous reported F-measures.

Keywords: Ontology Alignment, Support Vector Machine, Decision Tree, AdaBoost, K-Nearest Neighbor

1. Introduction

Managing distributed information across the web is going to be a difficult challenge. Ontologies have been a solution of this problem. However, reusing the existing ontologies has been considered recently. Different attitudes of ontology designers cause several similar ontologies in every particular domain [1, 2]. It is unlikely to find two ontologies describing a same thing (concept) with a fully complete match. This makes communication and interoperability difficult or impossible [3]. Ontology alignment overcomes these difficulties by exploring a map between similar entities that refer to the same concept in two different ontologies [4,5]. Therefore the importance of ontology alignment methods becomes more non-trivial, considering the fact that communication and interoperability are necessary for wide variety of areas. These areas include web service integration, agent communication, information retrieval from heterogeneous multimedia databases [6], learning resource management systems [1, 2], improving web-based searching [7], business processes management systems [8] and so on.

Ontology alignment process usually comprises six steps: (1) feature engineering, (2) search step selection, (3) similarity computation, (4) similarity aggregation, (5) interpretation and (6) iteration [9]. Manual solution of this process is usually time consuming and expensive. Therefore, having an automated solution becomes necessary. The current ontology alignment has applied automatic techniques in two parts: (1) training and generating the model and (2) the classification process [8]. ML techniques help to perform the last three steps of above more efficiently. Different well-known categories of similarity methods that are used to measure the similarity of two ontologies include: string, linguistic, structural and instance based methods. Each similarity measure is considered as a feature of the input sample, thus it is important to select effective similarity measures (features) from different categories (steps (1) and (2)).

There are several works which have already exploit ML techniques towards ontology alignment. In [9] a multi-strategy learning was used to obtain similar instances of hierarchies to extract similar concepts using Naive Bayes (NB) technique. In [10], following a parameter optimization process on SVM, DT

[*]Bita.Shadgar@scu.ac.ir.

and neural networks (NN) classifiers, an initial alignment was carried out. Then the user's feedback was considered to improve the overall performance.

In [11], some string and linguistic (using WordNet) measures were utilized as input features. It then used CART, NN and DT based classifiers to align ontologies. In [12], string, linguistic and structural measures (in total 23 features) were used to obtain the dataset of pair entities, and then the SVM algorithm was applied to classify the dataset samples. The idea of [13] is taken from [9] with an almost similar dataset. This work computes 10 similarity measures from string, linguistic and instance based methods. The DT and Naive Bayes were applied to classify the input samples. While [14] applied SVM classifier on 27 similarity measures from string, linguistic, structural and instance based methods. However, paper [15] presented a method for improving alignment results by not choosing a specific alignment method but applying ML techniques on an ensemble of alignment methods.

Some research works [7,8,16,17] have applied ontology instances in conjunction with the instance based methods of similarity. However, providing the ontology instances is expensive. Therefore, this work does not apply instance based methods.

Other studies use rule sets, RDF graph analysis, data mining and ML techniques to aggregate similarity measures of each individual category [18]. This paper for the first time composed different individual similarity metrics (features) of string, linguistic and structural based categories into one input sample. As each individual similarity measure is able to determine partial similarity of the whole feature space, considering all the measures simultaneously will probably achieve higher classification accuracy.

The ensemble method is an active research area which gives better performance than a single classifier [19]. Some research works have shown that using a single classifier performing well may not be the optimal choice [20]. It may lose potentially valuable information contained in other less accurate classifiers. Thus ensemble approach is proposed as a solution to combine several less accurate classifiers in this work.

Section 2 presents the most well-known and effective similarity measures which are utilized in this work. The exploited classifiers are briefly introduced in section 3. The proposed alignment method has been modeled and discussed in section 4. Section 5 evaluates the results and the paper is concluded in section 6.

2. Feature Selection

String, linguistic, structural and instance based methods are four different category of measuring similarities (features) in ontology alignment. Here, top 15 effective similarity methods from the first three categories have been selected. Instance based similarity measure is not used, because of its difficulty to provide dataset. Each method returns a similarity value in the range of [0,1] for a given entity pair from two ontologies. These methods are briefly introduced in the following subsections.

2.1. *String based Methods*

There are several string based methods in ontology alignment field. These techniques focus on entity's name (string) and find similar string entities. Here, the most popular methods which are already implemented in Alignment API and SecondString API have been selected [9,21]. Because of low accuracy of each string based method, more methods from this category are used compare to the others, so that each method calculates different view of similarity (distinct feature). The overall performance can be increased by having a diversity of distinct features. This work's experimental results have shown that the following methods provide the more accurate outcomes. These methods are performed on two entity's names (two strings).

- ***N-gram similarity*** compares two strings and calculates the number of common n-grams between them. An n-gram composed of all sequences of n characters [9]. For instance, three-gram of word "paper" are: "pap", "ape" and "per".
- ***Levenshtein distance*** computes the minimum number of insertion, deletion and substitution of characters is needed to transform one string to another [1].
- ***SMOA*** is based on the number of common part of the two strings, while considering the length of mismatched substrings and the length of the common prefix of both strings [22].
- ***Dice coefficient*** is defined as twice the number of common terms of compared strings over the total number of terms in both strings. The coefficient result of 1 indicates identical vectors, while 0 equals orthogonal vectors [23].
- ***UnsmoothedJS*** is kind of Jensen-Shannon distance for two unsmoothed unigram language models. Jensen-Shannon distance is a popular method of measuring the similarity between two

(or more) probability distributions [23].

- ***Monge-Elkan distance*** uses semantic similarity of a number of strings or substrings. Each substring is evaluated against the most similar substring in the comparison entities' name [17].
- ***Substring similarity*** calculates the similarity of two strings based on their common longest substring [2].
- ***Needleman-Wunsch*** applies a global alignment on two sequences (strings). It is suitable measure when the two sequences are of similar length, with significant degree of similarity throughout. It also determines whether it is likely that two sequences evolved from the same string [21].
- ***Smith-Waterman distance*** is a version of Needleman-Wunsch which measures local sequence alignment. In other words, it determines similar regions between two string sequences. Instead of looking at the total sequence, this algorithm compares segments of all possible lengths and optimizes the similarity measure [23].
- ***Cosine similarity*** transforms the input string into vector space so that the Euclidean cosine rule is used to determine similarity [22].
- ***Jaccard measure*** is operated on two vectors X and Y. In this case, each vector is an entity name. The inner product of X and Y, and Euclidean norm of each vectors are used to calculate the similarity measure [22].
- ***Jaro measure*** finds words with spelling mistakes [9].

2.2. Language based Methods

Apart from similar appearance of entities name which has been measured through the string based methods, there are some semantic similarities between which reflect the applied language in ontologies. For example, although "car" and "automobile" have almost no string based similarity, but they refer to the same concept from language point of view.

WordNet is the most popular lexicon in English [5]. It arranges the word semantically rather than morphologically. WordNet is a network which has several synset. Every synset includes words with the same sense. Here, WordNet's package of Alignment API tool has been used to measure possible linguistic similarities of correspond entities' names.

2.3. Structural based Methods

Ontology alignment solely based on string and linguistic similarities may fail. Because these similarities only investigate the entities name without considering the entity's relation to other entities in its ontology. For instance, the result of applying the string and linguistic methods on two entities named "jackpot" from two given ontologies shows they are equal entities, while investigation of each entity in its own ontology may result opposite, e.g. if they are from two different ontology like kitchenware ontology and game ontology. Thus, structural based methods are defined to evaluate the similarity of entities and relations in two ontologies.

This work has investigated two structural based methods from the OLA's tool [4]. These methods compute the similarity measure of class names and their property locally, which are then aggregated into one particular measure.

3. Machine Learning Techniques

When similarity features of two given entities from two ontologies are selected and measured, they will be aggregated. There are several techniques to comput the optimal aggregation for different type of similarity measures such as fuzzy, weighted product, weighted sum, Minkowski and etc. [7]. However choosing the optimum parameters of these techniques such as thresholds and other constraints are difficult. ML provides another possibility to combine different similarity measures. Here, supervised ML methods are utilized to extract optimal model of compound metrics. Thus the alignment problem is transformed to a supervised ML task.

The basis of any ML-based ontology alignment system is a classifier. So far, numerous classifiers have been developed and applied to ML-based decision making problems. Here, the ontology alignment (classification) is regarded as a probability density function modeling. In this way, a parametric approach is used, in which explicit assumptions are made about underlying model characteristic [24]. This includes some parameters that need to be optimized by fitting the model to the dataset.

In this work, the performance of several classifiers such as SVM, KNN, DT and a re-sampling ensemble method (AdaBoost) are analyzed to select the one with the most accurate results. These techniques are briefly introduced in the following sub-sections.

3.1. Support Vector Machine (SVM)

Given a set of training instances, which are marked as two categories of alignment and non-alignment, an SVM training algorithm builds a model that predicts the category into which a new instance falls. Intuitively, an SVM model is a representation of the instances as points in space, so that the instances of the separate categories are divided by a clear gap that is as wide as possible. A new instance is then mapped into that same space and its category is predicted [25].

In other words, an SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space which can be used for classification, regression or other tasks. A good separation is achieved by the hyperplane that has the largest distance to the nearest training datasets of any class.

For a separable classification task, the idea is to map the training data into a higher-dimensional feature space using a kernel function where a separating hyperplane (*w*,*b*) with *w* the weight vector and *b* the bias, can be found which maximizes the margin or distance from the closest data points. The optimum separating hyperplane can be represented based on the kernel function (as Eq. (1)).

$$f(\mathrm{x}) = \mathrm{sign}\sum_{\mathrm{i}}^{\mathrm{n}}(\alpha_{\mathrm{i}}\mathrm{y}_{\mathrm{i}}\mathrm{K}(\mathrm{x}_{\mathrm{i}}.\mathrm{x})+\mathrm{b}) \tag{1}$$

where n is the number of training examples, y_i is the lable value of example i, K represents the kernel. Subject to the constraints $\alpha_i \geq 0$ and $\sum\alpha_i\mathrm{y}_\mathrm{i}=0$, there is a Lagrangian multiplier α_i for each training point and only those training examples that lie close to the decision boundary have nonzero α_i. These examples are called support vectors. With a suitable choice of the kernel the original non-separable data in input space become separable in feature space. Thus, kernel substitution presents a solution for obtaining nonlinear algorithms previously restricted to handling linearly separable cases. There are many kernels that can be used such as Gaussian Radial Basis function (RBF) as shown in Eq. (2).

$$k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2) \tag{2}$$

where σ >0 is a constant that defines the kernel width.

3.2. K-Nearest Neighbors (KNN)

The KNN classifier has been broadly used in ML applications due to its conceptual simplicity, and general applicability [24]. A KNN classifier is trained by storing all training patterns presented to it. During the test stage, the K stored entity pairs closest to the test entity pair are found using the Euclidian distance measure. A vote is then taken amongst those K neighbors and the most frequent class is assigned to that test entity pair. This assignment minimizes the probability of the considered test entity pair being wrongly classified. The reader is referred to [25] for the details of this algorithm. In KNN classification, the number of neighbors, i.e. K needs to be predefined. A single nearest neighbor technique (K=1) is primarily suited to classifications where there is enough confidence in the fact that class distributions are non-overlapping and the features used are discriminatory. But in most practical applications, such as ours, more than one nearest neighbor is necessary for majority voting.

A reasonable and practical approach would be to use trial and error to identify K such that it gives the lowest misclassification error rate. This is performed with different K values ranging from 1 to 9 to find the optimum value (section 5).

3.3. Decision Tree (DT)

Different methods exist to build DTs, which summarize given training data in a tree structure, with each branch representing an association between feature values and a class label. The most famous and representative amongst these is, perhaps, the C4.5 algorithm [24]. It works by recursively partitioning the training dataset according to tests on the potential of feature values in separating the classes. The core of this algorithm is based on its original version, named the ID3. So, to have a basic understanding of how this algorithm works, the ID3 method is outlined below.

The DT is learned from a set of training instances through an iterative process, of choosing a similarity measure (i.e., feature) and splitting the given data set according to the values of that feature. The key question here is which feature is the most influential in determining the classification and hence should be chosen first. Entropy measures or equivalently, information gains are used to select the most influential, which is intuitively deemed to be the feature of the lowest entropy (or of the highest information gain).

In more detail, the learning algorithm works by: (a) computing the entropy measure for each feature, (b) partitioning the set of examples according to the possible values of the feature that has the lowest entropy, and (c) for each subset of instances repeating these steps until all features have been partitioned or other given termination conditions met. In order to compute the entropy measures, frequencies are used to estimate probabilities. Note that although feature tests are chosen one at a time in a greedy manner, they are dependent on results of previous tests.

Explaining the results is one of the most popularity reasons of DT classifier in ontology alignment domain. It can be easily converted to set of rules or expression logic and created very fast [24, 25].

3.4. AdaBoost

For an ensemble technique to achieve higher accuracy than a single classifier, it is crucial that the base classifiers are sufficiently diverse. Bagging and Boosting are among the most popular re-sampling ensemble methods that generate and combine a diversity of classifiers using the same learning algorithm for the base classifiers. Boosting algorithms are considered stronger than bagging on noise free data. However, there are strong empirical indications that bagging is much more robust than boosting in noisy settings [26]. AdaBoost is a practical version of the boosting approach. Our experimental results regarded to our dataset reveal that boosting methods outperform the bagging methods.

Having provided an input training set including *m* elements, AdaBoost calls a given weak or base learner algorithm repeatedly in a series of rounds $t = 1, \dots,$ T. One of the main ideas of algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example *i* on round *t* is denoted by w_i^t. Initially, all weights are set equally (e.g. $w_i^1 = 1/m$), but on each round, the weight of misclassified examples are increased so that the weak learner forces to focus on the hard examples in the training set. The weak learner is responsible to find a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ appropriate for the distribution w^t.

The distribution w^t is next updated using Eqs. (3) and (4):

$$w_i^{t+1} = \frac{w_i^t \exp[-\alpha_t y_i h_t(x)]}{C_t} \tag{3}$$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-e_t}{e_t}\right) \tag{4}$$

where C_t is a normalization factor, and e_t is the error of h_t. As the result of Eq. (3), the weight of misclassified examples by h_t is increased, and the weight of correctly classified examples by h_t is decreased. Thus, the weight tends to concentrate on hard examples.

The output is calculated by final hypothesis *f*, shown in Eq. (5). It is a weighted majority vote of the T weak hypothesis, where α_t is the weight assigned to h_t [27].

$$f(\mathrm{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} (\alpha_t h_t(\mathrm{x}))\right) \tag{5}$$

This work chooses DT and SVM as the base classifiers of AdaBoost.

4. Proposed Alignment Method

The proposed system is implemented in JAVA and adopted Alignment API framework and MATLAB.

The datasets are taken from Ontology Alignment Evaluation Initiative (OAEI) which provides framework in ontology alignment. These datasets are produced for alignment contest and provide several formats [28]. Indeed, the evaluation of proposed system is carried out by OAEI API.

Series #301-304 represent real-life ontologies for bibliographic references found on the web. Here, #301 is selected as training dataset, while #302-304 series are considered as test datasets. It should be noted that all series are aligned to #101.

To construct the *similarity matrix,* similarity measures (section 2) are applied to a pair of ontologies selected from the above datasets. The similarity matrix is a table with *m* rows and *n* columns; where *m* is the number of given entity pairs and *n* is the number of applied features (similarity measures). The truth alignment of each entity pair correspondent to each row of similarity matrix is called *actual value.* This value is defined by the expert and takes a value of 1 (i.e. aligned) or 0 (i.e. not aligned).

Having provided the similarity matrix and target values, the problem would be reduced to a supervised learning task comprised of training and testing phases. Figure 1 illustrates the details.

Training Phase	**Testing Phase**
Two Ontologies (as input)	Two Ontologies and Training Model (as input)
⇩	⇩
Extracting Similarity Matrix and Actual Values	Extracting Similarity Matrix and Actual Values
⇩	⇩
Aggregating Similarity Matrix via Classification	Using Training Model on Similarity Matrix
⇩	⇩
Adjusting Classifier's Parameters	System Alignment (as output)
⇩	⇩
Extracting Training Model (as output)	Comparison of Actual Values and System Alignment

Fig. 1. Training and testing phases in proposed alignment system.

In this work, a binary classification with the objective of achieving the best possible alignments in an automatic and efficient way is introduced.

Within the test stage, the trained optimum model is used to classify the new unseen similarity matrixes (test data) into two classes i.e. aligned or not aligned. This type of alignment is named *system alignment*.

Each classifier is quantitatively evaluated by independent test data; otherwise the evaluation would become biased and would not present a fair assessment of the classifier performance. To assess the classifier generalization ability and thus measure the classification accuracy, *system alignment* and *actual value* of each entity pair are compared.

4.1. Evaluation Criteria

In ontology alignment task, precision and recall criteria are generally used to evaluate the system's performance [29]. These measures are defined as Eqs. (6) and (7).

$$\text{Precision} = \frac{|\text{alignment given} \cap \text{correct alignment}|}{|\text{alignment given}|} \quad (6)$$

$$\text{Recall} = \frac{|\text{alignment given} \cap \text{correct alignment}|}{|\text{correct alignment}|} \quad (7)$$

F-measure is basically the harmonic mean of precision and recall, and defined as Eq. (8).

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})} \quad (8)$$

F-measure is a common performance measure in information retrieval which balances precision and recall. Indeed, Alignment API provides a utility to evaluate the result of alignment [2,5].

4.2. Experiments

Here, four experiments have been conducted; each experiment considers an aspect which has its impact on the training model and final results. Furthermore, each experiment is carried out using different classifier (DT, SVM, KNN and AdaBoost models) and the results are compared against each other.

These experiments are explained as follows.

4.2.1. First Experiment

The first experiment has simply chosen the optimum model based on those 15 similarity measures which are represented in section 2.

4.2.2. Second Experiment

This experiment investigates the comments role in ontology alignment, so that the comments of every entity (if exist) are added to the dataset. For extracting valuable words, each sentence is tokenized and then dummy and auxiliary words are eliminated, so that remained words are meaningful information. Furthermore to save the time, this process is only employed on entity pairs which are not fully aligned and their similarity measure is less than 1.

The rest of this experiment is the same as first experiment.

4.2.3. Third Experiment

This experiment explores the effect of training samples quantity on the quality of final trained model. In previous two experiments, two ontologies (#101, #301) are utilized to build the training model. In this experiment, the number of entity pairs is increased by using other ontologies such as #102, #103, i.e. entity pairs extracted from (#101, #102) and (#101, #103). So the diversity of instances in training phase is widened. To avoid training the model with similar input samples, those samples from #102 and #103 ontologies which represent the highest variances are selected.

4.2.4. Fourth Experiment

This experiment takes advantage of feature selection technique to eliminate the ineffective features. To do that, a feature selection method is used to rank features based on their weights in SVM. This method calculates the set of feature’s weight in SVM classifier and eliminates features that have less effect by iteration. So those features are only selected which lead to better discrimination ability [25]. As the result, 8 features from section 2 i.e. SOMA, Needleman-Wunsch, WordNet, Jaccard, Dice coefficient, N-gram, and two structural similarities are chosen.

Since the number of features is decreased from 15 to 8, the similarity matrix is created faster and in less memory compared to first experiment. Furthermore, based on the result of second and third experiments, comments (if exist) are also added to the dataset and the diversity of instances is enlarged in training phase.

5. Results and Evaluation

This study optimizes the classifiers. If every parameter of each classifier tunes well, the alignment result will be more accurate.

The design of the SVM classifier architecture is simple and mainly requires the choice of kernel and its associated parameters. There are currently no techniques available to learn the form of kernel; thus a Gaussian RBF kernel function has been employed. We construct a set of SVM classifiers with range of values for the kernel parameter σ and with no restriction on the Lagrange multipliers α_i. Having defined *classification rate* as the system alignment over the truth alignment, the best classification accuracy is achieved when $\sigma = 0.1$.

In KNN classification, the number of neighbors, i.e. K needs to be pre-defined. A reasonable and practical approach would be to use trial and error to identify K such that it gives the lowest misclassification rate. We performed such an experiment with different K values ranging from 1 to 9 (K is chosen to be odd to avoid tie votes), and found K = 3 as the optimum K value for the application at hand.

In DT, having minimum tree without losing accuracy significantly decreases the costs. Therefore after constructing DT, it is configured to estimate the minimum tree with the lowest cost for every test set. Here, the minimum tree size is experimentally found to be 12.

This work also experiments an AdaBoost method with two different base classifiers, i.e. SVM and DT noted as AdaBoost (SVM) and AdaBoost (DT), respectively.

In AdaBoost (SVM), finding a suitable σ for SVM base learner is non-trivial. Because having too large value for σ often results in too weak SVM classifier with RBF kernel. The AdaBoost (SVM) classification accuracy is often less than 50% and cannot meet the requirement of a given AdaBoost classifier. On the other hand, a smaller σ often makes stronger SVM with RBF kernel and so boosting them may become inefficient [30]. Here, the AdaBoost (SVM) algorithm initiates firstly by one SVM base learner with the optimum σ value from previous experiment (i.e. $\sigma = 0.1$). The final optimum architecture is comprised from three SVM base learners with the optimum σ values equal to 0.1, 0.09 and 0.08, respectively.

In AdaBoost (DT), having suitable tree size becomes important. Again, this value is set by 12 which obtained from previous experiment. The optimum number of AdaBoost rounds varies in each experiment. Here, these round numbers have been found 18, 2, 15, and 12 for experiment #1,…,#4, respectively.

Table 1 summarizes the best F-measure performances obtained from all experiments against the test set #302. As it can be seen, the KNN and AdaBoost provide the first and second best results, respectively. Indeed, the obtained performances for both AdaBoost (DT) and AdaBoost (SVM) classifiers are very close to each other. On the other hand, the worst results are provided by DT classifier.

Table 1. F-measure values against test set #302.

Experiment	KNN	DT	SVM	AdaBoost (DT)	AdaBoost (SVM)
#1	0.92	0.85	0.89	0.89	0.89
#2	0.91	0.88	0.89	0.91	0.91
#3	0.92	0.85	0.91	0.91	0.90
#4	0.92	0.83	0.87	0.89	0.90

Similarly, Table 2 and Table 3 summarize the best F-measure performances obtained from all experiments against the test set #303 and #304 respectively. As it can be seen, the fourth experiment which benefits from feature selection mostly outperforms the first experiment, while on average the KNN and AdaBoost (DT) classifiers perform better amongst all exploited classifiers.

Table 2. F-measure values against test set #303.

Experiment	KNN	DT	SVM	AdaBoost (DT)	AdaBoost (SVM)
#1	0.86	0.86	0.80	0.87	0.82
#2	0.81	0.84	0.76	0.87	0.79
#3	0.85	0.87	0.80	0.84	0.90
#4	0.87	0.86	0.88	0.90	0.88

Table 3. F-measure values against test set #304.

Experiment	KNN	DT	SVM	AdaBoost (DT)	AdaBoost (SVM)
#1	0.98	0.94	0.94	0.94	0.95
#2	0.97	0.96	0.96	0.96	0.96
#3	0.97	0.94	0.95	0.97	0.96
#4	0.98	0.96	0.96	0.99	0.98

In general, F-measure values which are obtained against test sets #304 and #303 are the best and worst results, respectively. This is due to the fact that test set #304 has similar structure and vocabularies to the reference ontology, i.e. #101, while test set #303 has the least vocabularies and linguistic information. This trend also validates the recent attentions on reusing the existing ontologies.

Although, the optimum AdaBoost (DT) model can provide the best results, but usually creation of training model for an ensemble-based classifier need much more time and memory compared to non-ensemble ones. To this end, Table 4 represents the needed test time in terms of seconds to perform fourth experiment using different classifiers. As it can be seen, the required time for AdaBoost (DT) is reasonable compared to other non-ensemble classifiers, but it is much longer for AdaBoost (SVM) classifier.

Table 4. Time comparison of different classifiers in experiment #4.

Classifiers	**KNN**	**DT**	**SVM**	**AdaBoost (DT)**	**AdaBoost (SVM)**
Time (seconds)	0.0798	0.3357	0.0304	0.3676	0.9070

Table 5 compares the F-measure of this system with the most important previous approaches. The result of our fourth experiment using the KNN and AdaBoost (DT) achieve remarkable improvement in ontology alignment.

Table 5. Comparison of different methods using the F-measure.

Alignment Method / Test set	**Fourth Experiment**		**FOAM[9]**	**DT[15]**	**NB[15]**	**Classes[16]**	**Properties[16]**	**OMAP[17]**	**OLA[18]**
	KNN	**AdaBoost (DT)**							
#302	**0.92**	**0.89**	0.77	0.759	0.753	0.69	0.85	0.74	0.34
#303	**0.87**	**0.90**	0.84	0.816	0.860	0.86	0.88	0.84	0.44
#304	**0.98**	**0.99**	0.95	0.960	0.960	0.94	0.98	0.91	0.69

6. Conclusions

This paper proposes an efficient method for ontology alignment based on the combination of different similarity categories in one input sample. This, in turn, increases the discrimination ability of the model and enhances the system's overall accuracy.

The proposed model determines the alignment process with no prior need to ontology instances, which facilitates alignment task.

Through a comprehensive optimization process of operational parameters, our proposed model does not require any user intervention, and it has consistent performance for both aligned and non-aligned entities.

AdaBoost (DT) model provides the best overall accuracy, especially when feature selection scheme is utilized. Experimental results demonstrate that the F-measure criterion improves up to 99% which is better than other related works that have used up to 23 similarity measures. Although, this work is using only 8 similarity measures in its optimum model, but the possible impacts of feature reduction has been compensated by using ontology comments, enlarging the diversity of training set samples, and choosing more effective similarity measures. This grants more accuracy and less computation cost which makes the proposed model appropriate even for an online ontology alignment task.

References

[1] J. Euzenat, T. L. Bach, J. Barrasa, P. Bouquet, J. De Bo, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. Van Acker, I. Zaihrayeu,"D2.2.3: State of the art on ontology alignment", Knowledge web, pp. 5-12, 2004.

[2] J. Euzenat, P. Shvaiko, "Ontology Matching", Springer, 2007.

[3] L. Predoiu, C. Feier, F. Scharffe, J. de Bruijn, F. Recuerda, D. Manov, M. Ehrig, "D4.2.2 State-of-the art survey on Ontology Merging and Aligning V2", SEKT Consortium, 2005.

[4] J. Euzenat, "Alignment API and server", INRIA & LIG, pp. 32, 2008.

[5] R. Zhang, Y. Wang, J. Wang, "Research on Ontology Matching Approach in Semantic Web", International Conference on Internet Computing in Science and Engineering, pp. 1, 2008.

[6] X. Li, J. Zhang, T. Huang, "A Standardized Learning Resources Retrieval System Based on Ontology Matching", Springer-Verlag Berlin Heidelberg, 2007.

[7] A. Doan, J. Madhavan, P. Domingos, A. Halevy, "Learning to map ontologies on the semantic web", Proceeding of www, 2002.

[8] M. Ehrig, S. Staab, Y. Sure, "Bootstrapping Ontology Alignment Methods with APFEL", Springerlink, 2005.

[9] J. David, F. Guillet, H. Briand, "Association Rule Ontology Matching Approach", International Journal on Semantic Web & Information Systems, Vol. 3, Issue 2, 2007.

[10] B. Bagheri Hariri, H. Sayyadi, H. Abolhassani, "A Neural-Networks-Based Approach for Ontology Alignment", Proceedings of the Joint 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on advanced Intelligent Systems, Japan, 2006.

[11] B. Bagheri Hariri, H. Sayyadi, H. Abolhassani, "Combining Ontology Alignment Metrics Using the Data Mining Techniques", Proceeding of the 17th European Conference on Artificial Intelligence, International Workshop on Context and Ontologies (C&O' 2006), Trento, Italy, 2006.

[12] H. Stuckenschmidt, M. Ehrig, J. Euzenat, A. Hess,W. R. van Hage, W. Hu, N. Jian, G. Cheng, Y. Qu, G. Stoilos, G. Stamou, U. Straccia, V. Svatek, R. Troncy, P. Valtchev, M. Yatskevich, "D2.2.4: Alignment implementation and benchmarking results", Knowledge Web Consortium, 2006.

[13] K. Eckert, C. Meilicke, H. Stuckenschmidt, "Improving Ontology Matching using Meta-level Learning", Proceedings of ESWC, 2009.

[14] R. Ichise, "Machine Learning Approach for Ontology Mapping using Multiple Concept Similarity Measures", Proceedings of ICIS, 2008.

[15] K. Eckert, C. Meilicke, H. Stuckenschmidt, "Improving Ontology Matching using Meta-level Learning," In Proceedings of ESWC, 2009.

[16] M. Mao, "Ontology Mapping: Towards Semantic Interoperability in Distributed and Heterogeneous Environment", Ph.D. Thesis, 2008.

[17] U. Straccia, R. Troncy, "oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies", Springer-Verlag Berlin Heidelberg, LNCS 3806, 2005, pp. 133–147.

[18] J. Euzenat, P. Guegan, P. Valtchev, "OLA in the OAEI 2005 alignment contest", 2005.

[19] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas, "On Combining Classifiers", IEEE Trans. On Pattern Analysis and Machine Intelligence, 20, pp. 226-239, 1998.

[20] K. Tumer, J. Ghosh, "Classifier combining: analytical results and implications", Working notes from the Workshop, Integrating Multiple Learned Models., 13th National Conference on Artificial Intelligence, 1996, Protland, Oregon.

[21] SecondString Project Page, <http://secondstring.sourceforge .net>.

[22] G. Stoilos, G. Stamou, S. Kollias, "A String Metric for Ontology Alignment," Springer-Verlag Berlin Heidelberg ISWC 2005, LNCS 3729, pp. 624–637, 2005.

[23] W. W. Cohen, P. Ravikumar, S. E. Fienberg, "A Comparison of String Metrics for Matching Names and Records", American Association for Artificial Intelligence, 2003.

[24] R. O. Duda, P. E. Hart, D. J. Storke, *Pattern Classification*, John Wiley & Sons, New York, 2001, ISBN:0471056693.

[25] E. Alpaydin, *Introduction to Machine Learning*, MIT press, 2004.

[26] S. B. Kotsiantis, P. E. Pintelas, "Combining Bagging and Boosting", International Journal of Computational Intelligence, Vol. 1, Number 4, 2004.

[27] R. E. Schapire, "A Brief Introduction to Boosting", Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence, 1999.

[28] Ontology Alignment Evaluation Initiative, (2009), <http://oaei.ontologymatching.org>.

[29] H. Doa, E. Rahm, "Matching large schemas: Approaches and evaluation", Information Systems 32, 2007, pp. 857–885.

[30] X. Li, L. Wang, E. Sung, "AdaBoost with SVM-based component classifiers", Engineering Applications of Artificial Intelligence, Vol. 21, pp. 785-795, 2008.