Journal of Web Engineering, Vol. 16, No.7&8 (2017) 619-652

A METRIC BASED AUTOMATIC SELECTION OF ONTOLOGY MATCHERS USING BOOTSTRAPPED PATTERNS

B. SATHIYA T.V. GEETHA

College of Engineering, Guindy, Anna University, Chennai, Tamil Nadu, India

sathiyabalu89@gmail.com tv_g@hotmail.com

VIJAYAN SUGUMARAN

Oakland University, Rochester, Michigan, USA

sugumara@oakland.edu

Received December 8, 2016
Revised May 16, 2017

The ontology matching process has become a vital part of the (semantic) web, enabling interoperability among heterogeneous data. To enable interoperability, similar entity pairs across heterogeneous data are discovered using a static set of matchers consisting of linguistic, structural and/or instance matchers that discover similar entities. Numerous sets of matchers exist in the literature; however, none of the matcher sets are capable of achieving good results across all data. In addition, it is both tedious and painstaking for domain experts to select the best set of matchers for the given data to be matched. In this paper, we propose two bootstrapping-based approaches, Bottom-up and Top-down, to automatically select the best set of matchers for the given ontologies to be matched. The selection is processed, based on the characteristics of the ontologies which are quantified by a set of quality metrics. Two new structural quality metrics, the Concept External Structural Richness (CESR) and the Concept Internal Structural Richness (CISR), have also been proposed to better quantify the structural characteristics of the ontology. The best set of matchers is chosen using the sets of patterns learned through the proposed Bottom-up and Top-down bootstrapping approaches. The proposed metrics and the patterns constructed using these approaches are evaluated using the COMA matching tool with existing benchmark ontologies (Benchmark, Conference and Benchmark2 tracks of the OAEI 2011). The proposed Bottom-up based patterns, along with the two proposed quality metrics, achieved better effectiveness (F-measure) in selecting the best set of matchers in comparison with the static set of matching, supervised ML algorithms and the existing automatic matching. Specifically, the proposed Bottom-up patterns achieve a 14.6% Average Gain/Task and a significant improvement of 129% in comparison with the existing KNN model's Average Gain/Task.

Key words: Automatic Matching, Ontology Matcher Selection, Bootstrapping Patterns, Ontology Matching, Ontology Metrics
Communicated by: D. Schwabe & Y. Deshpande

1 Introduction

Ontology matching systems have gained considerably in significance, given that they form the basis for information systems that use heterogeneous data. These data are represented in different formats with different levels of specifications such as unstructured raw text, semi-structured XML and structured data such as tables, schema, and ontologies. Basically, ontology matching systems comprising of a set of element matchers aim at finding a set of semantically-similar entity pairs, called an alignment set, across the input data to be matched. The set of element matchers can be categorised, based on the information used, as follows: (i) Lexical matchers (L) that explore textual information such as an entity's name, label, and comments, (ii) Structural matchers (S) that use details of entities like attributes, domain, range, relations, depth, and path, and (iii) Instance matchers (I) that use information from the tuples or instances of the data.

The alignments of matching systems are used in various applications of the web (Euzenat & Shvaiko, 2007), such as heterogeneous web data integration, data translation, peer-to-peer information sharing, web service composition, multi-agent communication on the web, query answering on the web and the Deep Web. In all these applications, matching systems are designed to eliminate assorted heterogeneities and semantic ambiguities prevailing among the data using the discovered alignments. Diverse existing ontology matching systems, their applications and various challenges are outlined in a number of survey papers [6, 16, 27, 30, 32, 34] and books [4, 13], which represent state-of-the-art developments in this field.

Generally, ontology matching systems use a static set of matchers to explore the two input ontologies to be matched from different perspectives so as to discover similarities between the entities. The two input ontologies which need to be matched are termed a matching task. However, based on the application, one of the inputs to the matching system can be a user query, schema, or catalogue. The static set of matchers comprising linguistic, structural and/or instance-based matchers is deployed for all matching tasks, irrespective of their characteristics. However, it is not mandatory for all matching tasks to have all the three types of information incorporated in them. Hence, these static set of matchers have shown reduced efficiency and/or effectiveness in a significant number of cases. For example, let us consider two ontologies, O1 and O2, with *n* and *m* number of entities respectively and no instances. An ontology matching system with the static set of matchers comprising linguistic, structural and instance matchers is deployed to find similarities across the entities of these two ontologies. Each of these matchers compares each entity of ontology O1 with each entity of ontology O2, leading to n*m computations. In total, for all the three matchers, 3*n*m computations are required. Such a huge number of computations would culminate in scalability issues like memory insufficiency and extremely time-consuming process. Let us consider a reverse scenario where the static set of matchers does not have an instant matcher and the matching task has instances. In this case, certain potential alignments will be left undiscovered by the static set of matchers.

Therefore, there is no single set of matchers [34] which can effectively and efficiently discover alignments for all matching tasks. In addition, as the size and heterogeneity [2] of the ontologies increase, the need for scalable, robust and automated ontology matching systems becomes inevitable. Also, manually choosing the best set of matchers with an appropriate combination strategy for the given matching task is a painstaking requisite for domain experts, owing to the availability of a large numbers of matchers and their corresponding combination strategies. There is, consequently, a great

need for an automated ontology matching system which can choose, combine and tune a set of matchers automatically, based on the given matching task. Such an automatic selection of a set of matchers, its combination strategy, as well as the automatic tuning of each matcher, are open challenges in the field of ontology matching [34]. In this paper, we propose an approach that tackles the automatic matcher selection issue where a set of matchers is selected, based on the characteristics of the matching task.

In the scenario above, if an automatic matcher selection strategy is deployed, only linguistic and structural matchers are selected and the number of computations reduced to 2*n*m. Generally, the benefits of the automatic selection of matchers are twofold: First, choosing an appropriate set of matchers leads to a robust system, i.e., more effective matching results are obtained. Second, eliminating unnecessary matchers leads to a more scalable system, resulting in greater efficiency. An automated matcher selection-based ontology matching system can be used in applications where data have varying characteristics. For example, a candidate application would be an integration of sets of schema pairs with varying characteristics. That is, one schema pair may have linguistic and structural characteristics, while the other pair may have only linguistic characteristics, and so on.

Hence, the objective of this research is to develop an approach for choosing the best set of matchers automatically from a pool of matchers, based on the characteristics of the matching task. To enable automatic choosing, the proposed approaches should learn which combination of characteristics of the matching task quantified using a set of ontology quality metrics is best matched by which set of matchers. To accomplish this, two new semi-supervised machine learning approaches, Bottom-Up bootstrapping (BUB) and Top-Down bootstrapping (TDB), have been proposed that learn the sets of patterns for each set of matchers. Each pattern consisting of a set of ontology quality metrics represents the required linguistic, structural and instance characteristics for a particular set of matchers. After the learning process, sets of patterns are used to automatically select the best set of matchers for the given matching task, based on its characteristics. Further, for an accurate selection of the best set of matchers, the characteristics of the matching task should be precisely quantified. Therefore, two new ontology quality metrics for computing ontology structural characteristics called the Class External Structural Richness (CESR) and the Class Internal Structural Richness (CISR) are proposed, which are designed to compute the percentage of concepts which have attributes and relations.

The rest of the paper is organised as follows. Section 2 presents the related work. Section 3 describes the construction procedure of the training dataset. Section 4 details the two new semi-supervised bootstrapping approaches. The experimental results and inferences are discussed in Section 5. Section 6 concludes the paper with a note on future work.

2 Current Practice and Research

There has been a growing number [13] of different automation strategies deployed in different stages of the matching system. The first stage of automation occurs in the selection of matchers which automatically choose a set of matchers from a pool of matchers. In the second stage, the tuning of the matchers is carried out, where the selected matcher's parameters are fixed automatically. In the final stage, the sets of outputs (sets of similar entity pairs) of the various selected matchers are combined automatically. Of the three stages, the first stage of automation is crucial because the rest of the two stages are dependent on the first. Also, to yield good results, the correct choice of matchers is

mandatory and hence this paper aims to automatically select them. This section details the existing work related to the automatic selection of matchers. Further, it also outlines miscellaneous ontology matching systems that automatically combine and tune matchers.

For the automatic tuning of the matchers, Yang et al. [39] used the particle swam optimization technique based on the parameter, space sampling, to automatically tune the matcher's parameters. Ehrig, Staab and Sure [12] employed the heuristic combination strategy and determined the threshold of each matcher using varied machine learning algorithm(s) such as decision trees (e.g., C4.5), neural networks, and support vector machines.

For an automatic combination of the matchers, different strategies have been used as follows. In [38], the weight of each matcher is computed for every pair of entities to be matched, depending on the features of the pair. Based on the features, the accuracies of certain sets of matchers are calculated using manual rules and others are predicted using a multilayer perceptron. Finally, for the given entity pair, the matchers are dynamically combined based on the weights.

In [9], machine learning techniques are used to discover alignments. A set of learners is trained to match, based on user-created training data and combined by means of a meta-learner which weighs each learner based on the accuracy of the matching results.

HADAPT: The harmony-based adaptive aggregation method [22] combines the matching results based on a measure called harmony (aka dominants [14]). The measure computes a weight for each matcher based on the number of correctly-matched entity pairs by the matcher. Matchers are then dynamically combined, based on the harmony value for the given matching task. HADAPT is classified as a global method by Ngo and Bellahsene [26], stemming from the use of the semantic context of the entity for aggregation.

Sagi and Gal [32] designed a set of matching predictors to access the accuracy of the matcher in discovering similar entity pairs. The predictors are combined to analyse the matcher using the statistical method, stepwise regression,

In [20], a particular matcher is used to combine the other matchers [20], and in [36], matchers are combined iteratively where pervious iteration alignments are used for the combination. Domain experts manually select suitable combination strategies for the chosen set of matchers [3, 8, 16]. Other methods such as user feedback-based learning methodologies [10, 33], Multi-agent techniques [32], linear combination of the matchers [21] are also used.

In the initial versions of matcher selection, semi-automated or manual selection strategies have been used as follows. Mochol, Jentzsch & Euzenat [25] used domain experts to answer a set of questions based on which appropriate matchers are chosen. Huza, Harzallah & Trichet [19] and Mochol & Jentzsch [24] used input ontology characteristics, output specifications [19] and the matcher descriptions available to manually construct a rule-based system for choosing the matchers. Peukert, Eberius & Rahm [28] used a graphical user interface through which a set of matchers can be selected. Mochol, Jentzsch, & Euzenat [25] used approximately 40 characteristic factors specifying details on the input ontology, available matchers, usage of the results, output and documentation describing the requirements of the application. These factors are represented in a hierarchical tree fashion and an Analytical Hierarchy Process (AHP) is used to choose the matcher from a pool of matchers. The major drawback of this work is that, all the factors mentioned above, as well as the description of each

matcher in the pool, is required to be manually described by domain experts, which can lead to serious bottlenecks. But, the proposed system, however, requires no domain experts, user intervention or manual selection.

Ontology matching systems falling under the category of automatic matcher selection are briefly described below. The ontology matching system, Rimom, [21] is designed to perform matcher selection and combination automatically. It uses linguistic and structural metrics to quantify the characteristics of input ontologies. Based on the two metrics, matchers are selected from both the lexical and structural category of matchers and the matching results are combined automatically. In our proposed approaches, unlike in this system, there is no constraint in choosing at least one matcher from each category, given that all matching tasks have little information in all categories. Also, despite being fully automated, the Rimom system uses only two categories of matchers, and instance matchers are not considered. This is a smaller number when compared to the eight metrics used in our proposed system for the precise quantification of the characteristics of the matching task.

A self-configuring matching system was introduced by Peukert et al. [29] that is able to select and combine the matchers dynamically, based on the matching task. A set of schema features, as well as features based on the intermediate matching results, are used to analyse the characteristics of the given matching task and rules are applied to select and combine the matchers based on the features. However, these rules are limited and designed manually, based only on the features and matchers used in this system. On the other hand, the proposed bootstrapping approach learns patterns automatically for any given set of features and matchers.

Hariri et al. [17] selected the best set of matchers and its combination strategy using the neural networks learning algorithm. It chooses the best set of matchers from a given pool of matchers, irrespective of the characteristics of the input ontologies. Further, the matchers are categorised into string, linguistic, structural metric and the like, and at least one matcher from each category is chosen compulsorily, which is unnecessary for all the matching tasks.

Marie & Gal [15, 23] used boosting, a machine learning algorithm, to select and combine the matchers. The training dataset consists of a set of entity pairs, one from each of the ontologies, labelled with a binary value indicating whether or not the entity pair is similar. An element in the hypothesis space is a similarity matrix for the two input ontologies obtained using the set of matchers and its combination strategy. Based on the hypotheses, the system selects the best set of matchers and its combination strategy from a given pool of matchers similar to Hariri et al. [17], irrespective of the characteristics of the input ontologies. However, our proposed system chooses only the necessary category matchers based on the characteristics of the matching task, in contrast to the two approaches above.

Cruz et al. [7] proposed a supervised KNN (K-Nearest Neighbour) learning algorithm for automatic matcher selection, based on the characteristics of the input ontologies. Their work chooses matchers from a particular category of matchers, if the characteristics of the matching task possess sufficient quantitative information needed for the particular category. To quantify the characteristics of the matching task, a set of nine ontology quality metrics categorised as lexical, structural, syntactic and instance have been used. Further matchers from the pool of matchers are combined to form sets of matchers whose matching results are combined in a predefined manner where each such set of matchers is called a configured matcher. Five such configured matchers are presented, each designed to match a unique set of characteristics. Any new matching task will be quantified using these nine

metrics and the constructed KNN model will choose the best configured matcher based on the said metrics.

However, no open datasets were available to construct the KNN model and hence this system had to construct the required dataset. Despite this, only 233 labelled datasets are constructed from the matching tasks of the OAEI 2011 (Ontology Alignment Evaluation Initiative) [46], since the construction of the labelled data requires matching tasks along with reference alignments. Nevertheless, its availability is limited as a result of the protracted creation process of reference alignment sets by domain experts. Consequently, the use of a very limited training dataset for modelling a supervised KNN model is the prime drawback of their work. Generally, any supervised learning algorithm needs a sufficient number of labelled training datasets to avoid the underfitting and overfitting of the model. Owing to the limited availability of the labelled data, our system proposes two new semi-supervised bootstrapping algorithms which require a little labelled data and large unlabelled data. 233 labelled training and 11,589 unlabelled training datasets have been constructed, which is a very large number compared to the system above. The following section will discuss the construction of the training dataset.

3 Construction of the Training Dataset

In this section, the ontology metrics and the set of configured matchers used in the construction of the training dataset are described. Following this, the construction methodology for the labelled and unlabelled training datasets is discussed in detail.

3.1. Ontology Quality Metrics

Various ontology quality metrics exist in the literatures which quantify the characteristics of the ontology with respect to linguistic, structural, instance, and quality. However, in the context of an ontology matching system, each entity pair across the given two input ontologies is matched in terms of the linguistic, structural and instance similarities prevailing between them. So then, a set of metrics which is used to quantitatively define the presence of the characteristics is to be chosen. But prior to that, it should be noted that Hu, Qu & Cheng [18] stated that at least 50% of the final matching entity pairs can be found just by lexical matchers using linguistic information such as the concept's name, comment, label, etc. This proves that, by default, almost all ontology pairs have rich linguistic characteristics. Therefore, no linguistic metrics are assigned to explicitly ensure the presence of linguistic characteristics. The system only checks for the presence of structural and instance characteristics by a set of corresponding metrics chosen from the literature [7, 37]. Further, two new structural quality metrics, the Concept External Structural Richness (CESR) and the Class Attribute Richness (CAR), are also proposed to further accurately quantify the structural characteristics of any given ontology. The set of chosen and proposed metrics used in the bootstrapping approaches is briefly discussed below.

3.1.1. Structural Characteristic Metrics

Structural metrics are chosen and proposed such that both the external and internal structural characteristics of a given ontology are quantified. External structural characteristics such as the depth of the concepts, and taxonomical and non-taxonomical relations, are quantified by metrics such as Average Depth, Inheritance Richness, Relational Richness. and Concept External Structural Richness (CESR). Attribute, an internal structural characteristic, is quantified by metrics such as Attribute

Richness and Class Attribute Richness (CAR). The list of structural characteristic metrics used in this system, alongside the two new proposed structural characteristic metrics, are defined in the following subsections. In all the definitions of the metrics, c_i represents the i^{th} concept over the set of concepts C in the given ontology.

Relationship Richness

The Relationship Richness [37] of an ontology is defined as the ratio of non-taxonomical relationships to the total number of relations in the ontology.

$$RR = \frac{|\,Non\_ISA\,|}{|\,Non\_ISA\,| + |\,ISA\,|} \qquad (1)$$

In the definition above, $|Non\_ISA|$ represents the number of non-taxonomical relationships and ($|Non\_ISA| + |ISA|$) represents the total number of relations in the ontology.

Inheritance Richness

Inheritance Richness [37] of an ontology is defined as the average number of subconcepts for each concept of the ontology.

$$IR = \frac{\sum_{c_i \in C} |\,SubConcepts(c_i)\,|}{|\,C\,|} \qquad (2)$$

where the $|SubConcepts(c_i)|$ represents the number of sub-concepts for the class c_i.

Average Depth

Average depth [7] of an ontology is used to determine the average specificity of each concept in the ontology. It is defined as the ratio of the sum of the depth of each concept ($D(c_i)$) in the ontology to the total number of concepts in the ontology.

$$AD = \frac{\sum_{c_i \in C} D(c_i)}{|\,C\,|} \qquad (3)$$

Attribute Richness

Attribute Richness [37] of an ontology is defined as the average number of attributes defined for each concept in the given ontology.

$$AR = \frac{|\,att\,|}{|\,C\,|} \qquad (4)$$

In the definition above, $|att|$ represents the total number of attributes defined in the ontology.

The various existing structural metrics compute the average number of attributes [37], average number of taxonomical relations [37], count of children [11], count of ancestors [11], count of properties [11], and so on. But the average number and count of the entities' computations can be biased by a few concepts with a huge number of relations or attributes, leading to defective greater values. Hence, the percentage of concepts which have relation(s) and the percentage of concepts which contain attribute(s) should also be quantified. This is incorporated by the proposed two new structural metrics, the Concept External Structural Richness (CESR) and the Concept Internal Structural

Richness (CISR). This percentage (*P*) and average number (*A*) together can convey the information that *P*% of concepts have, on average, *A* number of attributes or relations. Such a precise measure of structural characteristics will aid in the process of selecting the best configured matcher.

Concept External Structural Richness (CESR)

The proposed metric, CESR, of an ontology is defined as the percentage of the number of concepts with taxonomical or non-taxonomical relations which are represented by the subclass of relations and object properties respectively.

$$\text{CESR} = \frac{\sum_{c_i \in C} R(c_i)}{|C|} \qquad (5)$$

where $R(c_i)$ is a Boolean function which returns a value of 1 if the concept c_i has any relation associated with it, else 0 is returned. The summation of this function value for all the concepts will give the number of classes associated with the relation(s) from which the CESR percentage can be computed.

Concept Internal Structural Richness (CISR)

The proposed CISR metric of an ontology is defined as the percentage of the number of concepts containing attributes, where the attributes are represented by the data property in the ontology.

$$\text{CISR} = \frac{\sum_{c_i \in C} P(c_i)}{|C|} \qquad (6)$$

where $P(c_i)$ is a Boolean function which returns 1 if the concept c_i contains an attribute, else 0 is returned.

3.1.2. Instance Characteristic Metrics

The metrics Average Population and Class Richness are chosen, so that both the average number of instances and the percentage of the concepts with the instance are computed.

Average Population

Average Population [37] of an ontology is the average number of instances per class. In the definition, |I| represents the number of instances available in the ontology. The metric is represented in real values ranging from 0 to infinity.

$$\text{AP} = \frac{|I|}{|C|} \qquad (7)$$

Class Richness

Class Richness [37] of an ontology is the percentage of the concepts for which the instances exist.

$$\text{CR} = \frac{\sum_{c_i \in C} I(c_i)}{|C|} \qquad (8)$$

where $I(c_i)$ is a binary function which returns 1 if the concept c_i contains an instance, or else 0 is returned. The metric is represented in real values ranging from 0 to 1.

All the metrics are represented in real numbers. RR, CESR, CISR, AP and CR range from 0 to 1, while the rest of the 3 metrics range from 0 to infinity. Further, the larger value of each metric represents the richness of the corresponding characteristics while the smaller value represents the deficiency of the characteristics in the ontology. The metrics listed above form a numerical vector with 8 values which quantifies the characteristics of the ontology in terms of structural and instance richness. For any given matching task, two metric vectors are computed, one for each of the input ontologies which are combined by an aggregating function called the FS-A [7]. The two metric vectors should be combined to summarize the characteristics of both the input ontologies, since the configured matchers are chosen based on the characteristics in question. The inputs of the FS-A function are the two values of the same metric *m* from two metric vectors belonging to the two input ontologies. The output of the FS-A is the single aggregated value for that metric *m*, formally defined as follows.

$$FS-A_m = \left(\left(\frac{m_L}{m_H(\log(m_H - m_L + 1) + 1)}\right) + \left(\frac{m_L + m_H}{2}\right)\right)/2 \qquad (9)$$

where m_L and m_H represent the lower value and higher value respectively between the two values of the metric *m* from two metric vectors. Similarly, the aggregate function FS-A is applied for all the metrics in the two vectors to obtain a combined metric vector. These combined metric vectors are used as follows. In the training phase of the approaches, the set of combined metric vectors computed for the set of matching tasks is used for the creation of the unlabelled and labelled data. In the testing phase of the approaches, any input ontology pair's characteristics are represented by this combined metric vector, which is subsequently processed by the learned patterns to identify the best configured matcher.

3.2. Configured Matchers

This subsection outlines the configured matchers recommended by the learned patterns. The set of matchers which forms the configured matchers can be categorised into lexical (L), structural (S) and instance (I) matchers. These sets of matchers are combined in all possible combinations to obtain the possible set of configured matchers, CM, as follows: a configured matcher consisting of lexical, structural and instance matchers denoted by L+S+I, (ii) L, (iii) L+S, (iv) L+I, (v) S+I, (vi) S, and (vii) I.

However, configured matchers which have no lexical matcher are ignored for two reasons: (i) The presence of a lexical characteristic is by default [19] and hence the matchers cannot be ignored, and (ii) Based on the experiment conducted to construct labeled training data, configured matchers without lexical matchers are never chosen as the best configured matchers for matching tasks. Hence the set of configured matchers used in our system is as follows: (i) L+S+I,, called the configured matcher cm1, (ii) L, called cm2, (iii) L+S, called cm3, and (iv) L+I, called cm4. These sets of configured matchers and the set of combined metric vectors are used to create the set of labelled data described in the next section.

Among the configured matchers, cm1 is chosen the default matcher. That is, any matching task that would not be recommended with the best configured matcher by the learned patterns will be matched using cm1. Therefore, any given matching task will be selected with cm1 for the following reasons: (i) The given matching task contains all the three characteristics and hence the learned patterns recommend cm1, and (ii) The learned patterns are unable to recommend the best matcher.

This is either due to a lack of information in the matching task or the set of metrics considered is insufficient.

Even though existing ontology matching systems such as the Falcon [18], COMA++[8], and Rimom [21] used the L+S (cm3) as a default matcher, the proposed approaches use the L+S+I (cm1) as a default matcher for the following reasons. First, when the learned patterns are unable, occasionally, to recommend the best configured matcher, the presence of the characteristics for the given matching task becomes ambiguous. In this case, deploying configured matchers like cm2, cm3 or cm4 would result in the loss of potential similar entity pairs, since these configured matchers fail to provide a match despite utilizing all the available information in the matching task. To circumvent this, therefore, cm1 is chosen as the default matcher, which would aid in improved effectiveness at the cost of efficiency. Second, though the number of matching tasks with the presence of instance information is moderate, it is not negligible. Further, the instances of the ontology are also vital information, since an entity pair with the same set of instances will have high chances of being similar. Also, the importance of the instance matcher is reinforced by the latest OAEI competitions which have released a new and separate dataset for instance matching.

3.3. Construction of the Labelled and Unlabelled Training Vectors

The proposed approaches require a set of configured matchers, labelled and unlabelled training vectors, to learn patterns using the two bootstrapping approaches. Unfortunately, the training vectors are unavailable and consequently a tedious construction process is carried out to create it, which is described in this section.

First, the process of labelled training vector construction is presented. A set of matchable ontology pairs is needed, along with the reference matching results obtained from the OAEI. In the OAEI, a set of ontology pairs belonging to a domain is called a track, and each ontology pair in a track is called a matching task. Our system needs matching tracks with numerous multitasks to create a set of training vectors. Hence, the tracks of the OAEI 2011[46] which had the maximum number of tasks is chosen, rather than the latest OAEI campaign. The list of tracks used, its corresponding number of matching tasks, and the domain of the tracks are listed in Table 1.

Table 1 Ontology Tracks.

Track	**Number of matching tasks**	**Domain of the matching tasks**
Benchmark	110	Bibliographic references
Conference	021	Conference
Benchmark2	102	Conference

The Benchmark [41] track comprises 110 matching tasks which are variants of the bibliographic ontology containing 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The Benchmark2 [42] track consists of 102 matching tasks and the Conference track [44] consists of 21 matching tasks which are variants of the conference ontology containing 74 classes and 33 object properties. These tasks were created such that any given matching system is checked for all possible matching scenarios and thus forms a perfect dataset with all the combinations of the characteristics of the ontologies.

The methodology for the construction of the set of labelled vectors LV, depicted in Figure 1, is as follows. For each matching task *t* from the sets of matching tasks T, a labelled vector *lv* is constructed using the following five steps: (i) Two metric vectors, one for each ontology of t, are obtained and combined into a single metric vector using the FS-A method. (ii) Next, *t* is matched using all the four configured matchers deployed by the COMA ontology matching tool [43] to obtain the four sets of resultant-matched entity pairs. (iii) The reference matching results of *t* obtained from the OAEI is used to compute the quality of the four sets of results using the F-Measure metric. (iv) Thereafter, the optimal configuration selector chooses the configured matcher with the maximum F-Measure as the best configured matcher for *t*. (v) Finally, the *lv* is constructed, consisting of the unique identifier (ID) of the matching task and the combined metric vector of *t*. Further, the *lv* is labelled with the best configured matcher obtained from the step above.

Using the methodology above, 233 labelled vectors are created from 233 matching tasks. A small sample of labelled vectors constructed from the Benchmark track is shown in Table 2 and the illustration is as follows. The first matching task (101, 262-6) consisting of 8 metric values has negligible structural and instance characteristics and is labelled cm2 (L). The second matching task (101,224) is quantified with fewer instance characteristics and is labelled cm3 (L+S). Similarly, the ontology pair (101,254-8) is labelled cm4 (L+I).

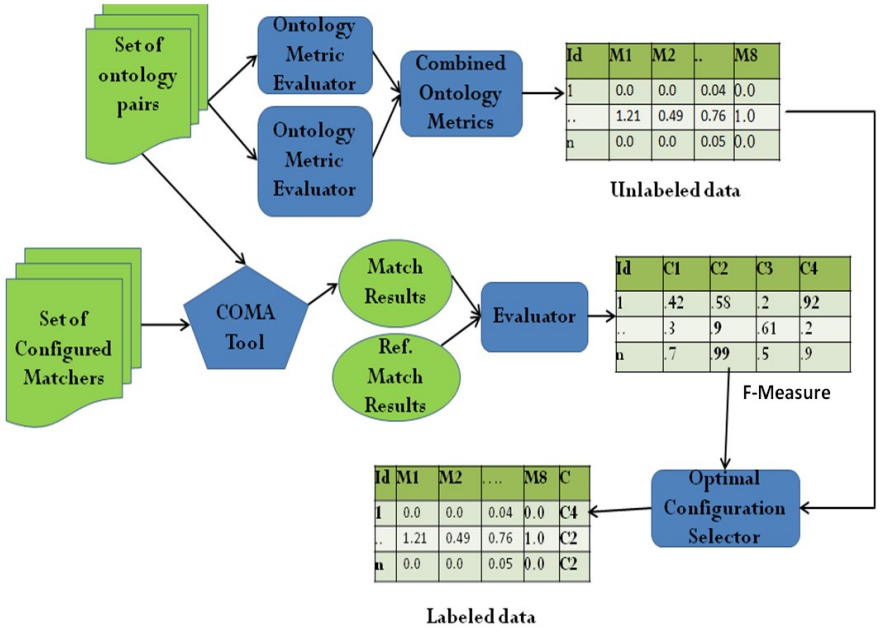

Figure 1 Unlabelled and Labelled Vectors Construction.

Second, the methodology for the construction of the set of unlabelled training vectors UV is outlined. It requires only the sets of matching tasks, since the data required for labelling such as a set of reference alignments and configured matchers are unnecessary. This results in the creation of new matching tasks without the reference alignments, in addition to the 233 matching tasks. For example,

let us consider the Benchmark track which comprises 110 matching tasks. In this track, an ontology named 101 is matched with itself and the rest of the 109 ontologies to create 110 matching tasks and the OAEI has provided the reference matching results for these 110 tasks. However, all the 110 ontologies belong to the same domain and so we plan to match each of the ontologies with all the other ontologies in this track, leading to 6105 matching tasks. Similarly, 231 and 5253 matching tasks are obtained for the Conference and Benchmark2 tracks. For each matching task *t*, created by the procedure above, an unlabelled vector *uv* is created which consists of the ID and the combined vector of *t*. In total, 11589 unlabelled vectors are created.

Table 2 Sample Labelled Vector.

Metrics	**Matching Tasks**		
	101, 262-6	**101, 224**	**101, 254-8**
Attribute Richness (AR)	0.0	1.2121	0.0
Relationship Richness (RR)	0.0	0.4898	0.0
Inheritance Richness (IR)	0.04	0.7575	0.05
Average Depth (AD)	0.5353	2.2424	0.5353
Class Richness (CR)	0.0158	0.0036	0.1013
Average Population (AP)	0.0	0.1088	1.8058
Concept External Structural Richness (CESR)	0.0025	1.0	0.003
Concept Internal Structural Richness (CISR)	0.0	1.0	0.0
Configured Matcher	cm2	cm3	cm4

4 Bootstrapping-based Pattern Construction

In this section, the two new seed pattern construction methodologies and the proposed two bootstrapped approaches, Bottom-up Bootstrapping (BUB) and Top-down Bootstrapping (TDB), are discussed in detail.

The flow of the proposed two semi-supervised bootstrapping approaches, BUB and TDB, can be split into training and testing phases. In the training phase, bootstrapping approaches need a small set of patterns and a large unlabelled vector. These patterns act as a seed used by the bootstrapping approaches to learn better patterns at each iteration and, as a result, these small sets of patterns are called seed patterns. The training phase can be divided into two major subprocesses: (i) seed pattern construction using the LV, and (ii) iteratively, learning the patterns from the seed patterns and UV. In the first subprocess, the sets of seed patterns corresponding to each of the configured matchers are constructed using the LV. Specifically, the seed patterns are constructed using two methods, the "2-seed pattern" and the "8-seed pattern", where two and eight (all) metrics are considered for each seed pattern construction respectively. In the second subprocess, the seed patterns and UV are used by the two bootstrapping approaches to iteratively construct new patterns that would enable an unknown matching task to be selected with the best configured matcher.

In the testing phase, two quality metric vectors representing the characteristics of the two input ontologies in the matching task are computed and combined using the FS-A aggregation method. The matching task is then selected with the best configured matcher using the learned patterns obtained

from either of the two bootstrapping approaches. To discover the matching entity pairs, the selected configured matcher is deployed using the configurable COMA [43] ontology matching tool. In the following subsections, the proposed two seed pattern construction methodologies and the corresponding BUB and TDB approaches-based pattern learning are detailed.

The logic behind both the proposed approaches is the "Curse of Dimensionality." This states that increasing the number of features with the static set of training samples to build a model may lead to an increase in classification errors. This increase is due to the more biased model construction towards the training vectors and hence fails to handle unknown samples, i.e., an overfitted model is constructed. Therefore, it is necessary to find the optimal set of features which best constructs the model. In our system, the metrics are the features and two premises are created to find the optimal set of features: (i) The premise, "A small subset of metrics is sufficient to form the optimal set of features", culminates in the formation of the BUB approach, and (ii) The premise, "A large subset of metrics is required to form the optimal set of features", results in the formation of the TDB approach. Based on these premises, the BUB needs to start with fewer metrics and so shorter seed patterns are needed, which is the underlying cause for developing the "2-seed pattern" method. Similarly, the TDB needs longer seed patterns and hence the "8-seed pattern" method is proposed.

4.1. Seed Pattern Construction

This section discusses in detail the two new seed pattern construction methodologies, the "2-seed pattern" and the "8-seed pattern". The structure of the set of seed patterns S_2 and $S_{8,}$ constructed using the "2-seed pattern" and the "8-seed pattern" methods, is as follows. Each of the seed patterns consists of a set of units where each unit (u_k) comprises the following three components: (i) a metric unique identifier (m), (ii) relational operators $\leq$ and $\geq$ (Θ), and (iii) a metric threshold value (v_m). Each constructed seed pattern,, s_2 and $s_{8,}$ is formally defined as follows.

$$s_2 = (cm, u_{1,} \ldots, u_k) \mid (s_2 \in S_2), (k = 2) \;\&\; (|S_2|=36) \tag{10}$$

$$s_8 = (cm, u_{1,} \ldots, u_k) \mid (s_8 \in S_8), (k = 8) \;\&\; (|S_8|=3) \tag{11}$$

$$\text{Given } u_k = (m \; \Theta \; v_m) \mid (1 \leq m \leq 8) \tag{12}$$

where *cm* represents the configured matcher for which the seed pattern has been constructed and k denotes the length of the pattern counted as the number of units. For each pattern s_2 constructed using the "2-seed pattern", the length of the pattern is preset as 2 (k = 2), since the BUB works under the premise that a small subset of metrics is sufficient. In the "2-seed pattern", totally 36 seed patterns are constructed, where each configured matcher has a set of 12 seed patterns except cm1, since it is the default matcher. Similarly, for each pattern s_8, the length of the seed patterns constructed using the "8-seed pattern" is 8 (k = 8), since the TDB works under the premise that all metrics are required. In the "8-seed pattern", totally 3 seed patterns are constructed, where each configured matcher has 1 seed pattern, except cm1.

Each seed pattern should be an initial representative of the corresponding cm and the construction methods are so designed. The steps of the "2-seed pattern" method, which constructs a set of seed patterns for each configured matcher *cm,* is as follows. (i) The set of metrics for each seed pattern of the *cm* should be chosen such that the presence or absence of structural and instance characteristics is quantified. The set of metrics for a seed pattern s_2 is selected by randomly choosing one metric from

the set of structural and instance characteristics metrics respectively. Similarly, for the rest of the *cm's* seed pattern construction, all the other possible pairs of metrics, one from each characteristic of metrics are chosen. (ii) Thereafter, the relational operator (≤ or ≥) for each chosen m of s_2 is determined, based on the cm. The basic idea here is to assign a ≥ operator for a metric *m*, if the characteristic of the matching task quantified by *m* should be checked for its presence. A characteristic is checked for its presence if the corresponding *cm* of the seed pattern has element matchers belonging to that characteristic. Else, m is assigned a ≤ operator to check for its absence. (iii) Finally, the threshold value of each chosen metric m can be calculated in two ways. The first is the Average method which computes the average value of the metric m from all vectors in the LV labelled with the configured matcher *cm*. The second method, MaxMin, selects the maximum value *(Max)* of the metric m from all the vectors in the LV labelled with the *cm*, if the logical operator of *m* is ≤. This is because only the *Max* of a metric represents the upper bound for the absence or insufficiency of a particular characteristic of the vector. Any value less than *Max* also represents absence or insufficiency. Similarly, choose a minimum value if the logical operator is ≥.

The set of metrics in the pattern is represented by unique identifiers such as 1 for RR, 2 for IR, 3 for AD, 4 for AR, 5 for CSER, 6 for CISR, 7 for AP and 8 for CR. A sample of the seed patterns formed is shown in Table 3. For example, let us outline the construction of the second seed pattern in the table. The randomly chosen metrics for this seed pattern are AR (4) from the structural characteristic metrics and CR (8) from the instance characteristic metrics. Next follows the determination of the relational operator. Since cm3 consists of L and S matchers, any given matching task t will be selected with cm3, if t has an adequate presence of the structural characteristic. Hence, the structural metric 4 is assigned a ≥ operator. Similarly, the instance metric 8 should be checked for its absence and hence a ≤ operator is assigned. Now, the seed pattern formed is "cm3, 4 ≥ threshold1, 8 ≤ threshold2". The value threshold1 is determined by taking the average or minimum value of the AR metric from the vectors belonging to cm3 from the LV. Similarly, threshold2 is determined by taking the average or maximum value of the CR metric from the vectors which belong to cm3, from the LV. Thus, for the rest of the cm3 seed pattern construction, all the other possible pairs of metrics are chosen and the corresponding relational operator and threshold values are learned.

Table 3 Sample Seed Pattern for the BUB Approach.

Configured Matcher	**Metrics and their threshold**	
cm2	3 ≤ 0.54	8 ≤ 0.016
cm3	4 ≥ 0.64	8 ≤ 0.006
cm4	3 ≤ 0.54	7 ≥ 1.81

The seed pattern formation using the "8-seed pattern" method is as follows. For each configured matcher, one seed pattern with all the structural and instance metrics is constructed. Subsequently, the relational operator and threshold value for each metric are determined, similar to the "2-seed pattern" method. Three seed patterns, one for each of the configured matchers except cm1 is constructed, which are shown in Table 4. After the set of necessary seed patterns is constructed, these patterns are iteratively processed by the bootstrapping approaches to construct the new sets of patterns discussed in the following subsection.

Table 4 Set of Seed Patterns for the TDB Approach.

CM	Metrics and its threshold							
cm2,	$1 \leq 0.0$,	$2 \leq 0.0$,	$3 \leq 0.54$,	$4 \leq 0.0$,	$5 \leq 0.0$,	$6 \leq 0.0$,	$7 \leq 0.0$,	$8 \leq 0.016$
cm3,	$1 \geq 0.26$,	$2 \geq 0.62$,	$3 \geq 1.57$,	$4 \geq 0.64$,	$5 \geq 0.68$,	$6 \geq 0.82$,	$7 \leq 0.075$,	$8 \leq 0.006$
cm4,	$1 \leq 0.0$,	$2 \leq 0.0$,	$3 \leq 0.54$,	$4 \leq 0.0$,	$5 \leq 0.0$,	$6 \leq 0.0$,	$7 \geq 1.81$,	$8 \geq 0.101$

4.2. Bottom-up and Top-down Bootstrapping Approaches

This section details the two new proposed BUB and TDB approaches to construct new patterns from the seed patterns. In both the approaches, for each configured matcher *cm*, a set of patterns is learned which represents the required characteristics by any matching task to be matched using the *cm*. The structure of the set of learned patterns *P* is similar to the set of seed patterns *S* which is given below.

$$p = (cm, u_{1,} ..., u_k) \mid (p \in \mathrm{P}), (2 \leq k \leq 8) \ \& \ (0 < |\mathrm{P}| < \infty) \qquad (13)$$

where *p* represents a pattern in *P*, *cm* represents the configured matcher and u_k is defined in (12). But, dissimilar to seed patterns, the length (k) of the learned patterns can vary from 2 to 8.

In both the approaches, new patterns are iteratively learned from the seed patterns used for automatic matcher selection. However, the approaches through which the patterns are constructed are contrasting and the major differences between them are as follows: (i) The BUB approach starts with generic seed patterns (fewer units) and builds more specific patterns for each iteration by appending a new metric. Conversely, the TDB approach starts with specific seed patterns (more units) and builds more generic patterns by masking the metrics at each iteration. This is analogous to the bottom-up and top-down clustering algorithms and, consequently, the proposed approaches are so named. (ii) Both the approaches learn new patterns based on the new information obtained from the unlabelled metric vectors matching the set of seed patterns at each iteration. For the BUB approach, in the initial iterations, the number of matched unlabelled vectors is greater when compared to the later iterations, since the seed patterns are generic in the initial iterations. Contrastingly, in the TDB approach, in the initial iterations, the number of matched vectors is lesser as a result of more specific seed patterns. Hence, the number of matched vectors used for new pattern construction at each iteration differs in each of these methodologies, ultimately influencing the quality of the patterns learned.

The pseudocode of both the proposed bootstrapping approaches is presented in Figure 2. The input to the BUB approach is the set of seed patterns S_2 and the set of unlabelled vectors UV. Similarly, for the TDB approach, the inputs are S_8 and UV. The output for both the approaches is a set of new patterns, P, which are used for the selection of configured matchers in the testing phase. Each iteration of both the proposed bootstrapped approaches consists of the following steps: (1) initializing, (2) matching, (3) updating the seed patterns, (4) constructing the new patterns, and (5) scoring the new patterns. All the five steps, except the 4^{th} , is common for both the approaches. In all the common steps, s_x denotes a seed pattern, S_x represents the set of seed patterns and *'x'* can be 2 or 8. The s_x and S_x denote the s_2 and S_2 respectively, if the common steps in the approaches are executed for the BUB approach. Similarly, it denotes s_8 and S_8 for the TDB approach. A detailed description of each step is as follows.

Input:

(i) Set of seed patterns S_2 for BUB or S_8 for TDB

(ii) Set UV

Output:

Set of new patterns P obtained by growing S_2 for BUB or obtained by masking S_8 for TDB

I. INITIALIZE

a. Assign set CV, new_metric[3][8] and set this_P to NULL

b. **BUB approach:** $P = S_2$

TDB approach: $P = S_8$

II. MATCHING

a. For each uv ∈ UV

For each $s_x \in S_x$

If Matches (uv, s_x), then

(i)CV = CV U {uv , cm of s_x }

(ii) Break

b. If none of the uv matches, then

Goto step 5 for **BUB approach**

Goto step 4 for **TDB approach**

III. UPDATE THE SEED PATTERNS

For each cm ∈ CM

a. For each m

new_metric[cm][m] = Update (m, fetch (CV, cm)

b. For each $s_x \in S_x$

If ((label(s_x)) = cm)

(i) s_x = pattern_updation(s_x, new_metric[cm][])

(ii) P = P U s_x

IV. NEW PATTERN CONSTRUCTION

BUB approach

If(∀ $s_2 \in S_2$ (length (s_2) = 8)) then Goto SCORING //Stopping criteria

Else

For each cm ∈ CM

For each $s_2 \in S_2$ | length (s_2) < 8 & ((label(s_2)) = cm)

For each m ∈ M

If (not_partof(s_2, m))

(i) New_pattern = pattern_growing (s_2, m, new_metric[cm][m]);

(ii) this_P = this_P U (New_pattern);

S_2 =this_p

P = P U this_p

TDB Approach

If($\forall$ $s_8 \in S_8$ | length (s_8) = 2) then Goto SCORING //Stopping criteria

Else

For each $s_8 \in S_8$ | length (s_8) >2

For each m in s_8

(i) New_pattern = pattern_masking (s_8,m);

(ii) this_P = this_P U (New_pattern);

S_8= this_P

P = P U this_p

Goto INITIALIZE

V. SCORING

a. For each p ∈ P

Scoring (p) = F-Measure (p)

b. For each cm ∈ CM

Choose Top K patterns based on Scoring(p).

Figure 2 Pseudocode of the Proposed Bootstrapping Algorithm.

4.2.1. Initialization

In the first step, the new data structures introduced in these approaches such as the set CV (configured vectors), the *new_ metric [3][8],* and the_P set are initialized to NULL at each iteration. For each iteration, the set CV is used to store the *uvs* which are labelled, the *new_metric[3][8]* is used to store the updated 8 metric values for all the 3 *cm*s, and the new patterns learned are stored in this_P. In the testing phase, the set of seed patterns S_x is also used for the selection of the best configured matcher and is stored in set P.

4.2.2. *Matching*

The basic idea behind step 2 is to create new labelled vectors for each configured matcher *cm* from the set of large unlabelled vectors UV obtained from section 3.3. This is done using the seed patterns which represent the simple constraint for each *cm*. These newly-created labelled vectors are called configured vectors (*cv*). This is accomplished in step (2.a), where each unlabelled vector *uv* from the UV is matched with every seed pattern s_x from S_x until a match is found. The function Matches (*uv*, s_x) checks whether the *uv* has the required characteristic represented by the s_x. If a match is found, a new *cv* is created by labelling the *uv* with the configured matcher *cm* of the s_x, and the *cv* is stored in the set CV. This process is repeated for all the *uv*s in the set UV to obtain the best configured matcher for each *uv* based on the S_x.

For example, let us consider the second unlabelled metric vector uv=(1.2121, 0.4898, 0.7575, 2.2424, 0.0036, 0.1088, 1.0, 1.0) from Table 2 and s_2 as <cm3, 1 ≥ 0.64, 7 ≤ 0.006>. The required characteristics of cm3 represented by this seed pattern are that Attribute Richness (1) should be greater than or equal to 0.64 and Class Richness should be less than or equal to 0.006. The given *uv* has the required characteristic and hence matches s_2. The configured vector (cm3, 1.2121, 0.4898, 0.7575, 2.2424, 0.0036, 0.1088, 1.0, 1.0) is added to the set CV. For each iteration at the end of step 2.a, a new set of configured vectors CV is constructed which becomes the labelled training vector representing

the required characteristics for each *cm*. These new, informative configured vectors are used in the following steps to compute the updated thresholds of all the metrics which will aid in new pattern construction.

It is also crucial to note that, at the end of each iteration, the new patterns stored in this_P become the seed patterns for the consecutive iteration. Therefore, in the BUB approach, after certain iterations in step 2.b, the *uv* does not match with the s_x, as seed patterns become too specific. Consequently, the set CV which is needed for both "seed pattern updation" and new pattern construction is empty and so the approach stops and switches to the "scoring" step. Contrastingly, in step 2.b of the TDB approach, the set CV is empty in the initial iterations. Despite this, the TDB approach can form new patterns by masking and thus in this scenario the approach switches to the "new pattern construction" step.

4.2.3. *Updating the Seed Patterns*

As a general rule, in supervised ML approaches, the model and its parameters are constructed from labelled training data. Here, the labelled training data are the CV, the model is the pattern, and the parameters are the metrics and their threshold values. In this step, the new threshold values are learned using the newly-constructed CV from the step above, which is used for the construction of new patterns. Also, the seed patterns are made more accurate by replacing old threshold values with newly-learned threshold values.

First of all, in step 3.a, the new threshold values are learned for each *cm*. Two functions, the fetch() and Update(), are introduced. The fetch(CV, cm) function is used to retrieve the *cv*s from the CV labelled *cm*. The Update(m, fetch(CV, cm)) function is designed to learn the new threshold value for the metric *m* by the Average or MaxMin methods. The vectors (cvs) needed for these methods are retrieved using the fetch() function. The process above is repeated for all the *m* in each *cm* to learn new values. The new values of all the 8 metrics are stored in an array *new_metric[][]* where the *new_metric[cm][m]* contains the new value of the m^{th} metric corresponding to the *cm*.

For the BUB approach, the newly-computed threshold values become more specific in comparison with the threshold values of the previous iterations, the reason being as follows. Initially, each *cm* has a larger number of *cv*s since the seed patterns are generic, with just two units. As the iterations increase, the seed patterns grow in length with more metrics and so the number of unlabelled vectors matching the seed patterns under each *cm* reduces. Simultaneously, the number of un-configured vectors increases and the threshold values computed with this new and comparatively small set of CVs become more specific, compared to the previous iterations. Therefore, in each iteration, the updated seed patterns and the newly-constructed patterns which use these more specific threshold values also become more specific. This scenario is reversed for the TDB approach, where it begins with the most specific threshold values and ends with generic threshold values.

After the new threshold values are learned for each *cm*, they are used to update the seed patterns in step 3.b. The function label(s_x) is introduced, which returns the *cm* of s_x. For each *cm*, the corresponding seed patterns are first retrieved using the label() function. Thereafter, the function pattern_updation (s_x, *new_metric[cm][]*) replaces the thresholds of these seed patterns with the newly-computed values which are stored in the *new_metric[cm][]*. These updated seed patterns are also new patterns which are stored in the set of new patterns, P.

4.2.4. *New Pattern Construction*

The next step is the "new pattern construction" where new patterns are constructed from the updated seed patterns and new threshold values obtained from step 3. Here, the terminology seed pattern represents the updated seed pattern obtained from the step above.

In the BUB approach, new patterns are constructed by appending new metrics to the seed patterns. This step first checks for the stopping criteria (IF part). The iterative process will stop when the length of all the seed patterns is 8. This is because, after a certain number of iterations, the entire set of seed patterns consists of all the metrics and it is impossible to append new metrics to form new ones. Henceforth, the approach switches to the scoring step. If the stopping criteria fail, for each cm, a set of seed patterns whose length is less than 8 is chosen. A seed pattern s_2 with a length less than 8 indicates that certain metrics are missing in s_2 and these missing metrics can be appended to make it grow. For each such growable seed pattern s_2 under each *cm*, the missing metrics are identified using the function not_partof(s_2,*m*) which returns 1 if s_2 does not contain the metric *m*. Now the new patterns are constructed by appending the missing metrics to the growable seed patterns using the function pattern_growing (s_2, *m, new_metric[cm][m]*). This function takes as input a selected seed pattern s_2, as well as one of the missing metrics, *m,* and the new updated metric threshold value for *m* which is stored in the *new_metric[cm][m]*.A new pattern is created by appending s_2 with *m* and its threshold value. Further, for each added metric *m*, the corresponding relational operator is determined based on the *cm* of $s_{x,,}$ similar to the seed pattern construction methodology. Likewise, the other missing metrics of s_2 are also appended to create a set of new patterns from s_2.

In the Top-down approach, the new patterns are constructed by masking, which is defined as follows. The term, 'masking a metric *m',* is defined as removing a metric *m* along with its relational operator and threshold value from the pattern. A new pattern is formed from a seed pattern by masking a metric *m* from the seed pattern. Similar to the BUB, this step also first checks for the stopping criteria (IF part). The iterative process will stop when the length of all the seed patterns is 2. This is because, after a certain number of iterations, the entire set of seed patterns will consist of only two metrics (the length of all the seed patterns being 2) and new patterns cannot be formed. In this scenario, the approach stops the iteration and switches to the scoring step. If the stopping criteria fail, for each cm, a set of seed patterns whose length is greater than 2 is chosen. A seed pattern s_2 with a length greater than 2 indicates that certain metrics can be removed to make it shrink. For each such shrinkable seed pattern s_8 under each *cm* with length k, the function pattern_masking (s_8,*m*) is used to mask each metric *m* from the *k* metrics, one at a time, to create *k* new patterns.

The newly-constructed patterns using the BUB and TDB approaches are called BU and TD patterns respectively. In both the approaches, the set of learned new patterns in this step becomes the set of seed patterns for the next iteration. Also, this set is stored in the set P and the control switches to the "initialize" step to repeat the iteration.

4.2.5. *Scoring*

Finally, in the "scoring" step, all the newly-constructed patterns in P are validated using the metric F-Measure. To compute the F-Measure values for each *p* in P, a set of reference metric vectors labelled with the best *cm* is required. The set LV is used as the reference dataset, since each $lv \in$ LV is labelled

with the best *cm* using the construction methodology described in section 3.3. The F-Measure of each pattern *p* is the harmonic mean of the precision (correctness) and recall metrics (completeness) of *p*.

Before defining the metrics, let us first describe the two sets used in the metrics, CL and PL. Let CL(*p*) be the set of $lv \in$ LV such that label(*lv*) = label(*p*). Basically, each *lv* in CL(*p*) is the sample metric vector for *p* and hence an ideal *p* should match with all the *lv*s in the CL(*p*). Also, let PL(*p*) be the set of the $lv \in$ LV which actually matches (step II) with p according to step II of the bootstrapping approaches. In other words, CL(*p*) is the ideal set of matched vectors and PL(*p*) is the actual set of matched vectors. Further, an *lv* is stated as correctly classified by *p* if the label (*cm*) of *lv* is the same as the *cm* of *p*. The formal definition of precision and recall for *p* are as follows.

$$\text{Precision(p)} = \frac{|\text{CL(p)} \cap \text{PL(p)}|}{|\text{PL(p)}|} \qquad (14)$$

$$\text{Recall(p)} = \frac{|\text{CL(p)} \cap \text{PL(p)}|}{|\text{CL(p)}|} \qquad (15)$$

where $|\text{CL(p)} \cap \text{PL(p)}|$ represents the number of vectors correctly classified by the pattern |PL(*p*)| and |CL(*p*)| denotes the number of patterns in the set PL(*p*) and CL(*p*) respectively.

Based on the precision and recall metrics defined above, the F-Measure for each pattern *p* is computed. Similarly, the F-Measure is computed for all the patterns under each *cm*. For each *cm*, the top K patterns are chosen based on the F-Measure which represents the required characteristic for the *cm*. These sets of chosen patterns are used to select the best *cm* for any given matching task.

5 Evaluation & Experimental Results

This section details the various experiments conducted to evaluate the proposed structural metrics, the BUB and TDB approaches. For all the experiments, the three ontology tracks used are mentioned in Table 1 and a detailed description presented in section 3.3. Also, for all the experiments conducted to prove the effectiveness of the patterns learned, a 10-fold cross validation is performed using the LV dataset which is used both as a training and testing dataset. In addition to the LV dataset, the UV dataset is also used as a training dataset for pattern learning.

The set of elements and configured matchers required for the experimental set-up can be deployed by any ontology matching tool, provided the tool is customizable. In all the experiments, the configured matchers are deployed using the configurable ontology matching tool COMA [8]. This tool is selected because: (i) It is a configurable tool where the matchers can be chosen based on the requirements of the user/task, (ii) It has been used in many OAEI competitions and reported good results, and (iii) The latest version, COMA 3.0 [43], is freely downloadable.

The set of element matchers available in COMA 3.0 is as follows. The available linguistic element matchers are Affix, Trigram, EditDistance, Soundex, Synonym, Type and Reuse. A structural element matcher called Statistics is available to quantify similarity, based on the number of children, parents, leaves, etc. Also, an instance matcher called Instance is available. These element matchers, except the instance matcher, are aggregated in different combinations to define 8 combined matchers: Name, NameType, NameStat, NamePath, Children, Leaves, Parents and Siblings. The first 4 combined matchers are designed for linguistic matching and the last 4 for structural matching. Using these 8 combined matchers and the instance matcher, the required configured matchers for the proposed

approaches are deployed. For cm1, all the combined matchers and instance matcher are used. Similarly, cm2 employs the first four combined matchers, cm3 uses all the combined matchers, and cm4 deploys the first four combined matchers and the instance matcher. The default parameters of the COMA tool, such as the Average Combination and Delta selection, are used. For a detailed description of the COMA tool and its default parameter settings, readers are referred to [8].

5.1. Proposed Semi-Supervised Approaches vs Existing Supervised Approaches

In this subsection, the two proposed semi-supervised approaches, BUB and TDB, are first compared for the effectiveness of the patterns constructed, following which the two approaches are evaluated against the existing supervised approaches.

Before comparing the two proposed approaches, the Average and MaxMin methods used to calculate the threshold of the patterns are evaluated, based on the effectiveness of the patterns constructed. As discussed in section 4.2.5, the effectiveness of each pattern is measured by the F-Measure metric. The average F-Measure of the top 2, 5 and 10 patterns is computed for each configured matcher. Based on the effectiveness of the constructed patterns, it is inferred that the MaxMin method is able to stringently and precisely compute the threshold values for each metric in comparison with the average method. This is shown in Tables 5 and 6, where the MaxMin threshold method achieves better effectiveness in comparison with the average threshold method for both the BU and TD patterns.

Table 5 Average F-Measure of the TopK Patterns for Each Configured Matcher based on the BUB Approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.879	0.768	0.924	0.69	0.686	0.835	0.567	0.632	0.799
MaxMin	0.923	0.804	1	0.778	0.799	0.917	0.721	0.797	0.819

Table 6 Average F-Measure of the TopK Patterns for Each Configured Matcher based on the TDB approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.947	0.693	0.925	0.867	0.643	0.905	0.747	0.583	0.715
MaxMin	1	0.779	0.935	0.77	0.763	0.88	0.756	0.732	0.795

To have a clear insight into the effectiveness of the patterns, the precision and recall values of the TopK patterns listed in Tables 5 and 6 are shown in Tables 7-10 respectively. It is to be noted that, on average, the precision values of the patterns are greater than the recall values. This is because the patterns are learned stringently to accurately choose the correct configured matcher leading to better precision and would occasionally result in missing out on the ontology pairs that are supposed to be matched by the configured matcher of the patterns leading to reduced recall. However, the majority of such ontology pairs are matched by the default matcher cm1 and hence the effectiveness of the matching results of the pair are not compromised. Further, in fewer cases, the precision or recall values of Average method are greater than MaxMin method due to the more stringent process of latter compared to the former leading to rare overfitted thresholds. Nevertheless, in the majority of the cases, the MaxMin method achieves better results due to the precise process and hence the average effectiveness is higher. Consequently, in further processing of the proposed approaches, the MaxMin method is used to determine the thresholds of the patterns.

Next, a brief analysis of the selected set of BU and TD patterns is presented. Tables 11 and 12 show the set of selected patterns for each of the configured matchers. The selection procedure of patterns from the set of patterns learned is as follows. For each configured matcher, one pattern with the maximum F-Measure is selected, i.e., the Top 1. If more than one pattern has the same F-Measure with a different pattern length, the shortest pattern is selected since it represents generalized and efficient modelling. If more than one pattern has the same F-Measure with the same pattern length, a pattern is selected at random.

Table 7 Average Precision of the TopK Patterns for Each Configured Matcher based on the BUB approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.9	0.83	0.96	0.74	0.73	0.82	0.58	0.69	0.85
MaxMin	0.97	0.86	1	0.79	0.86	0.95	0.79	0.83	0.8

Table 8 Average Recall of the TopK Patterns for Each Configured Matcher based on the BUB approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.86	0.71	0.89	0.65	0.65	0.85	0.55	0.58	0.75
MaxMin	0.88	0.75	1	0.77	0.75	0.89	0.66	0.77	0.84

Table 9 Average Precision of the TopK Patterns for Each Configured Matcher based on the TDB approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.97	0.77	0.93	0.92	0.67	0.95	0.77	0.56	0.73
MaxMin	1	0.83	0.97	0.8	0.74	0.87	0.79	0.79	0.82

Table 10 Average Recall of the TopK patterns for Each Configured Matcher based on the TDB approach.

	Top 2			**Top 5**			**Top 10**		
	cm2	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**	**cm2**	**cm3**	**cm4**
Average	0.93	0.63	0.92	0.82	0.62	0.86	0.73	0.61	0.7
MaxMin	1	0.73	0.9	0.74	0.79	0.89	0.72	0.68	0.77

Table 11 Selected BU patterns.

Configured Matcher	**Precision**	**Recall**	**F-Measure**	**Patterns**		
cm2	0.97	0.87	0.92	$3 \leq 0.64$,	$7 \leq 0.12$	
cm3	0.89	0.78	0.83	$6 \geq 0.82$,	$8 \leq 0.006$	
cm4	1	1	1	$4 \leq 0.12$,	$8 \geq 0.1$,	$3 \leq 2.2$

Table 12 Selected TD patterns.

Configured Matcher	**Precision**	**Recall**	**F-Measure**	**Patterns**			
cm2	1	1	1	$1 \leq 0.0$,	$4 \leq 0.05$,	$3 \leq 2.2$,	$8 \leq 0.1$
cm3	0.8	0.72	0.76	$5 \geq 0.0$,		$8 \leq 0.1$	
cm4	1	0.9	0.95	$4 \leq 0.0$,	$2 \leq 0.53$,	$8 \geq 0.1$,	$7 \geq 0.5$

The following observations are made from the set of patterns selected above:

1. The BU pattern of cm2, with a superior effectiveness of 0.92, is a seed pattern. Similarly, cm3's BU pattern has the same length and metrics like a seed pattern, with a difference in threshold values. Thus, it is evident that the set of constructed seed patterns is sufficiently adequate to be the base patterns for the construction of new patterns.

2. The list of metrics such as Average Depth (3), Attribute Richness (4), Concept Internal Structural Richness (6), Average Population (7) and Class Richness (8) are present in the selected BU patterns. Similarly, the TD patterns comprise a list of metrics such as Relationship Richness (1), Inheritance Richness (2), Average Depth (3), Attribute Richness (4), Concept External Structural Richness (5), Average Population (7) and Class Richness (8). From these lists, it is inferred that the number of metrics needed for a configured matcher selection by the BU patterns is 5 and TD patterns is 7. Hence, the BUB approach models the selection process with a smaller number of metrics and is therefore more efficient than the TDB approach. Basically, the BU patterns are small, generic patterns compared to the TD patterns.
3. On average, for all the three configured matchers, the effectiveness (F-Measure) of the selected BU patterns is 0.916 and TD patterns is 0.903. Consequently, it is inferred that the BU patterns achieve marginally more effective patterns with fewer metrics in comparison with the TD patterns with more metrics.

From the analysis above, it is concluded that the BU patterns are better than the TD ones. The BU patterns are effective and generic with a smaller number of metrics, compared to the TD patterns which are specific and with more metrics. Also, the proposed CESR and CISR metrics are present in both the set of selected patterns demonstrating the significance of the proposed metrics in the selection process of the best configured matcher. From this point onwards, the terms BU and TD patterns denote only the selected top patterns shown in Tables 11 and 12.

The third experiment evaluates the effectiveness of the BU and TD patterns in selecting the best *cm* for each matching task of all the three ontology tracks in Table 1. Following this, the selected *cm* is deployed using the COMA tool and the matching results obtained for all the matching tasks are compared with the reference alignments to compute the precision, recall and F-Measure metrics (Figures 3 and 4). As inferred from the figures, the BU patterns have achieved better effectiveness (F-Measure) than the TD patterns. Although the TD patterns use more metrics to quantify the characteristics of the ontology pair, it is unable to construct effective patterns owing to the "Curse of Dimensionality" effect discussed earlier. From these results and inferences from the learned patterns, it can be concluded that the premise for the BUB approach, "A small subset of metrics is sufficient to form the optimal set of features" is true.

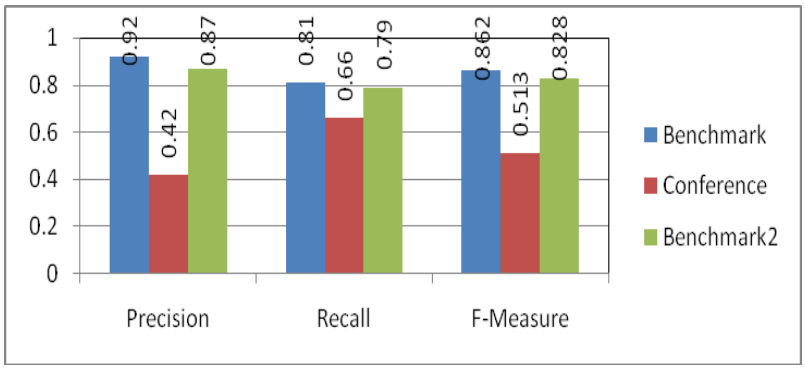

Figure 3 Precision, Recall and F-Measure of the Matching Task using BU Patterns.

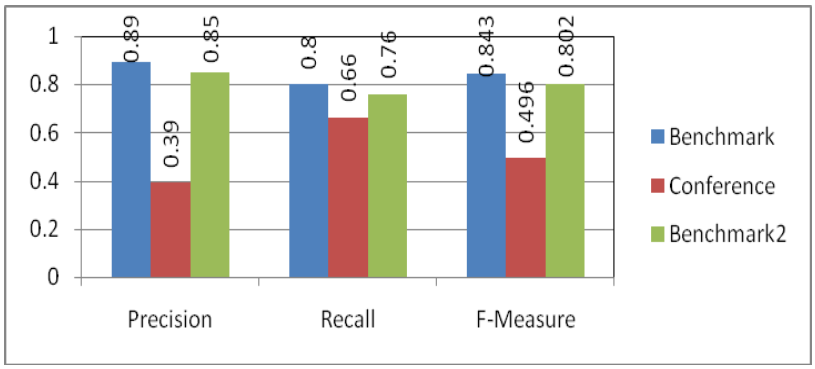

Figure 4 Precision, Recall and F-Measure of the Matching Task using TD Patterns.

Further, to prove that the proposed approaches can surpass any supervised machine learning (ML) algorithms, well-known supervised ML algorithms belonging to miscellaneous categories such as the instance-based learner KNN, probability-based Naive Bayes, decision tree C4.5 and the neural network-based multilayer perceptron are used for comparison (Figure 5). As similar to the first experiment, all the approaches are evaluated using the average precision, recall and F-Measure of the matching tasks belonging to all the three tracks and the selected configured matcher is deployed using the COMA 3.0 [43] tool. It is also essential to note that the supervised KNN model evaluated in this subsection, and the KNN model of Cruz et al. [7], are the same, the difference being that the latter uses an ontology matching tool, AgreementMaker, to deploy the selected configured matchers, whereas the former uses the COMA 3.0 tool and, additionally, utilizes two new structural metrics. Furthermore, the AgreementMaker tool used by Cruz et al. is not openly available and so the experimental results of their work can only be cited from their paper. Since the Cruz et al. work's F-Measure values for all the tracks are unavailable in their paper, the supervised KNN with the COMA 3.0 tool is used for simulation. So, when the proposed approaches are compared with the generic supervised KNN approach, an indirect comparison with the Cruz et al. [7] work is drawn.

The results from Figure 5 show that the proposed BU patterns are superior in choosing the best configured matchers, compared to the proposed TD patterns and the existing supervised models. This result is due to the three reasons that follow: (i) The two proposed structural metrics, CESR and CISR, aided the system in better selection by precisely quantifying the structural characteristic of any given ontology. (ii) Though both the BU and TD patterns use the proposed structural metrics, the BU patterns achieved better accuracy, largely because the BUB approach is able to choose the necessary (and sufficient) set of metrics needed to learn generic and effective patterns. (iii) The BUB approach is able to construct more accurate patterns than the KNN model, since the proposed approach used a small labelled and a large unlabelled vector, whereas the KNN model used only a little labelled vector for pattern construction. It can also be noted that the KNN achieved the top mean F-Measure of 0.683 among the supervised algorithms, the reason being that, instance-based algorithms can learn better models with a smaller set of training data, whereas other supervised algorithms require larger training data.

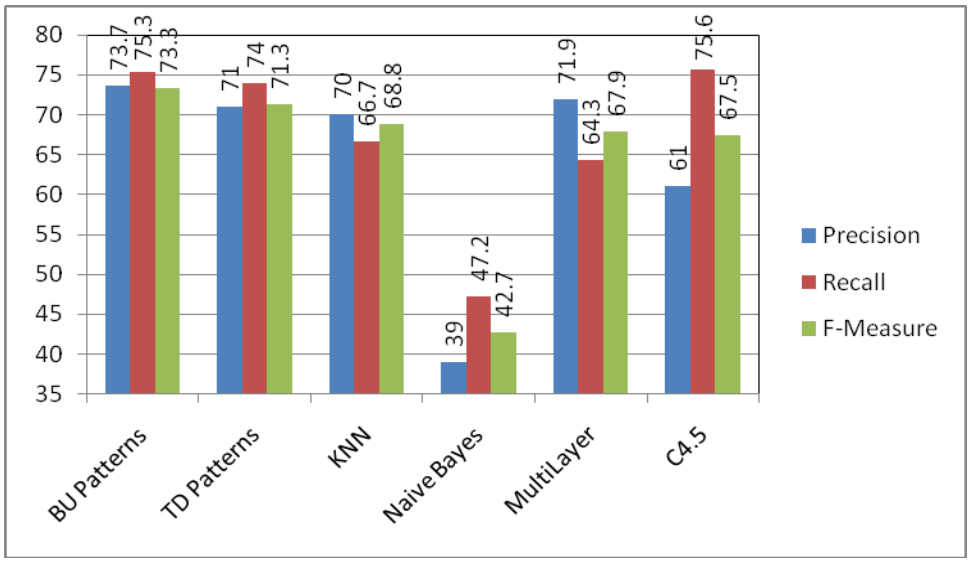

Figure 5 Precision, Recall and F-Measure of the Matching Tasks using the Proposed and Various Existing ML Approaches.

The next experiment is carried out to test the individual effectiveness of the proposed BU patterns, as well as the structural metrics CESR and CISR, using precision, recall and F-Measure metrics. The methodologies compared are the proposed BU patterns which achieved better effectiveness than the TD patterns in the three experiments above, and the supervised KNN model which achieved the best accuracy among the supervised approaches. As shown in Figures 6-8, four models are evaluated. The first model, M1, is the baseline consisting of the KNN which uses only the set of existing metrics listed in section 3.1. For all the three tracks, this model achieves the least average precision, recall and F-Measure, since the supervised KNN is unable to build an accurate model with a small labelled dataset. The second model, M2, consists of the KNN with the existing and proposed metrics. The latter aided in better selection by computing the percentage of the internal (CISR) and external structural richness (CESR), and so M2 achieves better effectiveness than M1, validating the contribution of the proposed metrics towards a better selection of configured matchers. The third model, M3, consists of the BU patterns with the existing metrics. Since M3 is a semi-supervised model, it can construct an accurate model with a small set of labelled and a large set of unlabelled datasets, compared to M1 and M2. Therefore, M3 achieves greater precision, recall and F-Measure than M1 and M2, even without the proposed metrics. The final model, M4, is the proposed BUB approach comprising the BU patterns with the existing and proposed metrics which achieves the best average effectiveness for all the three tracks. This is due to the precise characteristic quantification by the proposed metrics and the accurate modelling by the BU patterns. From the set of experiments, it can be concluded that the proposed BU patterns learned using the BUB approach are generic, as well as more effective and accurate, than the TD patterns and existing supervised ML algorithms.

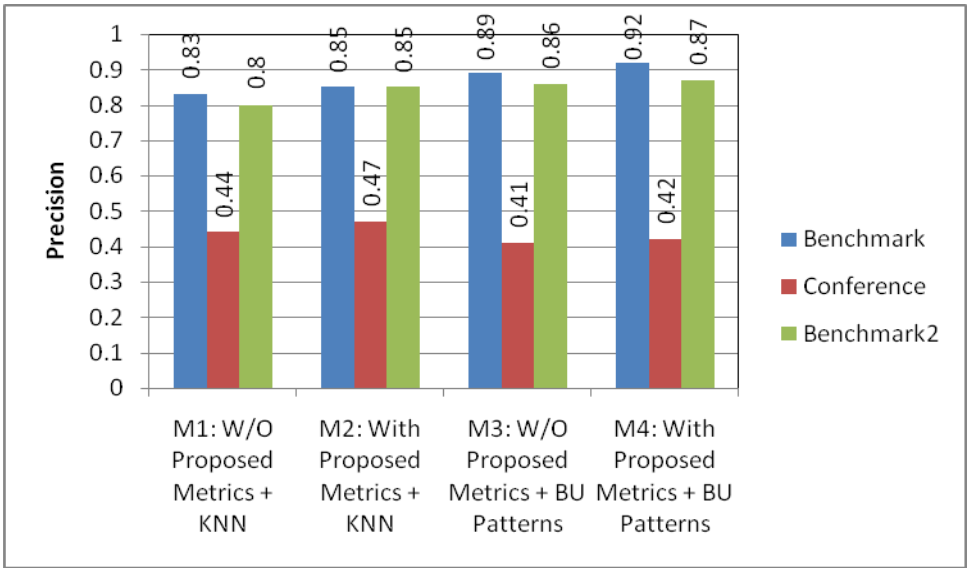

Figure 6 Precision of the Matching Tasks for the Versions of the Proposed Approach.

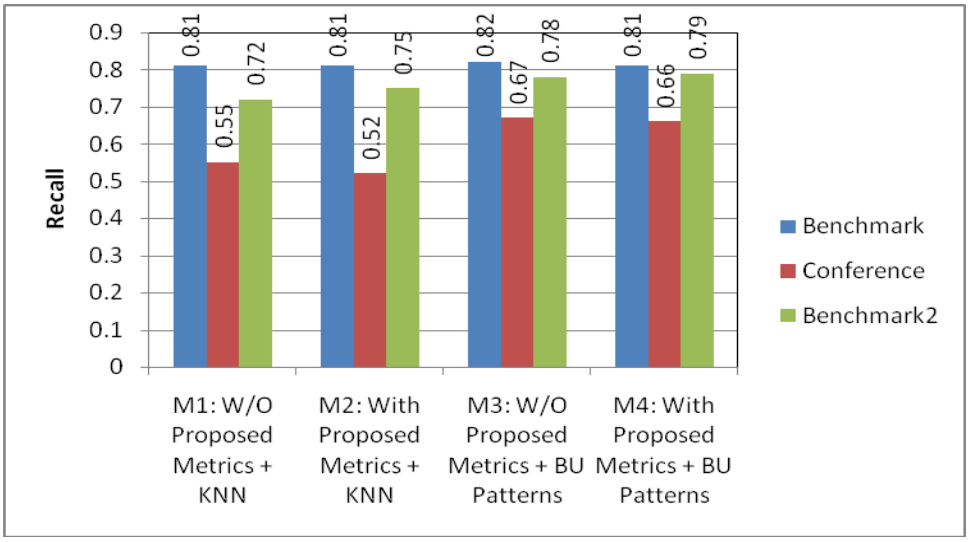

Figure 7 Recall of the Matching Tasks for the Versions of the Proposed Approach.

5.2. One-way ANOVA: A Statistical Significance Test

This section briefly discusses the statistical significance test carried out using the one-way ANOVA [45] to statistically demonstrate the performance of the BU and TD patterns in terms of the F-Measure of the matching tasks. The supervised KNN model which obtained the maximum accuracy among the supervised ML approaches is used for comparison. Here, the ANOVA is used to prove that the improvements in the performances of the proposed approaches as against the supervised KNN model are statistically significant.

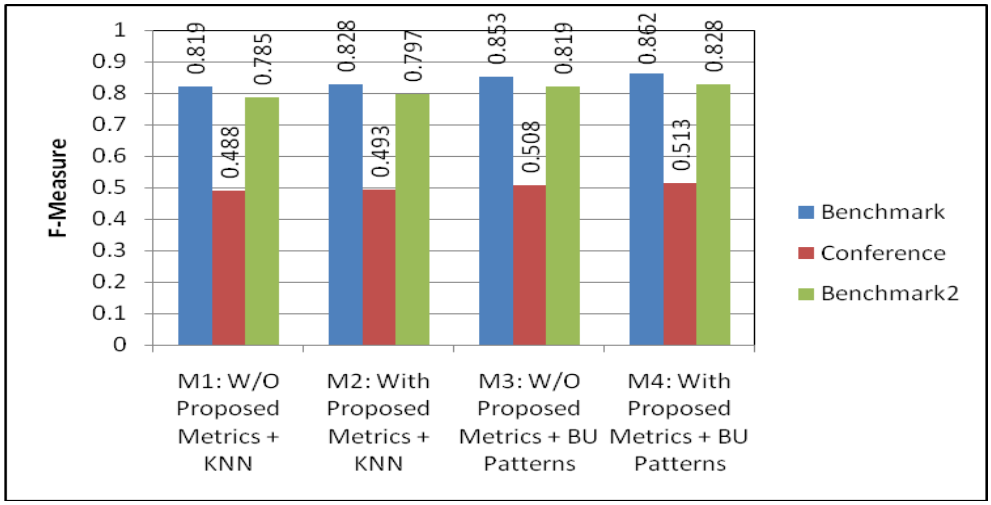

Figure 8 F-Measure of the Matching Tasks for the Versions of the Proposed Approach.

Before commencing the statistical evaluation, the data required for the performance evaluation and null hypothesis is to be defined. The dataset used for the analysis consists of the F-Measure values of the matching tasks belonging to the tracks mentioned in Table 1. For each matching task, the F-Measure values corresponding to each of the three methods are computed and the dataset analysed by the ANOVA to identify the statistical differences among the methods. Now, the null hypothesis is defined thus: "The F-Measure means of all the three methods are equal".

Initially, the ANOVA test is carried out for the data belonging to the Benchmark track and Table 13 shows its descriptive statistics. For each method, 110 (N) matching tasks are matched using the selected configured matcher and the corresponding F-Measure computed. The means and standard deviations of the three methods clearly indicate that the BU patterns achieve both maximum and consistent performances. Table 14 shows the analysis of the variance, from which it is inferred that the null hypothesis is rejected with the risk of just 0.01%, as indicated in the last column of the table (pr>F). This indicates that all the three methods are statistically different from each other, and the improvement achieved by the proposed approaches is significant.

Table 13 Descriptive Statistics.

Level	N	Mean	StDev
BU	110	86.2324	3.732038
TD	110	79.91667	4.024341
KNN	110	84.30901	4.082548

Table 14 Analysis of variance.

Source	**DF**	**Sum of squares**	**Mean squares**	**F**	**Pr > F**
Model	2	2288.033	1144.017	131.493	< 0.0001
Error	328	2844.964	8.700		
Corrected Total	330	5132.997			

However, ANOVA can only indicate the presence or absence of statistically significant differences in the means of the F-Measure, hence the Fisher test is used to quantify the difference between the means. The statistical difference between the three methods is shown in the "Difference" column in Table 15. The difference in performance between the BU patterns and KNN model is high, since the former is a better semi-supervised model than the latter supervised model. The difference between the BU and TD patterns is low, given that both are semi-supervised approach models. These statistically significant different values prove that the three models constructed are significantly different, as is also indicated in the last column of Table 15.

Table 15 Method / Fisher (LSD) / An Analysis of the Differences between the Categories with a Confidence Interval of 95%.

Contrast	Difference	Standardized difference	Critical value	Pr > Diff	Significant
BU vs KNN	6.316	15.842	1.967	< 0.0001	Yes
BU vs TD	1.923	4.858	1.967	< 0.0001	Yes
TD vs KNN	4.392	11.017	1.967	< 0.0001	Yes
LSD-value:			0.782		

The same can be inferred from Table 16, which shows the group formation based on the F-Measure of the matching tasks. Here, groups are formed based on the similarity among the F-Measure values, irrespective of the method to which they belong. Further, only three groups are formed, since only three methods are compared. After the groups are formed, each method *M* is assigned a set of groups which have the F-Measure values of M. Since the F-Measure values across each method are dissimilar and values within the methods are similar, each group comprised the F-Measure values belonging only to a single method. Therefore, as shown in Table 16, the three methods belong to three different groups, indicative of the statistically significant difference between the three methods. Thus, from the set of statistical results above, it can be concluded that the BU patterns are statistically different and perform the best. Further, the same statistical testing is carried out using the precision and recall values of the matching tasks and similar results are obtained. Also, the test is repeated for the Conference and Benchmark2 tracks and similar conclusions are derived. Hence, similar to the previous section, statistical testing also proves that the BU patterns are more effective in comparison with the TD patterns and the supervised ML approach, KNN. Therefore, for the purpose of further experiments, only the BU patterns are evaluated.

Table 16 Group Formation.

Category	LS means	Groups		
BU	86.232	A		
TD	84.309		B	
KNN	79.917			C

5.3. Existing Static and Automatic vs Proposed Automatic Matching

In this section, to gauge the effectiveness of the selected BU patterns (Table 11), a series of experiments was conducted using the three tracks listed in Table 1.

The first experiment evaluates automatic matching against static matching in terms of the precision, recall and F-Measure of the matching task. Similar to the existing static matching systems such as the Falcon [18], COMA++ [8], and Rimom [21], the configured matcher cm3 (Lexical + Structural) is used to statistically match the given tasks, irrespective of their characteristics. In automatic matching, for each matching task, the best configured matchers for matching are automatically selected by the BU patterns. The chief objective of the proposed approach is to increase

the effectiveness of the matching results by a precise and automatic selection of the configured matcher based on the characteristics of the matching tasks. The effectiveness is improved by enhanced correctness (precision) or completeness (recall) or both, depending on the chosen configured matcher. Table 17 shows the improved performance in terms of the average precision, recall and F-Measure for each of the considered tracks. This result proves that the objective of the proposed approach is successfully accomplished.

Table 17 Comparison of Precision, Recall and F-Measure Values on Automatic vs Static Matching.

	Benchmark			**Conference**			**Benchmark2**		
	P*	**R***	**FM***	**P**	**R**	**FM**	**P**	**R**	**FM**
Automatic (A)	0.92	0.81	0.86	0.42	0.66	0.51	0.87	0.79	0.83
Static (S)	0.77	0.69	0.73	0.55	0.44	0.49	0.84	0.78	0.81

The next evaluation is based on the matching time required by automatic and static matching. As shown in Figure 9, on average for all the three tracks, the matching time required by the proposed automatically selected matcher is 95.66 seconds. This is higher, compared to the static matcher's average matching time of 81.66 seconds. Because the Benchmark track had many tasks with lexical, structural and instance characteristics, the proposed approach chose the cm1 matcher. This led to extra execution time in comparison with the static matcher comprising only the lexical and structural matchers. Meanwhile, the Conference and Benchmark2 tracks have no instances and therefore the proposed automatic matching chooses between cm2 and cm3. Hence, for these tracks, the execution time of automatic matching marginally outperforms that of static matching. From the results above, it can be concluded that, depending on the characteristics of the matching tasks, the matching time required by the proposed automatic matching may increase or decrease in comparison with that of the static.

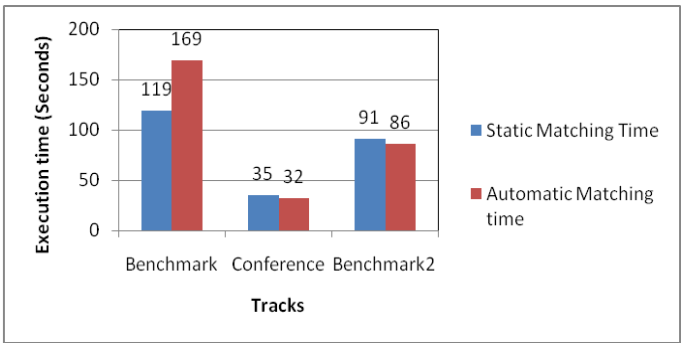

Figure 9 Execution Time of Static and Automatic Matching

It is also vital to note that the automatic matching time also includes the negligible pre-processing time required to quantify the characteristics of the ontologies. For example, the pre-processing time required for the Anatomy track [40], with a single matching task of the OAEI 2011, is 45 seconds. The automatic matching chose the cm3 matcher, which is the same as the static matcher, and the matching time required is 35 minutes. From this it can be inferred that the pre-processing time is negligible when compared to the matching time.

Finally, the proposed BU patterns are compared with the existing work in automatic matcher selection. However, to the best of our knowledge, as discussed in section 2, the proposed approach and that of Cruz et al. [7] are the only two systems which fully automate the process of matcher selection

without constraints like with the Rimom system [21]. Moreover, Rimom automates both the matcher selection and match results combination, whereas this work aims only at matcher selection. Hence the proposed approach can only be compared with the work of Cruz et al. [7]. As mentioned earlier, the results of Cruz et al., presented below, are obtained from their paper [7].

Now, the percentage of matching tasks in which the proposed automatic matching surpasses the static matching is computed (Figure 10). Similarly, the percentage of improved matching tasks of the KNN model of Cruz et al. [7], in comparison with the static matching, is also shown in Figure 10. On average, for all the three tracks, the improved percentage of the task for the proposed BU pattern-based automatic matching is 31.67% and KNN-based matching 27.3%. The proposed automatic matching is able to achieve a 16% increase in terms of an improved percentage of tasks in comparison with the existing KNN model [7]. Next, to gauge the percentage of the increased F-Measure resulting from these improved matching tasks, an evaluation is conducted in terms of the Average Gain/Task [7]. It is defined as the average increase in the F-Measure for the improved tasks. Figure 11 shows that, on average for all the three tracks, the Average Gain/Task for the proposed automatic matching is 14.6% and 6.37% for KNN-based matching. The proposed automatic matching is able to achieve a 129% increase in the Average Gain/Task compared to the existing KNN model. The improved effectiveness is due to the precise selection of the configured matcher by the BU patterns, based on the characteristics of the matching task.

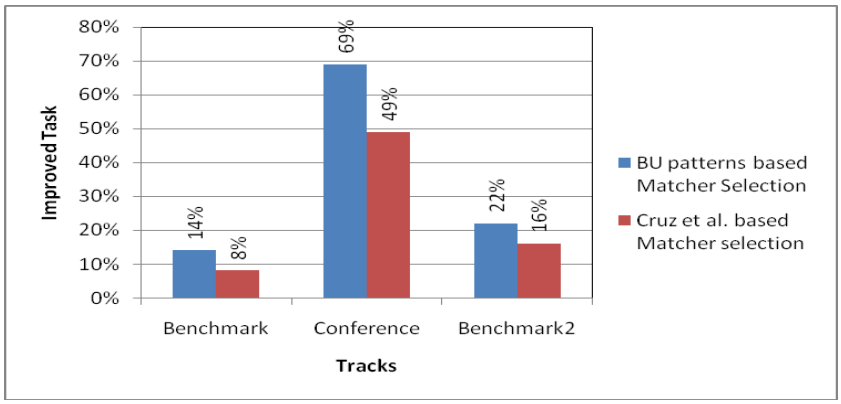

Figure 10 Percentage of Improved Number of Tasks
of the Proposed vs Cruz et al.-based Matcher Selection

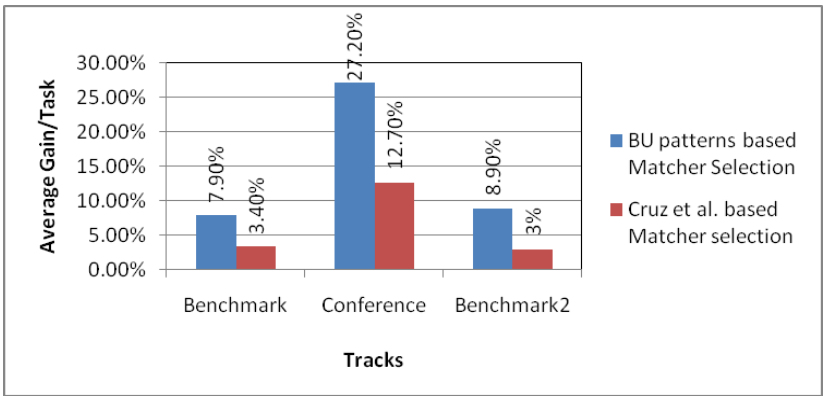

Figure 11 Average Gain/Task of the Proposed vs Cruz et al.-based Matcher Selection

6 Conclusions and Future Work

This paper aims to automatically select configured matchers based on the characteristics of the matching task using the proposed Bottom-up and Top-down Bootstrapping-based pattern learning. In addition, two new structural quality metrics, the Class External Structural Richness (CESR) and the Class Internal Structural Richness (CISR), are proposed to accurately gauge the structural characteristics of the matching task, facilitating the learned patterns for better matcher selection. From the patterns learned, it is inferred that the BUB approach constructs more effective and generic patterns with only 5 metrics, compared to the TDB approach patterns with 7 metrics. Also, the F-Measures of the matching tasks obtained using the BU patterns are better than the TD patterns. Hence, it can be concluded that the BU patterns are more efficient and effective than the TD patterns. Similarly, the effectiveness of the proposed metrics is evaluated by automatically selecting the matchers with and without the proposed metrics. Based on the results, it is evident that the proposed metrics do aid in automatic matcher selection. Further, the proposed pattern-based automatic matcher selection has been compared with the existing supervised ML approaches and the KNN model of Cruz et al. Experimental results in terms of the F-Measure of the constructed patterns, ANOVA statistical testing, percentage of the improved number of tasks and Average Gain/Task have demonstrated the improved effectiveness of the proposed BU pattern-based automatic matcher selection. From these results, it can be concluded that semi-supervised approaches are a better choice to model sparsely-labeled data than supervised ML approaches. Also, as stated by Occam's razor principle, the model should be built with the smallest possible (and sufficient) features.

In future, the limited number of configured matchers considered can be increased for handling automatic matcher selection in a range of matching scenarios. This is possible by using a larger set of element matchers to be classified in a fine-grained manner with such details as names, labels, comments, attributes, relations, and contexts, rather than merely lexical, structural and instance. In addition, the matching process can take into account a background knowledge of the domain or ontology so as to enable better matching.

Acknowledgements

This work of the first author was financially supported by Anna University, Chennai, India under the Anna Centenary Research Fellowship. The work of the third author was partially supported by a 2014 School of Business Administration Spring/Summer Fellowship at Oakland University, Rochester, Michigan, USA.

References

1. Abney, S. (2004). Understanding the yarowsky algorithm. *Computational Linguistics*, *30*(3), 365-395.
2. Agrawal, R., Ailamaki, A., Bernstein, P. A., Brewer, E. A., Carey, M. J., Chaudhuri, S., et al. (2008). The Claremont report on database research. *ACM Sigmod Record*, *37*(3), 9-19.
3. Algergawy, A., Nayak, R., Siegmund, N., Köppen, V., & Saake, G. (2010).Combining schema and level-based matching for web service discovery. In B. Benatallah, F. Casati, G. Kappel, & G.Rossi(Eds.), *Web Engineering*(pp. 114-128).Springer Berlin Heidelberg.
4. Bellahsene, Z., Bonifati, A., & Rahm, E. (2011). *Schema matching and mapping* (Vol. 20). Heidelberg (DE): Springer.

5. Burton-Jones, A., Storey, V. C., Sugumaran, V., & Ahluwalia, P. (2005). A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering*, *55*(1), 84-102.
6. Choi, N., Song, I. Y., & Han, H. (2006). A survey on ontology mapping. *ACM Sigmod Record*, *35*(3), 34-41.
7. Cruz, I. F., Fabiani, A., Caimi, F., Stroe, C., & Palmonari, M. (2012). Automatic configuration selection using ontology matching task profiling. In E. Simperl, P. Cimiano, A. Polleres, O. Corcho,& V. Presutti(Eds.), *The Semantic Web: Research and Applications* (pp. 179-194). Springer Berlin Heidelberg.
8. Do, H. H., & Rahm, E. (2007). Matching large schemas: Approaches and evaluation. *Information Systems*, *32*(6), 857-885.
9. Doan, A., Domingos, P., & Halevy, A. Y. (2001, May). Reconciling schemas of disparate data sources: A machine-learning approach. In *ACM Sigmod Record* (Vol. 30, No. 2, pp. 509-520). ACM.
10. Duan, S., Fokoue, A., & Srinivas, K. (2010). One size does not fit all: Customizing ontology alignment using user feedback. In P.F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J.Z. Pan, I, Horrocks,& B. Glimm(Eds.), *The Semantic Web–ISWC 2010* (pp. 177-192). Springer Berlin Heidelberg.
11. Duque-Ramos, A., Fernández-Breis, J. T., Stevens, R., & Aussenac-Gilles, N. (2011). OQuaRE: A SQuaRE-based approach for evaluating the quality of ontologies. *Journal of Research and Practice in Information Technology*, *43*(2), 159.
12. Ehrig, M., Staab, S., & Sure, Y. (2005). Bootstrapping ontology alignment methods with APFEL. In Y. Gil, E. Motta, V.R. Benjamins, & M. Musen(Eds.), *The Semantic Web–ISWC 2005* (pp. 186-200). Springer Berlin Heidelberg.
13. Euzenat, J., & Shvaiko, P. (2007). *Ontology matching* (Vol. 333). Heidelberg: Springer.
14. Gal, A. (2011). Uncertain schema matching. *Synthesis Lectures on Data Management*, *3*(1), 1-97.
15. Gal, A., & Sagi, T. (2010). Tuning the ensemble selection process of schema matchers. *Information Systems*, *35*(8), 845-859.
16. Gal, A., & Shvaiko, P. (2009). Advances in ontology matching. In E.J. Chang, & K. Sycara(Eds.) *Advances in web semantics* (pp. 176-198). Springer Berlin Heidelberg.
17. Hariri, B. B., Sayyadi, H., Abolhassani, H., & Esmaili, K. S. (2006, August). Combining Ontology Alignment Metrics Using the Data Mining Techniques. In *ECAI International Workshop on Context and Ontologies* (pp. 65-67).
18. Hu, W., Qu, Y., & Cheng, G. (2008). Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, *67*(1), 140-160.
19. Huza, M., Harzallah, M., & Trichet, F. (2007). OntoMas: a tutoring system dedicated to ontology matching. In R. Jardim-Gonçalves, J. Müller, K. Mertins,& M. Zelm (Eds.), *Enterprise Interoperability II* (pp. 377-388). Springer London.
20. Lee, Y., Sayyadian, M., Doan, A., & Rosenthal, A. S. (2007). eTuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal—The International Journal on Very Large Data Bases*, *16*(1), 97-122.
21. Li, J., Tang, J., Li, Y., & Luo, Q. (2009). Rimom: A dynamic multi strategy ontology alignment framework. In *IEEE Transactions on Knowledge and Data Engineering*, *21*(8), 1218-1232.

22. Mao, M., Peng, Y., & Spring, M. (2008). A Harmony based Adaptive Ontology Mapping Approach. In *SWWS* (pp. 336-342).
23. Marie, A., & Gal, A. (2008). Boosting schema matchers. In Tari, Z. (Ed.), *On the Move to Meaningful Internet Systems: OTM 2008* (pp. 283-300). Springer Berlin Heidelberg.
24. Mochol, M., & Jentzsch, A. (2008). Towards a rule-based matcher selection. In *Knowledge Engineering: Practice and Patterns* (pp. 109-119). Springer Berlin Heidelberg.
25. Mochol, M., Jentzsch, A., & Euzenat, J. (2006). Applying an analytic method for matching approach selection. In *Proc. 1st ISWC 2006 international workshop on ontology matching (OM)* (pp. 37-48).
26. Ngo, D. H., & Bellahsene, Z. (2012). Evaluating the Interaction between the different Matchers (or Strategies) in Ontology Matching Task. In *International Semantic Web Conference-ISWC 2012* (p. 12).
27. Otero-Cerdeira, L., Rodríguez-Martínez, F. J., & Gómez-Rodríguez, A. (2015). Ontology matching: A literature review. *Expert Systems with Applications*, *42*(2), 949-971.
28. Peukert, E., Eberius, J., & Rahm, E. (2011, April). AMC-A framework for modelling and comparing matching systems as matching processes. In *IEEE 27th International Conference on Data Engineering (ICDE)*(pp. 1304-1307). IEEE.
29. Peukert, E., Eberius, J., & Rahm, E. (2012, April). A self-configuring schema matching system. In *2012 IEEE 28th International Conference on Data Engineering* (pp. 306-317). IEEE.
30. Rahm, E. (2011). Towards large-scale schema and ontology matching. In Z. Bellahsene, A. Bonifati, & E.Rahm (Eds.), *Schema matching and mapping* (pp. 3-27). Springer Berlin Heidelberg.
31. Sagi, T., & Gal, A. (2013). Schema matching prediction with applications to data source discovery and dynamic ensembling. *The VLDB Journal*, *22*(5), 689-710.
32. Saruladha, K., Aghila, G., & Sathiya, B. (2011). A comparative analysis of ontology and schema matching systems. *International Journal of Computer Applications*, *34*(8), 14-21.
33. Shi, F., Li, J., Tang, J., Xie, G., & Li, H. (2009). Actively learning ontology matching via user interaction. *In A.* Bernstein, D.R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, & K. Thirunarayan(Eds.), *The Semantic Web- ISWC 2009* (pp. 585-600). Springer Berlin Heidelberg.
34. Shvaiko, P., & Euzenat, J. (2013). Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering, 25*(1), 158-176.
35. Spiliopoulos, V., & Vouros, G. (2012). Synthesizing ontology alignment methods using the max-sum algorithm. *IEEE Transactions on Knowledge and Data Engineering, 24*(5), 940-951.
36. Steyskal, S., & Polleres, A. (2013). Mix'n'Match: iteratively combining ontology matchers in an anytime fashion. In *OM* (pp. 223-224).
37. Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-Meza, B. (2005). OntoQA: Metric-based ontology quality analysis. In: *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, vol. 9, (pp. 45–53). IEEE.
38. Tu, K., & Yu, Y. (2005, April). CMC: Combining multiple schema-matching strategies based on credibility prediction. In *International Conference on Database Systems for Advanced Applications* (pp. 888-893). Springer Berlin Heidelberg.

39. Yang, P., Wang, P., Ji, L., Chen, X., Huang, K., & Yu, B. (2014). Ontology Matching Tuning Based on Particle Swarm Optimization: Preliminary Results. In D. Zhao, J. Du, H. Wang, P. Wang, D. Ji, & J.Z. Pan (Eds.) *The Semantic Web and Web Science* (pp. 146-155). Springer Berlin Heidelberg.
40. Anatomy Track (2011). Available: http://oaei.ontologymatching.org/2011/anatomy/index.html
41. Benchmark track (2011). Available: http://oaei.ontologymatching.org/2011/benchmarks/
42. Benchmark track2 (2011). Available: http://oaei.ontologymatching.org/2011/benchmarks2/
43. COMA 3.0 ontology matching tool (Nov 2012). Available: http://sourceforge.net/projects/coma-ce/
44. Conference track (2011). Available: http://oaei.ontologymatching.org/2011/conference/index.html
45. One Way ANOVA - University of Wisconsin - Stevens Point. [Online]. Available: http://www.uwsp.edu/psych/stat/12/anova-1w.ht
46. Ontology Alignment Evaluation Initiative (OAEI). (2011). Available: http://oaei.ontologymatching.org/2011/