Instance Coreference Resolution in Multi-ontology Linked Data Resources

Aynaz Taheri and Mehrnoush Shamsfard

Computer Engineering Department, Shahid Beheshti University, Tehran, Iran
ay.taheri@mail.sbu.ac.ir, m-shams@sbu.ac.ir

Abstract. Web of linked data is one of the main principles for realization of semantic web ideals. In recent years, different data providers have produced many data sources in the Linking Open Data (LOD) cloud upon different schemas. Isolated published linked data sources are not themselves so beneficial for intelligent applications and agents in the context of semantic web. It is not possible to take advantage of the linked data potential capacity without integrating various data sources. The challenge of integration is not limited to instances; rather, schema heterogeneity affects discovering instances with the same identity. In this paper we propose a novel approach, SBUEI, for instance co-reference resolution between various linked data sources even with heterogeneous schemas. For this purpose, SBUEI considers the entity co-reference resolution problem in both schema and instance levels. The process of matching is applied in both levels consecutively to let the system discover identical instances. SBUEI also applies a new approach for consolidation of linked data in instance level. After finding identical instances, SBUEI searches locally around them in order to find more instances that are equal. Experiments show that SBUEI obtains promising results with high precision and recall.

Keywords: Linked Data, Coreference Resolution, Ontology, Schema, Instance, Matching.

1 Introduction

Linked data is a new trend in the semantic web context. Nowadays increasing the amount of linked data in Linking Open Data project is not the only challenge of publishing linked data; rather, matching and linking the linked data resources are also equally important and can improve the effective consuming of linked data resources. Linked data integration is one of the main challenges that become more important considering development of linked data. Without these links, we confront with isolated islands of datasets. The fourth rule of publishing linked data in [2] explains the necessity of linking URIs to each other. Therefore, extension of datasets without interlinking them is against the Linked Data principles.

In the web of linked data with so large scale, there are obviously many different schemas in the various linked data sources. Considering that there is no compulsion for data providers in utilizing specific schema, we confront with the problem of schema heterogeneity in data sources. This issue is considerable in instance coreference resolution. Paying attention to schemas in linked data consolidation has many advantages. When we are going to discover instances with unique identity in two data sources, it is a complicated process to compare all the instances of two data sources in order to find equivalents. Processing all of the instances has harmful effects on execution time and needs more computing power. However, if we know about schema matching of two data sources, it is not necessary to look up all the instances. Rather, it is enough to search only instances of two matched concepts of schemas, so performance would become better. In addition, ignoring the schema may cause precision decrease in instance matching. In many cases, the internal structure and properties of instances do not have enough information to distinguish distinct instances and this increases the possibility of wrong recognition of co-referent instances that are apparently similar in some properties in spite of their different identities.

Although ontology/schema matching can be beneficial for instance matching, it could be detrimental if it is done inefficiently. In [18] effects of ontological mismatches on data integration (in instance level) are described. They divide all types of mismatches into two groups: conceptual mismatches and explication mismatches. They represent that these kinds of mismatches such as conceptual mismatches could be harmful for instance matching and could decrease the amount of precision by wrong matching of concepts in the schema level. They do ontology matching at the first step and instance matching at the second step. Because of this sequential process, the errors of the first step can propagate into the next step.

In this paper, we propose a solution, SBUEI, to deal with the problem of instance matching and schema matching in linked data consolidation. SBUEI proposes an interleaving of instance and schema matching steps to find coreferences or unique identities in two linked data sources. SBUEI, unlike systems such as [13, 21, 27] -which uses just instance matching- or systems such as [15, 24] -which use just schema matching- exploits both levels of instance and schema matching. The main difference between SBUEI and other systems like [19], which exploit both levels, is that SBUEI exploits an interleaving of them while [19] exploits them sequentially one after the other (starts instance matching after completing schema matching). SBUEI utilizes schema matching results in instance matching and use the instance matching results in order to direct matching in schema level. SBUEI also has a new approach for instance matching.

This paper is structured as follows: section 2 discusses some related work. Section3 explains the instance coreference resolution algorithm at the first phase, and section 4 describes the schema matching algorithm at the second phase. Section 5 demonstrates the experimental results of evaluating SBUEI. Finally, section 6 concludes this paper.

2 Related Work

We divide related works in the area of entity coreference resolution in the context of semantic web into four groups:

Some related works deal with specific domains. Raimond et al. in [25] proposed a method for interlinking two linked data music-related datasets that have similar ontologies and their method was applicable on that specific ontology. In [10], authors described how they interlink a linked data source about movies with other data sources in LOD by applying some exact and approximate string similarity measures. In [32] a method for linking WordNet VUA (WordNet 3.0 in RDF) to DBpedia is proposed. The methodology of this project is customized for only these two datasets. Finding identical instances of foaf:person at social graph is explained in [26] by computing graph similarity.

Some pieces of related works follow the challenge of linked data consolidation so that their methods are not constrained to special domains but their methods are based on the assumption that schemas of different data sets are equal and their approach only concentrate on consolidation of data in instance level. In [11], capturing similarity between instances is based on applying inverse functional properties in OWL language. In [21], authors used a similarity measure for computing similarity of instance matching between two datasets with the same ontology. LN2R [27] is a knowledge based reference reconciliation system and combines a logical and a numerical method. LN2R requires manual alignment at first and then turns to the reconciliation of instances.

Another group of approaches, which focus on consolidating linked data, claims that their approach does not depend on schema information and can identify same instances in heterogeneous data sources without considering their ontologies. Hogan et al. [12] proposed a method for consolidation of instances in RDF data sources that is based on some statistical analysis and suggests some formula in order to find "quasi" properties. ObjectCoref [13] is a self-training system based on a semi supervised learning algorithm and tries to learn discriminative property-value pair based on a statistical measurement. Song et al. [31] proposed an unsupervised learning algorithm in order to find some discriminable properties as candidate selection key. SERIMI [1] is an unsupervised method and has a selection phase in order to find some specific properties and a disambiguating phase. Zhishi.links [20] is a distributed instance matching system. It does not follow any special techniques for schema heterogeneity. It uses an indexing process on the names of instances and uses string similarity to filter match candidates.

Some other approaches of instance matching in linked data sources take advantage of schema in data sets. HMatch(τ) [4] is an instance matcher and use HMatch 2.0 for TBox matching and then tries to capture the power of properties at instance identification [9]. RiMOM [33] , ASMOV [16] and AgreementMaker [5] are three ontology matching systems that recently equipped with instance matchers and participated in instance matching track of OAEI 2010. CODI [22] is also a system for ontology and instance matching and is based on markov logic. Seddiqui et al. [30] proposed an instance matching algorithm and Anchor-Flood algorithm [29] for ontology matching at first and then begin instance matching. Nikolov et al. [18] discussed the effects of ontology mismatches on instance resolution and proposed Knofuss architecture in [19]. Linked Data Integration Framework (LDIF) [28] has two main components: Silk Link Discovery Framework [14] and R2R Framework [3] for identity resolution and vocabulary normalization respectively.

From the four above-mentioned groups, approaches in the last two groups are not dependent on any schemas or domains. It seems that the forth group have more advantages in comparison with third group, because the approaches in the third group have deprived themselves of utilizing useful information in the schema level. As we said in the section 1, paying attention to schema has beneficial effects on linked data consolidation. Our proposed approach, SBUEI, belongs to the forth group. What distinguish SBUEI from the aforementioned approaches in the forth group are its consecutive movements between schema and instance level and the matching algorithm in the instance level. SBUEI exploits the instance level information in order to find accurate proposal about schema matching and then applies schema matching in order to do instance resolution with high precision and recall.

3 Instance Coreference Resolution In Linked Data Sources

The instance coreference resolution algorithm has two phases that are executed iteratively. The first phase needs to receive an anchor as input. Anchor is defined as a pair of similar concepts across ontologies [23, 29]. As the first and second phases are executed in a cycle, for the first round, the user should provide this input, but in the next times the input of the first phase (the anchors) is provided by the output of the second phase.

The first phase starts by getting anchor: $X \equiv \langle C_1, C_2 \rangle$. C_1 and C_2 are two similar concepts from ontologies $O_1$and O_2 respectively. It comprises three steps explained in the following in details.

3.1 First Step: Create Linked Instances Cloud

In the first step, we introduce a new construction that is called Linked Instances Cloud (*LIC*), as the basis of our matching algorithm.

For anchors $\langle C_1, C_2 \rangle$, we must create *LICs*. For each instance of concepts C_1 and C_2 we make one *LIC*. C_1 has some instances. For example the URI of one of the C_1 instances is called '*i*' .We explain how to create *LIC* for instance '*i*'. For creating this *LIC*, SBUEI extracts all of the triples whose subjects are instance '*i*' and adds them to the *LIC*. Then, in the triples that belong to *LIC*, we find neighbors of instance '*i*'. If instance '*j*' is one of the neighbors of instance '*i*', SBUEI repeats the same process for instance '*j*'. Triples whose subjects are instance '*j*', are added to *LIC* and the process is repeated for neighbors of the instance '*j*'. This process is actually like depth first search among neighbors of instance '*i*'. To avoid falling in a loop and to eliminate the size of search space, the maximum depth of search is experimentally set to 5. The *LIC* that is created for instance '*i*' is called LIC_i. Starting point for this *LIC* is instance '*i*'. The process of creating *LICs* is done for all of the instances of the two concepts C_1 and C_2.

Sometimes the identities of instances are not recognizable without considering the instances that are linked to them, and neighbors often present important information about intended instances. In some cases in our experiments, we observed that even discriminative property-value pairs for an instance may be displayed by its neighbors. Fig. 1 shows an illustration about an instance that its neighbors describe its identity. This example is taken from IIMB dataset in OAEI 2010. Fig. 1 shows $LIC_{item21177}$. '*Item21177*' is the starting point of this *LIC* and is an Actor, Director and a character-creator. Each instance in the neighborhood of '*Item21177*' describes some information about it. For example '*Item74483*' explains the city that '*Item21177*' was born in and '*Item27054*' explains the name of the '*Item21177*'.

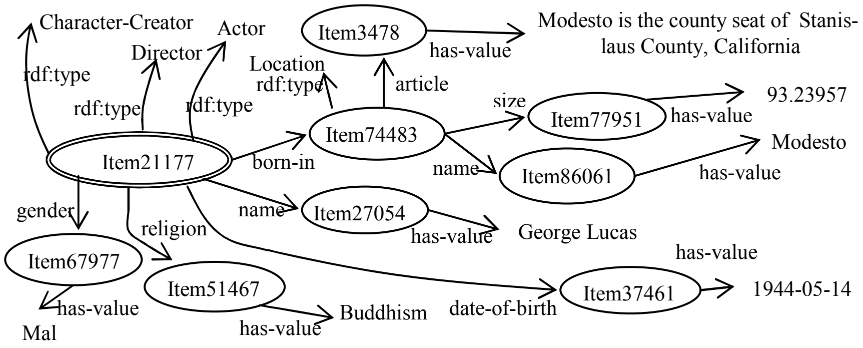

Fig. 1. An Illustration of LIC

Creating *LICs* not only helps us in discovering identities of instances, but also it helps us to find locally more similar instances. This issue is explained in section 3.3.

3.2 Second Step: Compute Similarity between LICs

In the previous step, SBUEI created the LICs of two concepts C_1 and C_2. In this step, we must compare them. Each LIC from concept C_1 should be compared with all LICs of concept C_2 in order to find similar LICs. Starting points of two similar LICs would be equal. Therefore, the triples of two LICs should be compared so that two LICs can be compared. In this process, only triples whose objects are data type values (and not instances) would participate in the comparison. Properties values are very important in comparison. This does not mean that properties (predicates in triples) are effectless in similarity computation. However, finding equal properties in two different ontologies is not usually easy, especially in very heterogeneous ontologies. Therefore, values of properties have more importance for SBUEI. However, if we could find similar properties, the process of similarity computation between properties values would be easier and more effective in increasing of similarity value.

SBUEI computes similarity of LICs in three separate parts as the followings.

Part 1: Normalization of Properties Values

SBUEI applies some normalization techniques on properties values for improving the result of string comparison. These techniques consist of: case normalization, blank normalization, punctuation elimination and removing stop words.

Part 2: Compare Triples in Two LICs

In this part, all of the triples whose objects are values of data type will be compared in two *LICs*. However, before comparing values of properties, properties itself will be compared. Similarity computation for properties are done regarding to three different aspects: range similarity, lexical similarity and properties hierarchies. Equation (1) describes the function that computes properties similarities.

HierSim function computes hierarchal similarity between two properties p_1 and p_2. Properties in the hierarchy are examined by two functions *LexicalSim* and *RanSim*. *RanSim* function compares ranges of two properties p_1 and p_2. *LexicalSim* function computes similarity between the labels of two properties p_1 and p_2. Section 4.1 explains these functions in detail.

$$PropertySim(p_1,p_2) = \frac{(RanSim(p_1,p_2) + LexicalSim(p_1,p_2) + HierSim(p_1,p_2,H^P))}{3} \tag{1}$$

If SBUEI finds similar properties in two *LICs*, then objects of triples that have equal properties will be compared. Other triples in two *LICs* that we could not find equal properties for them, will be compared considering only their values of properties. Our experiments show that often in most cases SBUEI does not find similar properties because of lexical, structural and semantical heterogeneity in schemas. Thus, we focus on values of properties when similar properties are not found.

We use Edit Distance method for computing similarity of properties values. Considering v_1 and v_2 as values of two properties p_1 and p_2, SBUEI computes their similarity according to (2).

$$EditDisSim(v_1,v_2) = \begin{cases} EditDistance(v_1,v_2), if & EditDistance(v_1,v_2) \geq \delta \\ 0, & if \; EditDistance(v_1,v_2) < \delta \end{cases} \tag{2}$$

Similarity values of triples objects -obtained by *EditDisSim* function- are added together for obtaining similarity value of two *LICs*. In (2) we applied a threshold equal to 0.6 for edit distance method. This threshold was found by making a benchmark and execution of edit distance algorithm based on the benchmark. Equation (2) removes similarity values that are less than threshold. This prevents accumulating small similarity values in the sum of similarity values of triples objects.

Some properties are comments about instances. For similarity computation of these kinds of properties, we do not use (2). SBUEI in (3) applies a specific method for finding comment similarity. It is based on the number of common tokens. Again, we made a benchmark, found a threshold (δ) equal to 0.7.

$$ComSim(x, x') = \begin{cases} CommentSim(x, x'), if & CommentSim(x, x') \geq \delta \\ 0, & if \; CommentSim(x, x') < \delta \end{cases}$$
$$CommentSim(x, x') = \frac{2 \times (|x \cap x'|)}{|x| + |x'|} \tag{3}$$

After calculating similarity of properties values, SBUEI computes similarity of two *LICs*. Similarity of two *LICs* is dependent on similarity of their properties values. Triples in two *LICs* have specific importance depend on the depth of their subjects (instances that triples belong to) in the *LICs*. We noted in section 3.1 that depth of instances are estimated towards the starting point of *LIC*. When depth of instances in

LIC increases, their effectiveness in similarity computation of *LICs* decreases. The following triples belong to $LIC_{item21177}$ in Fig. 1.

1 (‘Item67977’, has-value, Male)
2 (‘Item37461’, has-value, 1944-05-14)
3 (‘Item77951’, has-value, 93.23957)
4 (‘Item3478’, has-value, Modesto is the county seat of Stanislaus County, California)

The above triples describe some information about the starting point of $LIC_{item21177}$. Two first triples explain that ‘*item21177*’ has male gender and date of his birth is 1994-05-14. Instances in the subjects of these two triples have depth equal to two. Two second triples explain that ‘*item21177*’ has born in a city that its size is 93.23957 and also is the county seat of Stanislaus County, California. Instances in the subjects of these two triples have depth equal to three. As you can see, the first two triples have more important role for determining the identity of ‘*item21177*’ than the second two triples. Gender of a person and date of his birth is more important than some comments about the city that he lives in. Nevertheless, this does not mean that existence of instances with greater depth are not beneficial in the *LICs*; rather, they are less important in identity recognition of the starting point of the *LIC* than those with less depth. In addition, we utilize such instances in step three for finding more instances that are similar.

In this regard, similarities of properties values are added with an particular coefficient. SBUEI uses a weighted sum for computing similarity of *LICs*. The coefficients (4) in this sum have inverse relations to the depth of the subject of triples in *LIC* .

$$S_n = \frac{1}{n} \tag{4}$$
$$S = \{S_n : n \in N\} \quad n\ is\ the\ depth\ of\ the\ subject\ s$$

SBUEI normalizes the sum of similarities of properties values in two *LICs* into a range of 0 and 1 by dividing the result to the sum of the numbers of triples in two *LICs*.

In (5) LIC_1 and LIC_2 have n and m triples, respectively. We find for each object of LIC_1 the most similar object in LIC_2 via *LexicalPropValSim* function. *LexicalPropValSim* works based on (2) and (3). Each of the two objects with most similarity; have a coefficient equal to inverse of the depth of the triple that belongs to LIC_1. We do the same process for LIC_2 and then add obtained values of similarity LIC_1 and LIC_2. Normalization is done by the sum of the triples numbers in LIC_1 and LIC_2 (In (5) this value is m+n), but each triple has a coefficient according to what mentioned in (4).

$$LIC_1 = \{t_1, t_2, \dots, t_n\}$$
$$LIC_2 = \{t'_1, t'_2, \dots, t'_m\}$$

$$LICSimilarity(LIC_1, LIC_2) = \frac{\begin{pmatrix} \sum_{i=1}^{n} \frac{1}{Depth(t_i)} max_{1\le j \le m}\left(LexicalPropValSim\left(Obj(t_i), Obj(t'_j)\right)\right) + \\ \sum_{j=1}^{m} \frac{1}{Depth(t'_j)} max_{1\le i \le n}\left(LexicalPropValSim\left(Obj(t'_j), Obj(t_i)\right)\right) \end{pmatrix}}{\left(\sum_{i=1}^{n} \frac{1}{Depth(t_i)} + \sum_{j=1}^{m} \frac{1}{Depth(t'_j)}\right)} \tag{5}$$

Part 3: Choose Two Similar LICs

In the previous part, we computed the similarity of two *LICs*. Now we confront with a complex challenge. The main challenge is determining the value of the threshold for deciding whether two *LICs* are equal or not. This threshold is the final approver about equality of two *LICs* based on their similarity value. Our experiments show that the value of threshold is completely variable depending on the data sources and their characteristics. The amounts of threshold have considerable differences in a range between zero and one. For example results of several tests on two data sources, indicated that the best value for similarity threshold of *LICs* are 0.7. While on two other data sources this value was 0.2. Wide range of obtained values for threshold led us to have a dynamic selection for threshold of *LICs* similarity.

Dynamic selection of threshold means that after creating *LICs* and computing their similarities, we choose the value of threshold depending to the concepts that *LICs* belong to. LIC_1 and LIC_2 belong to two concepts C_1 and C_2 respectively. We calculate a specific threshold for two concepts C_1 and C_2. Afterward we apply acquired threshold for comparing *LICs* from these two concepts. For all *LICs* that their starting points belong to two particular concepts from two ontologies, SBUEI use a specific threshold.

The purpose of threshold value variation and the necessity of dynamicity in threshold selection can be justified because of the heterogeneity of ontologies viewpoints regarding to concepts. Two ontologies may have information considering different aspects of concepts and therefore their instances even with the same identities have diverse information with little overlap. In such cases, ontologies are semantically different. For example, one ontology has some properties such as name, director, characters for describing a film and another ontology has name, actors, character-creator properties for describing a film. Hence, equal instances of these two ontologies have low similarities despite the fact that they have the same identities.

SBUEI uses a heuristic for solving this problem. After finding the most similar *LICs* of two concepts C_1 and C_2, SBUEI calculates the similarity average of the most similar *LICs* of two concepts. Instances of concept C_1 have almost the same properties for describing individuals and instances of concept C_2 are also the same way. Hence, the similarity amounts of instances of these two concepts are approximately predictable and are the same.

SimLIC in (6) is the set of most similar *LICs* of two concepts C_1 and C_2. Each member of the set is a triple: the first element is the intended *LIC* from concept C_1, the second element is the most similar *LIC* from concept C_2 and the third element is the value of their similarity. *LICSimThreshold* computes the threshold of two concepts C_1 and C_2.

$$SimLIC = \{(L_1, L'_1, S_1), (L_2, L'_2, S_2), \dots, (L_n, L'_n, S_n)\}$$

$$LICSimThreshold = \frac{\sum_{i=1}^{n} S_i}{n} \qquad (6)$$

3.3 Third Step: Determine New Equal Instances in Two Similar LICs

In this step, we continue the process of matching on those LICs of the previous step that led to discovering equal instances or in the other words, those LICs that have equal starting points. The strategy in this step is searching locally around the identical instances in order to find new equal instances. In [29] an algorithm for ontology matching is created, and their algorithm is based on the idea that if two concepts of two ontologies are similar, then there is a high possibility that their neighbors are similar too. We use this idea but in instance level. This means that if two instances are identical, then there is possibility that their neighbors are similar too.

Suppose that 'i' and 'j' are two instances of two concepts C_1 and C_2 and they are detected identical in the previous step. Their LICs are called LIC_i and LIC_j. In this step we describe how SBUEI finds more identical instances in LIC_i and LIC_j. This step is composed of three parts as following.

Part 1: Create Sub-Linked Instances Cloud
We define a new construction called 'Sub-LIC'. SBUEI makes one LIC for each instance in LIC_i and LIC_j. Each LIC has some Sub-LICs in itself as many as the number of its instances.

Part 2: Discover Similar Sub-LICs
For discovering similar instances in LIC_i and LIC_j, we compare their *Sub-LICs*. Similarity of *Sub-LICs* is computed same as *LICs* with the same threshold value. *Sub-LICs* that have similarities more than the threshold are equal and their starting points are considered as identical instances.

Finding identical instances of two concepts initially costs a lot because of considering all neighbors of an instance; later we can find locally more identical instances by paying low computational cost.

Part 3: Compute Concept Similarities in Schema Level
After finding identical instances in *Sub-LICs*, now it is time to arrange for moving to schema level. In this part instance matcher gives feedback to schema matcher. Concepts whose instances are the starting points of equal *Sub-LICs* are candidates to be similar. Suppose that $Sub\_LIC_{i_1}$ and $Sub\_LIC_{j_1}$ are two *Sub-LICs* of LIC_i and LIC_j respectively and are detected similar in the previous part. Starting points of $Sub\_LIC_{i_1}$ and $Sub\_LIC_{j_1}$ belong to two concepts C_3 and C_4 from ontologies O_1 and O_2 respectively. We can conclude that C_3 and C_4 are probably similar because they have identical instances.

SBUEI repeats three above parts for all LICs of concepts C_1 and C_2 that are detected similar in step 2. We define the following measure for similarity estimation between two concepts C_3 and C_4: 'ratio of equal Sub-LICs of two concepts C_1 and C_2 that their starting points belong to concepts C_3 and C_4 to the total numbers of LICs of two concepts C_1 and C_2'. Instance matcher utilizes this measure for helping schema matcher in order to find equal concepts.

4 Schema Matching in Linked Data Sources

The second phase is done by a schema matcher. It receives feedback from the first phase, which contains some similarities between concepts from the viewpoint of instance matcher. At this time, schema matcher begins the process of matching in schema level by applying some ontology matching algorithms. SBUEI compares all of these similarity values that are proposed by instance matcher or obtained by schema matcher, and choose a pair of concepts that have the most similarity. SBUEI repeats these two phases consecutively until all of similar concepts that are given feedback to schema level or are detected by schema matcher, to be selected and schema matcher could not find any similar concepts.

4.1 Compute Similarity between Concepts

When SBUEI wants to do ontology matching, it considers to the concepts that are proposed as equal concepts in the previous iterations and the process of ontology matching starts in the neighborhood of these concepts.

We applied the definition of concept neighborhood in [29]. In this definition, neighbors of a concept include its children, grandchildren, parents, siblings, grandparents and uncles. Schema matcher utilizes two similarity measures for ontology matching: label-based techniques and structure-based techniques. Therefore, we have two kinds of matchers: lexical matcher and structural matcher.

Compute Lexical Similarity

Lexical similarity of two concepts or two properties are described by *LexicalSim()* function in (8). We compute the similarity of two concepts or two properties with using a string-based method and utilizing a lexical ontology.

Lexical features of a concept are such as id, label and comments. For computing comments similarities between concepts or properties, we use a token-based method, the same as equation (3). For other features of concepts or properties, SBUEI uses Princeton WordNet [8], a lexical ontology, for finding similarities.

Let l_1 and l_2 be two labels or ids for two concepts C_i and C_j respectively, and let *Synset* (l_1) denotes a synset in WordNet that l_1 belongs to it and *Senses(s)* returns all of senses of a synset *s*, then

$$WNSenses(l_1) = \{x | x \in Senses(Synset(l_1))\} \tag{7}$$

$$LexicalSim(l_1, l_2) = \begin{cases} 1, & if \;\; l_1 = l_2 \\ 1, & if \;\; l_1 \in WNSenses(l_2) \;\; or \;\; l_2 \in WNSenses(l_1) \\ WNStructuralSim(l_1, l_2), & if \;\; l_1 \notin WNSenses(l_2) \;\; AND \\ & \quad l_2 \notin WNSenses(l_1) \\ EditDistance(l_1, l_2), & if \;\; l_1 \notin WordNet \;\; or \;\; l_2 \notin WordNet \end{cases} \tag{8}$$

l_1 and l_2 have the most lexical Similarity value, if they are equal or l_1 is one of the senses of the synset that l_2 belongs to or vice versa. *WNStructuralSim*(l_1, l_2) in (8) computes the similarity of two synsets that l_1 and l_2 are one of their senses respectively. The similarity of those synsets is computed regarding to their positions in the hierarchy of WordNet. We used the Wu-Palmer measure presented in [34]. If l_1 or l_2 is not in WordNet we use Edit Distance method for computing their similarity.

Compute Structural Similarity

In addition to the names, ids, comments and all other lexical features of concepts, their structure in ontologies are also important for calculating concepts similarities. In [6] structure based techniques are divided into two groups based on the internal structure and relational structure. SBUEI utilizes internal and relational structures for computing similarities between concepts.

Internal Similarity.

Internal similarity compares properties of two concepts from two different ontologies. Let p_i and p_j be two properties belong to two concepts C_i and C_j respectively, *InternalStructuralSim()* function in (9) is calculated using the following formula:

$$InternalStructuarlSim(C_i, C_j) = \frac{\left(2 \times \left(\sum_{p_i \in Property_{(C_i)}} Max_{p_j \in Property(C_j)} \begin{pmatrix} RanSim(p_i, p_j) + \\ LexicalSim(p_i, p_j) + \\ CardSim(p_i, p_j) \end{pmatrix}\right)\right)}{\left(|Property(C_i)| + |Property(C_j)|\right)} \tag{9}$$

We defined *LexicalSim* previously. *CardSim* compares cardinality of properties. It will be 1.0 if Maximum and Minimum values of properties are equal and 0 otherwise. *RanSim* computes similarity value between data types of two properties range. Utilizing hierarchy of XML schema data types is proposed in [6] for computing data type similarity. SBUEI also uses XML schema data type hierarchy but applies Wu-Palmer method on the hierarchy in order to find the similarity of two data types regarding to their positions in the hierarchy.

Relational Similarity

Relational similarity in SBUEI computes similarity of hierarchies that two concepts from two different ontologies belong to. SBUEI considers taxonomic structure and pays more attention to rdfs:subClassOf for relational similarity computation. Parents and children of a concept are compared with the parents and children of another concept. For comparing parents and children in (10), we use *LexicalSim* and *InternalStructuralSim* functions.

$$\begin{aligned} RelationalStructuralSim(C_i, C_j) &= \left(\frac{\sum_{n_i \in Parent(C_i)} Max_{n_j \in Parent(C_j)} \begin{pmatrix} LexicalSim(n_i, n_j) + \\ InternalStructuarlSim(n_i, n_j) \end{pmatrix}}{\left(|Parent(C_i)| + |Parent(C_j)|\right)}\right) \\ &+ \left(\frac{\sum_{n_i \in Children(C_i)} Max_{n_j \in Children(C_j)} \begin{pmatrix} LexicalSim(n_i, n_j) + \\ InternalStructuarlSim(n_i, n_j) \end{pmatrix}}{\left(|Children(C_i)| + |Children(C_j)|\right)}\right) \end{aligned} \tag{10}$$

Similarity Aggregation

SBUEI uses a weighted linear aggregation process to compose the results of different similarity values .The weights have been determined experimentally. We obtained values as: $w_{lexical}$=0.4, $w_{internal}$=0.3 and $w_{relational}$=0.3

5 Evaluation

The experimental results are downloadable at our website[1] and some statistics of SBUEI operations in comparison with other systems are prepared[2].

For evaluating SBUEI, we use the datasets in OAEI [7] , a benchmarking initiative in the area of semantic web. We report the experimental results of our proposed approach on four different datasets: PR in OAEI 2010, IIMB in OAEI 2010, IIMB in OAEI 2011 and TAP-SWETO datasets in OAEI 2009. In the last experiment, we found an interesting use case that takes advantage from our approach.

5.1 Person-Restaurants Benchmark

Person-Restaurants benchmark is one of the tasks in instance matching track in OAEI 2010 campaign. This benchmark is composed of three datasets: Person1, Person2 and Restaurants. Each of them has two different owl ontologies and two sets of instances. Person1 and Person2 contain instances about some peoples, and Restaurants has instances about some restaurants. Sizes of these datasets are small. Reference alignments are provided in OAEI 2010 and all of participants in this task must compare their results with the reference alignments. Five systems have participated in this task and we compare our results with the results of other systems.

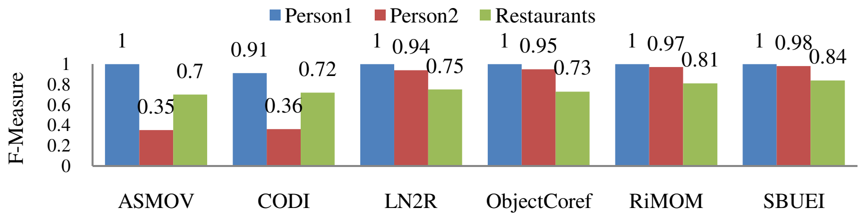

Fig. 2. Result of OAEI'10 Person-Restaurant Benchmark

F-measure values of all systems on the Person-Restaurant benchmark are shown in Fig. 2. Performance of all systems (including SBUEI) are quite good on Person1 dataset. Collected statistics of all the participants show that SBUEI has the best values of F-measure and precision on Person2 and Restaurants datasets. According to obtained values, we conclude that SBUEI performed well in matching at both schema and instance levels of these three datasets.

[1] `http://nlp.sbu.ac.ir/sbuei/result.rar`

[2] `http://nlp.sbu.ac.ir/sbuei/statistics.html`

5.2 IIMB'10 Track

The second part of our experiments includes IIMB task of OAEI 2010. IIMB composed of 80 test cases. Each test case has OWL ontology and a set of instances. Information of test cases in IIMB track is extracted from Freebase dataset. IIMB divided test cases in four groups. Test cases from 1 to 20 have data value transformations, 21 to 40 have structural transformations, test cases from 40 to 60 have data semantic transformations and 61 to 80 have combination of these three transformations. All of these 80 test cases must be matched against a source test case. We choose IIMB 2010 test cases for the evaluation because this track of OAEI has a good number of participants and its test cases have all kinds of transformations and we could compare all aspects of our system against the other systems.

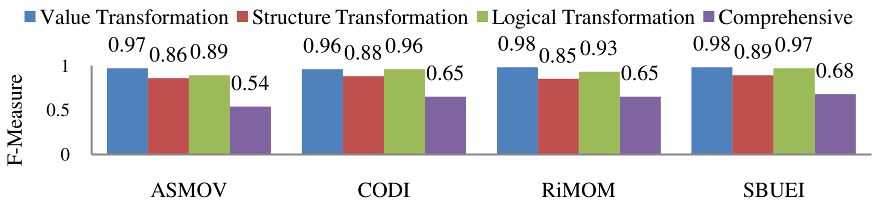

Fig. 3. Results of OAEI'10 IIMB Track

The results of four systems on F-measure are depicted in Fig. 3. We could observe that all four systems have good values of F-measure on datasets with data value transformation. SBUEI and ASMOV have the best values for F-measure. All participating systems have weaker results in test cases with structural value transformation.

SBUEI has better operations than others in these datasets. This means that SBUEI is more stable in modifications such as removing, adding and hierarchal changing of properties. Systems have better results in test cases with semantic value transformation against structure value transformation. SBUEI has the best F-measure in such cases. Four systems do not have desirable results in datasets with combination all kinds of transformations.

5.3 IIMB'11 Track

In this part of our experiments, we are going to show the results of SBUEI on IIMB track of OAEI 2011. This track has also 80 test cases with four kinds of transformations just like the last year IIMB track. The purpose of selecting this track as one of our experiments is that the size of IIMB 2011 has increased greatly compared to last year and is more than 1.5 GB. Increased amount of the dataset size lets us evaluate scalability of our approach. Unfortunately, there has been just one participant in this track, CODI, with which we will compare our results. This shows the scalability difficulties in systems performance at large scale datasets. We observe in Fig. 4 that the recall values of SBUEI in four kinds of transformations are better than CODI but this is not always true for precision value. The operations of SBUEI is clearly better than CODI in datasets with structure transformation considering three aspects of precision, recall and F-measure.

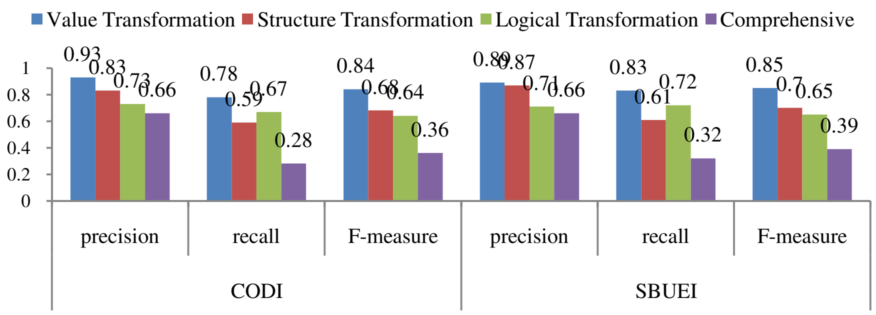

Fig. 4. Results of OAEI'11 IIMB Track

5.4 TAP-SWETO Datasets

OAEI 2009 created a benchmark in instance matching track that was created according to different ontologies. This benchmark includes SWETO and TAP datasets. In [18], authors used these two datasets and explained the effects of ontology mismatches in instance matching on these two datasets. We explain again one of the problems that Nikolov pointed and describe how SBUEI overcame some difficulties in instance matching of these two datasets. Unfortunately, this track was cancelled due to the lack of participants and there are no reference alignments or any other systems to be compared with our results.

Therefore, we mention only how SBUEI performed the process of matching. There are some instances about computer researchers in two datasets. TAP ontology has a concept that is called '*ComputerScientist*' and on the other hand SEWTO ontology has a concept that is called '*ComputerScienceResearcher*'. These two concepts have also some structural similarities in addition to lexical similarities, because their positions in the hierarchy of ontologies are similar. The first one has Sub-class realtion with '*Scientist'* concept and the second has Sub-class relation with '*Reasercher'* concept. More over both '*Scientist'* and '*Reasercher'* concepts are children of '*Person'* concept in their ontologies. Therefore, '*ComputerScientist*' and '*ComputerScience Researcher*' are good candidates to be matched by structural and lexical matchers. If we confirm this matching and then start instance resolution process between their instances, we confront with a few numbers of matched instances. This is because of conceptualization mismatches in schema. '*ComputerScientist*' in TAP includes only famous computer scientist and '*ComputerScienceResearcher*' has more general sense and includes who have a paper in computer science research areas.

SBUEI removed this problem and had a good instance resolution process. SBUEI began the process of matching after receiving a pair of concepts as anchor: ('*ResearchPaper*' , '*Scientific_Publication*'). After acquiring these two concepts, SBUEI is confident about similarity of instances between the concepts. SBUEI created LICs around the instances of two concepts '*ResearchPaper*' and '*Scientific_Publication'* and computed similarity of LICs and then selected equal LICs with similarities more than threshold. SBUEI found more similar instances in equal LICs of these two concepts. In the neighborhood of LICs starting points, SBUEI found some equal instances that are authors of papers. These newly founded instances belonged to concepts such as: '*CMUPerson*' and '*W3CPerson*' from TAP and '*ComputerScienceResearcher*' from SEWTO. Thus, SBUEI discovered similarity between '*CMUPerson*' and '*W3CPerson'* concepts of TAP and '*ComputerScienceResearcher*' concept of SWETO. Then SBUEI started the process of instance resolution and found more matched instances compared to the previous time. In particular, TAP has divided computer researchers considering to their place of their works. These kinds of mismatches in schema level could not be discovered easily by ontology matchers. Ontology matchers and instance matchers can improve their performance cooperatively.

6 Conclusion

In this paper, we proposed a new approach, SBUEI, for linked data consolidation. This approach is applicable in with heterogeneous schemas. SBUEI pays attention to matching in both schema and instance level. Instance resolution process starts in SBUEI after getting two equal concepts as input by instances matcher. Instance matcher creates LICs around the instances of two equal concepts and then compares these clouds. After discovering instances with the same identity in *LICs*, instance matcher utilizes them and sends some similarity feedback to the schema matcher. Schema matcher receives feedback, applies some ontology matching algorithms, determines two equivalent concepts in ontologies and gives them to instance matcher as input. This process continues consecutively. Our experiments showed that our approach achieved high precision and recall and outperforms other systems.

Our future target includes utilizing some methods that are proposed at the third group of section 2 in our approach. This means that we are going to use some learning algorithms to find discriminable properties in the *LICs*. This will help us to find similar *LICs* efficiently. Considering that SBUEI is a recently created approach, does not have appropriate user interface. Therefore, it is important to make a powerful user interface for SBUEI.

References

1. Araujo, S., Hidders, J., Schwabe, D., de Vries, A.P.: SERIMI - Resource Description Similarity, RDF Instance Matching and Interlinking. CoRR, abs/1107.1104 (2011)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data-The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
3. Bizer, C., Schultz, A.: The R2R Framework: publishing and discovering mapping on the web. In: 1st International Workshop on Consuming Linked Data, China (2010)
4. Castano, S., Ferrara, A., Montanelli, S., Lorusso, D.: Instance matching for ontology population. In: 16th Symposium on Advanced Database Systems, Italy (2008)
5. Cruz, I.F., Store, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F.M., Palmonari, M.: Using AgreementMaker to align ontologies for OAEI 2011. In: 6th International Workshop on Ontology Matching, Germany (2011)
6. Euzenat, J., Shvaiko, P.: Ontology Matching, 1st edn. Springer, Heidelberg (2007)

7. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: Six Years of Experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
8. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
9. Ferrara, A., Lorusso, D., Montanelli, S.: Automatic identity recognition in the semantic web. In: 1st ESWC Workshop on Identity and Reference on the Semantic Web, Spain (2008)
10. Hassanzadeh, O., Consense, M.: Linked movie data base. In: 2nd Link Data on the Web, Spain (2009)
11. Hogan, A., Harth, A., Decker, S.: Performing object consolidation on the semantic web data graph. In: 1st Identity, Identifiers, Identification Workshop, Canada ((2007)
12. Hogan, A., Polleres, A., Umbrich, J., Zimmermann, A.: Some entities are more equal than others: statistical methods to consolidate linked data. In: 4th International Workshop on New Forms of Reasoning for the Semantic Web, Greece (2010)
13. Hu, W., Chen, J., Qu, Y.: A Self-training Approach for Resolving Object Coreference Semantic Web. In: 20th International World Wide Web Conference, India (2011)
14. Isele, R., Jentzsch, A., Bizer, C.: Silk server- adding missing links while consuming linked data. In: 1st International Workshop on Consuming Linked Data, China (2010)
15. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)
16. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: ASMOV: Results for OAEI 2010. In: 5th International Workshop on Ontology Matching, China (2010)
17. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. J. Web Sem. 7(3), 235–251 (2009)
18. Nikolov, A., Uren, V., Motta, E.: Toward data fusion in a multi-ontology environment. In: 2nd Linked Data on the Web Workshop, Spain (2009)
19. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 332–346. Springer, Heidelberg (2009)
20. Niu, X., Rong, S., Zhang, Y., Wang, H.: Zhishi.links results for OAEI 2011. In: 6th International Workshop on Ontology Matching, Germany (2011)
21. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging terminological structure for object reconciliation. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 334–348. Springer, Heidelberg (2010)
22. Noessner, J., Niepert, M.: CODI : Combinatorial Optimization for Data Integration – Results for OAEI 2010. In: 5th International Workshop on Ontology Matching, China (2010)
23. Noy, N., Musen, M.: Anchor-PROMPT: using non-local context for semantic matching. In: Ontologies and Information Sharing Workshop, USA (2001)
24. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and building ontologies of linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 598–614. Springer, Heidelberg (2010)
25. Raimond, Y., Sutton, C., Sandler, M.: Automatic Interlinking of music datasets on the semantic web. In: 1st Link Data on the Web, China (2008)

26. Rowe, M.: Interlinking Distributed Social Graphs. In: 2nd Linked Data on the Web Workshop, Spain (2009)
27. Sais, F., Niraula, N., Pernelle, N., Rousset, M.: LN2R a knowledge based reference reconciliation system: OAEI 2010 results. In: 5th International Workshop on Ontology Matching, China (2010)
28. Schultz, A., Matteini, A., Isele, R., Bizer, C., Becker, C.: LDIF-Linked data integration framework. In: 2nd International Workshop on Consuming Linked Data, Germany (2011)
29. Seddiqui, M.H., Aono, M.: An Efficient and Scalable Algorithm for Segmented Alignment of Ontologies of Arbitrary Size. J. Web Sem. 7(4), 344–356
30. Seddiqui, M.H., Aono, M.: Ontology Instance Matching by Considering Semantic Link Cloud. In: 9th WSEAS International Conference on Applications of Computer Engineering, Russia (2009)
31. Song, D., Heflin, J.: Automatically generating data linkages using a domain-independent candidate selection approach. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 649–664. Springer, Heidelberg (2011)
32. Taheri, A., Shamsfard, M.: Linking WordNet to DBpedia. In: Proceedings of the 6th Global WordNet Conference, Japan (2012)
33. Wang, Z., Zhang, X., Hou, L., Zhao, Y., Li, J., Qi, Y., Tang, J.: RiMOM Results for OAEI 2010. In: 5th International Workshop on Ontology Matching, China (2010)
34. Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection. In: 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces (1994)