LCS: A Linguistic Combination System for Ontology Matching

Qiu Ji, Weiru Liu, Guilin Qi, David A. Bell

School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast
Belfast, BT7 1NN, UK
{Q.Ji,W.Liu,G.Qi,DA.Bell}@qub.ac.uk

Abstract. Ontology matching is an essential operation in many application domains, such as the Semantic Web, ontology merging or integration. So far, quite a few ontology matching approaches or matchers have been proposed. It has been observed that combining the results of multiple matchers is a promising technique to get better results than just using one matcher at a time. Many aggregation operators, such as *Max*, *Min*, *Average* and *Weighted*, have been developed. The limitations of these operators are studied. To overcome the limitations and provide a semantic interpretation for each aggregation operator, in this paper, we propose a linguistic combination system (LCS), where a linguistic aggregation operator (LAO), based on the ordered weighted averaging (OWA) operator, is used for the aggregation. A weight here is not associated with a specific matcher but a particular ordered position. A large number of LAOs can be developed for different uses, and the existing aggregation operators *Max*, *Min* and *Average* are the special cases in LAOs. For each LAO, there is a corresponding semantic interpretation. The experiments show the strength of our system.

1 Introduction

The Semantic Web [1] has gained a lot progress in recent years. In this field, ontology is a key technique for the interoperability of heterogeneous systems. Currently, a large amount of ontologies have been developed in various research domains or even in the same domain. But for different ontologies, the same entity may be named differently, or defined in different ways. Even the same name may represent different entities. Ontology matching, which takes two different ontologies as input and outputs the correspondences between semantically equivalent entities (e.g., classes, properties, instances), becomes a critical solution to deal with these problems. It has been applied in many application domains, such as the Semantic Web, ontology merging or integration.

Now, quite a few ontology matching approaches or matchers [2–7] have been proposed. Good surveys of the matchers are provided in [8, 9]. These matchers exploit various kinds of information in ontologies, such as entity names, entity descriptions, name paths, taxonomic structures. It has been accepted that combining the results of multiple matchers is a promising technique to get better results than just using one matcher at a time [2–4, 10].

Some matcher combination systems, such as LSD [3], COMA [2], CMC [10] have been developed. In general, the combination methods or aggregation operators in these systems include *Max*, *Min*, *Average* and *Weighted* (such as *Weighted Average*). It is clear that *Max* and *Min* [2] are too extreme to perform well. While *Average* [2] is inefficient to cope with the ontologies with very different structures. A *Weighted* based method [3, 2, 10] needs to compute the weights of different matchers. One way to get the weights is to assign them manually, and the other is by machine learning technique. Obviously, it is difficult for a person to estimate the weights by experience and rich data sets are needed to train the algorithm to obtain reliable weights for machine learning methods.

To overcome these limitations, in this paper, we propose a linguistic combination system (LCS), which combines the results of multiple matchers based on the ordered weighted average (OWA) operator and linguistic quantifiers [11]. The OWA operator generally includes three steps [12]:

- Reorder the input arguments in descending order.
- Determine the weights associated with the OWA operator.
- Utilize the OWA weights to aggregate these reordered arguments.

A weight here is associated with a particular ordered position not a specific matcher. In the OWA operator, determining weights is a key step. We adopt the way to obtain weights by a linguistic quantifier [11, 13]. So we call our system the linguistic combination system (LCS). A linguistic aggregation operator (LAO) will be used for the aggregation in LCS. LAO is an OWA operator where the associated weights are obtained by the linguistic quantifiers [11]. Specifically, it is composed of the following four steps:

- Reorder the similarity values to be combined in descending order. These values are obtained by the base matchers on the current task.
- Choose or define a linguistic quantifier.
- Obtain the OWA weights by the linguistic quantifier.
- Apply the OWA weights to aggregate these similarity values.

It is interesting that existing aggregation operators like *Max*, *Min* and *Average* are special cases of LAOs. Besides, there is a semantic interpretation for each LAO to facilitate users to choose an appropriate LAO. So LAO provides a good way to supply a gap for existing aggregation operators without considering the weights to matchers.

This paper is organized as follows. Some related work is introduced in the next section. In Section 3, we give more details on the background knowledge on ontology matching and the OWA operator. The linguistic aggregation operators (LAOs) are described in Section 4. Section 5 defines the matching process which uses LAO to aggregate the results of multiple matchers. Experiment results are analyzed in Section 6. Finally, we conclude the paper and give some future work in Section 7.

2 Related Work

Ontology matchers have been developed by many researchers for all kinds of information provided in ontologies. Due to the space limitation for the paper, we only introduce some of them here. NOM [4] proposed seventeen rules by experts, which can be seen as seventeen matchers. These rules contain different aspects of an ontology, such as super concepts, sub concepts, super properties and sub properties. Cupid [5] integrates linguistic and structural matching. Importantly, it makes use of a wide range of techniques to discover mappings between schema elements. The techniques based on element names, data types, constraints and schema structures are included. In Lite [7], a universal measure for ontology matching is proposed. It separates the entities in an ontology into eight categories like classes, objects, datatypes. For an entity in a category, all the features about its definition are involved. In these papers, most of the matchers can be selected as base matchers to be combined in a combination system [10].

In existing matcher combination systems, some typical systems are mentioned as follows. LSD [3] is a data-integration system which semi-automatically finds semantic mappings by employing and extending machine-learning techniques. It aggregates multiple similarities obtained by the individual matchers by means of weighted average, where the similarities and weights are acquired by machine learning. COMA [2] exploits *Max*, *Min*, *Average* and *Weighted* strategies for combination. The *Weighted* strategy needs a relative weight for each matcher to show its relative importance. For each category in Lite [7], a set of all relationships in which the category participates is defined. And the relative weights are assigned to each relationship. Only the entities in the same category can be matched. CMC [10] combines multiple schema-matching strategies based on credibility prediction. It needs to predict the accuracy of each matcher on the current matching task first by a manual rule or a machine learning method. Accordingly, different credit for the pair is assigned. Therefore from each base matcher, two matrices including the similarity matrix and the credibility matrix are provided. It aggregates all the similarity matrices into a single one by weighted average, where the weights are determined by the credibility matrices. In NOM [4], it is mentioned that not all matchers have to be used for each aggregation, especially as some matchers have a high correlation. Both manual and automatic approaches to learn how to combine the methods are provided. The weights they use are determined manually or by a machine learning method.

To sum up, when it is not necessary or difficult to get weights for matchers, the aggregation operator which we can choose for aggregation includes *Max*, *Min* and *Average*. However, since each base matcher performs differently in different conditions, these operators may be not enough to show the various performance for complex situations [14].

In this paper, we propose a linguistic combination system (LCS), which includes rich linguistic aggregation operators (LAOs). And according to the semantic interpretation of LAOs we provide, it is much more convenient for users to choose an appropriate LAO.

3 Background

3.1 Ontology matching

Typically, an ontology is to define a vocabulary in a domain of interest, and a specification of the meaning of entities used in the vocabulary. In this paper, the entities in an ontology are separated into three categories: classes, properties and instances. We only match the entities in the same category and represent ontologies in OWL[1] or RDF(S)[2].

The similarity for ontologies is defined as a similarity function: $sim(e_{1i}, e_{2j}) \in [0,1]$, where e_{1i}, e_{2j} are two entities in the same category from a source ontology onto1 and a target ontology onto2 separately. Especially, $sim(e_{1i}, e_{2j}) = 0$ indicates e_{1i} and e_{2j} are different, and $sim(e_{1i}, e_{2j}) = 1$ shows they are the same. If the similarity $sim(e_{1i}, e_{2j})$ exceeds a threshold $th_{final} \in [0,1]$, we call e_{2j} the matching candidate of e_{1i}. Furthermore, if there is more than one matching candidate in onto2 for e_{1i}, the one with the highest similarity is selected as its matched entity.

3.2 The ordered weighted averaging (OWA) operator

The ordered weighted averaging (OWA) operator is introduced by [11] to aggregate information. It has been used in a wide range of application areas, such as neural networks, fuzzy logic controllers, vision systems, expert systems and multi-criteria decision aids [15].

Given a set of arguments $V_1 = (a_1, a_2, ..., a_n), a_i \in [0,1], 1 \leq i \leq n$, reorder the elements of the set in descending order and mark the ordered set as $V_2 = (b_1, b_2, ..., b_n)$, where b_j is the jth highest value in V_1. An OWA operator is a mapping F from I^n to I, $I = [0,1]$:

$$\begin{aligned} F(a_1, a_2, ..., a_n) &= \sum_{i=1}^{n} w_i b_i \\ &= w_1 b_1 + w_2 b_2 + ... + w_n b_n, \end{aligned}$$

where each weight $w_i \in [0,1]$ and $\sum_{i=1}^{n} w_i = 1$.

Note that the weight w_i is not associated with a particular argument a_i, but with a particular ordered position i of the arguments. That is w_i is the weight associated with the ith largest argument whichever component it is [11].

4 The linguistic aggregation operator (LAO)

From the previous section, it is obvious that a critical technique in the OWA operator is to determine the OWA weights $w_i, 1 \leq i \leq n$. So far, quite a few approaches have been proposed, for example, O'Hagan [16] introduced a procedure to generate the OWA weights by a predefined degree of orness and maximizing the entropy of the OWA weights. An interesting way to obtain the weights is developed by Yager using linguistic quantifiers [11, 13].

[1] http://www.w3.org/TR/2004/REC-owl-guide-20040210/
[2] http://www.w3.org/TR/rdf-schema/

We adopt the linguistic quantifiers to determine the OWA weights. We use such kind of OWA operators as linguistic aggregation operators (LAOs). The following gives more details on how to aggregate the results of multiple matchers for an entity pair to be compared by LAO.

Assume there is n matchers of concern, $\{m_1, m_2, ..., m_n\}$, in an ontology matching problem. Let (x, y) be an entity pair, where x is an entity from a source ontology onto1, y from a target ontology onto2. For each matcher m_i, $m_i(x, y) \in [0, 1]$ indicates the similarity of x and y, i.e., the degree to which this matcher is satisfied by (x, y). The final similarity between x and y, $sim(x, y)$, can be computed by the results of the n matchers. That is,

$$\begin{aligned} sim(x,y) &= F(m_1(x,y), m_2(x,y), ..., m_n(x,y)) \\ &= \textstyle\sum_{i=1}^{n} w_i b_i \\ &= w_1 b_1 + w_2 b_2 + ... + w_n b_n, \end{aligned}$$

where F is the same function as that in the previous section, b_i is the ith largest value in $\{m_1(x, y), m_2(x, y), ..., m_n(x, y)\}$. According to the linguistic approach [11, 13], the weight w_i is defined by

$$w_i = Q(\tfrac{i}{n}) - Q(\tfrac{i-1}{n}),\ i = 1, 2, ..., n, \qquad (1)$$

where Q is a nondecreasing proportional fuzzy linguistic quantifier and is defined as the following:

$$Q(r) = \begin{cases} 0, & \text{if } r < a; \\ (r-a)/(b-a), & \text{if } a \le r \le b,\ \ a, b, r \in [0, 1]; \\ 1, & \text{if } r > b, \end{cases} \qquad (2)$$

where a and b are the predefined thresholds to determine a proportional or relative quantifier. $Q(r)$ indicates the degree to which r portion of objects satisfies the concept denoted by Q [17].

There are many proportional fuzzy quantifiers, such as *For all*, *There exists*, *Identity*, *Most*, *At least half*(*Alh*), *As many as possible*(*Amap*). Table 1 gives more details on some examples of LAOs.

Quantifier	$Q(r)$	w_i	LAO
There exists	$Q(r) = 0$, if $r = 0$ $Q(r) = 1$, if $r > 0$	$w_i = 1$, if $i = 1$ $w_i = 0$, if $i \neq 1$	Max
For all	$Q(r) = 0$, if $r < 1$ $Q(r) = 1$, if $r = 1$	$w_i = 0$, if $i < n$ $w_i = 1$, if $i = n$	Min
Identity	$Q(r) = r$, if $0 \le r \le 1$	$w_i = \frac{1}{n},\ i = 1, 2, ..., n$	Average
Most	$Q(r) = 0$, if $0 \le r < 0.3$ $Q(r) = 1$, if $0.8 < r \le 1$ $Q(r) = 2(r - 0.3)$, if $0.3 \le r \le 0.8$	$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n})$ $i = 1, 2, ..., n$	Most
Alh	$Q(r) = 2r$, if $0 \le r \le 0.5$ $Q(r) = 0$, if $0.5 < r \le 1$	$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n})$ $i = 1, 2, ..., n$	Alh
Amap	$Q(r) = 0$, if $0 \le r < 0.5$ $Q(r) = 2(r - 0.5)$, if $0.5 \le r \le 1$	$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n})$ $i = 1, 2, ..., n$	Amap

Table 1. Definitions of some LAOs

From this table, it is clear that the existing aggregation operators *Max*, *Min* and *Average* are special cases of LAOs. The following simple example illustrates the use of some LAOs.

Example: Assume the similarity values to be combined are $V_1 = (0.6, 1, 0.3, 0.5)$, where each value is obtained by a base matcher on the current matching task. After re-ordering V_1 in descending order, we get V_2=(1,0.6,0.5,0.3).

a) For *Most*, if we let a be 0.3 and b be 0.8 (see Table 1) for Equation (2), we obtain $Q(r) = 2(r - 0.3)$, if $0.3 \leq r \leq 0.8$; $Q(r) = 0$, if $0 \leq r < 0.3$; Q(r)=1, if $0.8 < r \leq 1$. So, the weights for the four positions in V_2 are computed as followings by Equation (1):

$w_1 = Q(\frac{1}{4}) - Q(0) = 0$
$w_2 = Q(\frac{2}{4}) - Q(\frac{1}{4}) = 2(\frac{2}{4} - 0.3) - 0 = 0.4$
$w_3 = Q(\frac{3}{4}) - Q(\frac{2}{4}) = 2(\frac{3}{4} - 0.3) - 2(\frac{2}{4} - 0.3) = 0.5$
$w_4 = Q(1) - Q(\frac{3}{4}) = 1 - 2(\frac{3}{4} - 0.3) = 0.1$

Hence,

$$\begin{aligned} F(0.6, 1, 0.3, 0.5) &= \textstyle\sum_{i=1}^{4} w_i b_i \\ &= (0)(1) + (0.4)(0.6) + (0.5)(0.5) + (0.1)(0.3) \\ &= 0.52 \end{aligned}$$

b) For *Alh*, we obtain $w_1 = 0.5$, $w_2 = 0.5$, $w_3 = 0$, $w_4 = 0$, by setting $a = 0$, $b = 0.5$, $n = 4$ for Equation (1) and (2). Here,

$$\begin{aligned} F(0.6, 1, 0.3, 0.5) &= \textstyle\sum_{i=1}^{4} w_i b_i \\ &= (0.5)(1) + (0.5)(0.6) + (0)(0.5) + (0)(0.3) \\ &= 0.8 \end{aligned}$$

5 An Overview of LCS

In this section, we first give an overview of ontology matching process in LCS. We then give more details on the matchers we will use for our evaluation and the semantic interpretation for LAOs.

5.1 Ontology matching process

Based on COMA [2], the matching process in LCS is illustrated in Figure 1, where LAO is used for the aggregation.

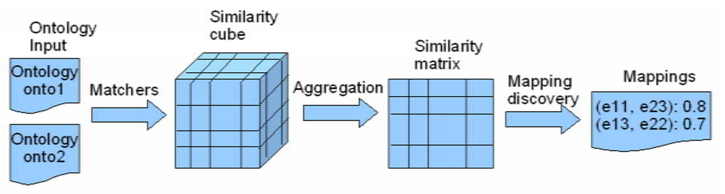

Figure 1. Matching process in LCS

The entities in an ontology are classes, properties or instances. For different entity category, different matchers may be used according to the features of each category. For example, only a property has domain and range, so that the matcher of *Domain and Range* could only be used for the properties category. Similar to the matcher of *Mother Concept*, it is only suitable for instances category.

According to COMA [2] and the survey of approaches to automatic schema matching [8], matchers can be divided into three categories: individual matchers, hybrid matchers and composite matchers. We assume in this paper that individual matchers do not rely on any initial similarity matrix provided by other matchers. In contrast, a hybrid matcher needs such a matrix and directly combines several matchers, which can be executed simultaneously or in a fixed order. A composite matcher combines its independently executed constituent matchers, including individual matchers and hybrid matchers.

A main step of LCS is to combine the results of multiple base matchers. A base matcher is the matcher to be combined, which can be individual matcher, hybrid matcher or composite matcher. After executing each matcher, a similarity matrix is obtained. Multiple similarity matrixes form a similarity cube, which can be combined by an aggregation operator to obtain a final similarity matrix. We use LAO to combine the results of multiple base matchers. Some linguistic quantifiers for LAOs have been described in Table 1, such as *At least half*, *Most*. Others can be defined by users by adjusting the two parameters in Equation (2).

The final step of matching process is to select match candidates from the final similarity matrix. We only focus on finding the best matching candidate or matched entity from the target ontology onto2 for each entity in the source ontology onto1 if possible, which is the task of mapping discovery.

5.2 Ontology matchers

In LCS, we choose some existing ontology matchers to evaluate our system. The details of these matchers are described as followings.

- **Name**: When comparing entity names, some preparation skills are adopted. For example, separate the name string into some tokens by the capital letters and some symbols such as ‘#’, ‘_’, ‘.’. Then delete some words like “the”, “has”, “an” from the token sets.
- **Name Path**: It considers the names of the path from the root to the elements being matched in the hierarchical graphs, which regards the classes or instances as nodes and relations as edges.
- **Taxonomy**: This matcher is a composite one. For classes, it consists of *super concept*(*Sup*), *sub concept*(*Sub*), *sibling concept*(*Sib*), and *properties*(*Prop*), which are the properties directly associated with the class to be matched. For properties, only *super properties*(*Sup*) and *sub properties*(*Sub*) are included here.
- **Domain and Range**: If the domain and range of two properties are equal, the properties are also similar.
- **Mother-concept**: Obviously, this matcher considers the type of an instance.

Among these matchers, *Name* is an individual matcher to provide the initial similarity matrix for other matchers. A good example for the composite matcher is *Taxonomy*. For *Name Path*, *Sup*, *Sub*, etc., they are hybrid matchers which rely on the initial matrix obtained by *Name*.

5.3 The semantic interpretation of LAOs

In Section 4, the definitions of some LAOs were given. We now give the semantic interpretation for them according to Yager's interpretation of the linguistic quantifiers [11, 13]. The interpretation makes it convenient for users to choose an appropriate LAO for the aggregation.

Based on the definition of LAO, the explanation for some LAOs is given as followings. For an entity pair (x, y),

- **Max**: $Max(x, y) = Max\{m_1(x, y), m_2(x, y), ..., m_n(x, y)\}$. *Max* means that (x, y) satisfies at least one of the matchers, i.e., satisfies m_1 or m_2 ... or m_n.
- **Min**: $Min(x, y) = Min\{m_1(x, y), m_2(x, y), ..., m_n(x, y)\}$. *Min* means that (x, y) satisfies all the matchers, that is to say, we are essentially requiring to satisfy m_1 and m_2 ... and m_n.
- **Avg**: *Avg* (*Average*) means identity. It regards all similarity values equally.
- **Most**: Obviously, *Most* means that most of the matchers is satisfied. Usually, this operator ignores some higher and lower similarity values, that is to give small weights on them, while paying more attention to the values in the middle of the input arguments after re-ordering.
- **Alh**: *Alh* (*At least half*) satisfies at least half matchers. Actually, it only considers the first half of similarity values after re-ordering them in descending order.
- **Amap**: *Amap* (*As many as possible*) satisfies as many as possible matchers and is opposite to *Alh*. The second half of values after reordering is considered. So after an aggregation operation, the result obtained by *Alh* is always higher than that by *Amap*.

6 Experiments

The base matchers we will use in LCS for experiments include *Name*(N for short), *Taxonomy*(T for short), *Name Path*(P for short), *Domain and Range*(D for short), *Mother concept*(M for short). Moreover, for two entities from two different ontologies, if they are in the classes category, N, T and P are used to compare the two entities. If they are in the properties category, N, T and D are used. If they are in the same instances category, we use N, P and M.

The experiments are made on some ontologies provided in the context of the I^3CON conference[3]. We give more details on these ontology pairs as followings. The labels with bold font are used to represent each ontology pair.

[3] http://www.atl.external.lmco.com/projects/ontology/i3con.html

- **Animals**: It includes two ontologies which are defined in a similar way and around 30 entities for each ontology. They have 24 real mappings.
- **Cs**: Cs represents two computer science(cs) departments at two Universities respectively. More than 100 entities are involved in the first ontology, while about 30 entities in the second one. The number of real mappings is 16.
- **Hotel**: Hotel describes the characteristics of hotel rooms. The two ontologies in Hotel are equivalent, but defined in different ways. In each ontology, around 20 entities are defined. About 17 real mappings are identified by humans.
- **Network**: networkA.owl and networkB.owl describe the nodes and connections in a local area network. networkA.owl focuses more on the nodes themselves, while networkB.owl is more encompassing of the connections. Each ontology has more than 30 entities. In total we have 30 real mappings.
- **Pets1**: Pets1 is composed of ontology people+petsA.owl and people+petsB.owl which is a modified version of people+petsA.owl. More than 100 entities are defined in each ontology and 93 real mappings are determined manually.
- **Pets2**: Identical to Pets1 above without instance data. 74 real mappings are created.
- **Russia**: The pair of russiaA.rdf and russiaB.rdf for Russia describes the locations, objects, and cultural elements of Russia. Each of them has more than 100 entities. The total number of theoretical mappings is 117.

To evaluate LCS, the common matching quality measures are exploited in the next section. The following three sections are to show the performance of the combination methods in LCS by using some public ontologies. It is noted that, in order to discover the mapping candidates from the aggregated similarity matrix, we tune the threshold th_{final} to get the best performance by experience according to the characteristics of the combination methods and ontologies to be matched.

6.1 The criterion of evaluation

The standard information retrieval measures [18] are adopted by us.

- **Precision**: $precision = \frac{|I|}{|P|}$. It reflects the share of the correct mappings among all mappings returned automatically by a matcher.
- **Recall**: $recall = \frac{|I|}{|R|}$ specifies the number of correct mappings versus the real mappings determined by manually.
- **F-Measure**: $f - Measure = \frac{2*precision*recall}{precision+recall}$, which represents the harmonic mean of Precision and Recall.
- **Overall**: $overall = recall * (2 - \frac{1}{precision})$, which is introduced in [6]. It is a combined measure and takes into account the manual effort needed for both removing false and adding missed matches.

Where, $|I|$ indicates the number of the correct mappings that are found by the automatic matchers. $|P|$ is the number of all mappings that are found automatically, which includes the false and correct mappings. $|R|$ shows the number of the manually determined real mappings.

6.2 Single matchers vs. combination

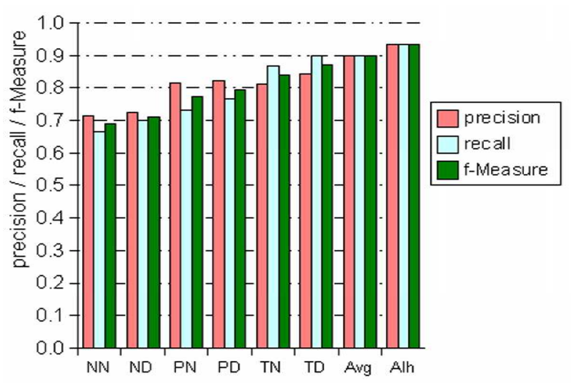

Figure 2. Single matchers vs. combination methods

Figure 2 shows the performance of some single matchers and combination methods on the ontology pair "network", which includes ontology networkA.owl and networkB.owl. Since there is no instance in networkA.owl, we only compare the classes and properties in two ontologies.

In Figure 2, each single matcher is marked as two capital letters, where the first one indicates a single matcher for classes category and the second one for properties category. For instance, "*ND*" means the matcher *Name*(*N* for short) for classes category, and the matcher *Domain and Range*(*D* for short) is used for properties category. The combination methods are *Avg* (*Average*) and *Alh* (*At least half*) based on all single matchers for each category (see Section 5.2 and the first paragraph of Section 6).

It has been shown that f-Measure increases with the increase of recall from *NN* to *Alh*. Obviously, it is much more helpful to use several matchers together at a time than just use one matcher to get better results, because more information can be obtained for multiple matchers than that for one matcher.

6.3 Comparing different LAOs

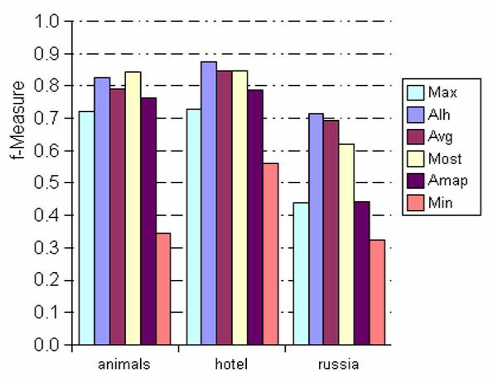

Figure 3. The performance of several LAOs

Due to the existence of some composite matchers like *Taxonomy* for our experiments, the aggregation should be executed twice. One is for the composite matchers to combine their constituent matchers. For properties category, it needs to aggregate first the results of the constituent matchers, *Super properties* and *Sub properties*, for *Taxonomy.* The second aggregation is to combine all the base matchers, *Name*, *Domain and Range* and *Taxonomy* to get the final similarity matrix.

Since *Max*, *Min* and *Avg* are not only existing aggregation operators without weights to matchers, but the special cases in LAOs, we choose six LAOs including the three special operators to compare their performance. From Figure 3, we can see that: *Alh*, *Most*, *Avg* and *Amap* (*As many as possible*) outperform *Max* and *Min.* Because *Max* and *Min* are extra optimistic or too pessimistic, that is to consider one extreme similarity value at a time. While *Alh*, *Most*, *Avg* and *Amap* combine some or all the similarity values. Moreover, *Alh* could perform better than *Avg* in most cases while *Most* outperforms *Avg* in some cases because of their own characteristics.

6.4 The comparison of *Average* and *At least half*

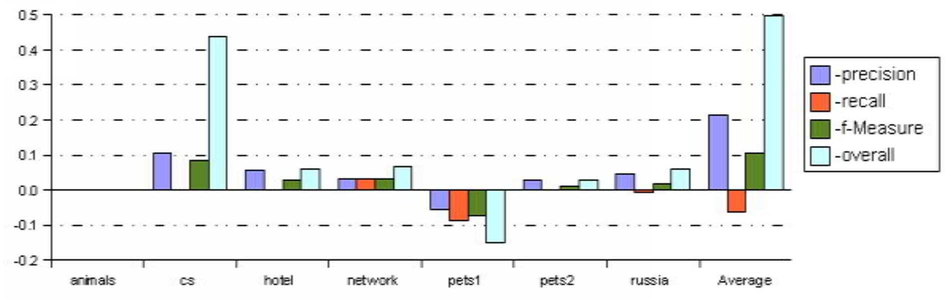

Figure 4. Compare the performance of *Avg* and *Alh* in LCS

As *Avg* outperforms *Max* and *Min* as existing aggregation operators, we use *Alh* to compare with *Avg* on seven ontology pairs to give more details on their performances. The purpose of this experiment is to give more detail on that some operators in LAOs like *Alh* could perform better than existing aggregation operators like *Avg* in most cases. As we have said, we do not consider the weights to matchers.

Based on the matching process in LCS and the base matchers we have chosen, we compare the performance of *Avg* and *Alh* on all the ontology pairs we have introduced above. See Figure 4, the data in Y axis is computed by subtracting the results of *Avg* from the results of *Alh*, where the results are expressed by precision, recall, overall and f-Measure. We use "-" in the figure to indicate the subtraction. For example, "-precision" indicates the subtraction by subtracting the precision of *Avg* from the precision of *Alh* on a specific ontology pair (O_1, O_2) (i.e., $-precision = precision_{Alh}(O_1, O_2) - precision_{Avg}(O_1, O_2)$).

From Figure 4, *Alh* outperforms *Avg* on all the ontology pairs except animals and pets by higher precision and f-Measure basing on the similar recall for each ontology pair. For animals, the performance for *Avg* and *Alh* is the same. The overall of *Alh* is only reduced in one out of the seven ontology pairs, while is increased in five of the seven pairs with the highest increase near 0.5, which means nearly 50% manual effort is saved.

7 Conclusion and Future Work

Ontology matching is an essential solution to deal with the interoperability of heterogeneous systems. It has been proved that, in most cases, combining the results of multiple matching approaches or matchers is a promising technique to get better results than just using one matcher at a time [2–4, 10]. Due to the limitations of existing combination methods, we propose a new system LCS where a LAO is used for the aggregation. Through experiments, the power of LCS has been shown.

The main contribution of this paper is the introduction of OWA operator to ontology matching. The weight here is not associated with a specific matcher but a particular ordered position. We choose the linguistic quantifiers to determine the OWA weights. To our convenience, we name the OWA operator based on the linguistic quantifier to obtain weights as the linguistic aggregation operator (LAO). So a large number of LAOs can be defined according to different linguistic quantifiers. Besides, we provide a semantic interpretation of LAOs to facilitate users to select an appropriate LAO for the aggregation. Specially, some existing aggregation operators like *Max*, *Min* and *Average* are the special cases in LAOs.

From the experiments (see Figure 3 and 4), we can see that some LAOs like *Alh* and *Most*, can perform better than *Max*, *Min* and *Avg* in most cases. So LAO provides a good way to supply a gap for existing aggregation operators without considering the weights to matchers.

In the future, we will further develop the application of OWA operators to combine multiple ontology matchers. Specifically, the following aspects are involved: First, since we intend to compare different combination methods without weights to the matchers, we provide a simple platform for such comparison. In our further work, we will compare the performance of our system with other systems. Last but not the least, we did not consider the weights of matchers, not because they are not important, but we want to propose a flexible and efficient way to aggregate the results of multiple matchers when it is not necessary to use weights. If the weights of matchers can be obtained by experts or machine learning, we can use the weighted OWA operators [19] for the aggregation.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. *Scientific American*, 284(5):34-43, 2001.

2. Do, H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the 28th VLDB Conference*, pp. 610-621, 2002.
3. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pp. 509-520, 2001.
4. Ehrig, M., Sure, Y.: Ontology mapping - an integrated approach. In *Proceedings of the First European Semantic Web Symposium, ESWS 2004, Volume 3053 of Lecture Notes in Computer Science*, pp. 76-91, Heraklion, Greece, 2004. Springer Verlag.
5. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In *Proceedings of the Twenty-seventh International Conference on Very Large Data Bases(VLDB)*, pp. 49-58, Roma, Italy, 11-14th September 2001. Los Altos, CA, USA, Morgan Kaufmann Publishers (2001).
6. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of Eighteenth International Conference on Data Engineering*, San Jose, California, 2002.
7. Euzenat, J. and Valtchev, P.: Similarity-based ontology alignment in OWL-Lite. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pp. 333-337, Valencia, Spain, 2004.
8. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases(VLDB)*, 10(4): 334-350, 2001.
9. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. *Journal on Data Semantics*, No. 4, LNCS 3730, pp. 146-171, 2005.
10. Tu, K., Yu, Y.: CMC: Combining mutiple schema-matching strategies based on credibility prediction. In *Proceedings of the 10th International Conference on Database Systems for Advanced Applications (DASFAA)*, LNCS 3453, pp. 17-20, 2005, China.
11. Yager, R.R.: On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Trans. on Systems, Man and Cybernetics*, 18(1988): 183-190.
12. Xu, Z.: An overview of methods for determining OWA weights. *International Journal of Intelligent Systems*, 20(8): 843-865, 2005.
13. Yager, R.R.: Family of OWA operators. *Fuzzy Sets and Systems*, 59(1993): 125-148.
14. Yatskevich, M.: Preliminary evaluation of schema matching systems. *Technical Report # DIT-03-028*, Department of Information and Communication Technology, University Of Trento (Italy) (2003).
15. Yager, R. R. and Kacprzyk, J.: *The Ordered Weighted Averaging Operation: Theory, Methodology and Applications.* Kluwer Academic Publishers, pp. 167-178, Boston, 1997.
16. O'Hagan, M.: Aggregating template or rule antecedents in realtime expert systems with fuzzy set logic. In *Proceedings of the 22nd Annual IEEE Asilomar Conference on Signals, Systems, Computers*, pp. 681-689, Pacific Grove, CA, 1988.
17. Herrera, F., Herrera-Viedma, E. and Verdegay, J.L.: A sequential selection process in group decision making with a linguistic assessment approach,*Information Sciences*, 85 (1995), pp. 223-239.
18. Do, H., Rahm, E.: Comparison of schema matching evaluations. In *Proceedings of the second international workshop on Web Databases (German Informatics Society)*, 221-237, 2002.
19. V. Torra, The Weighted OWA operator, *International Journal of Intelligent Systems*, 12(1997): 153-166.