Ranking Entities from Multiple Ontologies to Facilitate Federated Queries

Neda Alipanah1, Julie Rauer1, Latifur Khan1, Steven B. Seida2, Bhavani Thurasingham1

Department of Computer Science
University of Texas at Dallas1
Raytheon Corporations2

{na061000, jrr053000, lkh, bxt043000}@utdallas.edu1, steven_b_seida@raytheon.com2

Abstract. In view of the need of highly distributed and federated architecture, integrating schemas from different data sources in a specific domain has great impact on the performance of web applications. As the web scales, there are ample sources that provide structured information (ontologies) in the same domain. Since ontologies for a same domain usually overlap, we aim to determine a global ontology based on the commonality of overlapping entities using an efficient algorithm. Our algorithm examines ontologies by considering entities and relationships between them in ontology’s graph. First it finds out the Largest Common Subgraph (LCS) between two ontologies. For this, LCS is aligned to other ontologies using the lexical and structural similarity of entities. Next, we propose a novel statistical model based on maximum likelihood to determine a global ontology. This statistical model exploits the commonality of each entity across different ontologies. It will be helpful for federated query expansion.

Keywords: Federated Query, Global Ontology, Query Expansion, Ontology Matching.

1. Introduction

The internet continues to point out the benefits of highly distributed, federated architecture to solve many web problems. However, there are always several related data sources to be considered for web applications. Different data sources have their own related ontologies. Federated query utilizes efficient middleware to search multiple geographically and technologically disparate data sources. It extracts, transforms and presents query responses as if the data sources were all one. Ontologies from same domain usually overlap with each other. The overlapped segment is a good indicator for the quality of ontology. For example, consider several nation-based ontologies which present knowledge about countries like food,
agriculture, tourism and industry of the countries. They include several concepts and their relationships between them. When some concepts overlap between these ontologies, those concepts have a high probability of correctness and good quality. Finding the common segments in different ontologies helps to rank entities of ontologies based on their frequencies. Recall that documents will be associated with entities in ontologies. For federated query result, a set of documents will be retrieved; some of them will be highly ranked and some of them not. These highly ranked documents may appear in the top of the result set due to their association with these common segments.

In developing a strategy for specifying global ontology by their verified entities, we consider ontology alignment approach. Some previous works exist for ontology alignment based on instance matching [17][18][19]. Partyka et al. [17] explain using EBD by extracting keywords on instances and group distinct keywords in semantic clusters to calculate similarity. Wang et al. [18] approach the mapping problem as a classification problem based on similarity between instances of concepts. Wartena et al. [19] use approximate similarity measure between concepts based on distribution of annotations. In our work, we extract all entities and relations between them using RDF graph of ontologies. RDF graph can interchangeably be defined by set of triples called RDF Sentences. RDF graph is the basement for determining the global ontology in mediator layer. First, we find out the Largest Common Schema Subgraph (LCS) between two ontologies using the name similarity and structural similarity of RDF graph with an efficient algorithm [2]. Then LCS is used for discovering the LCS of all other ontologies. The final LCS between ontologies indicates the frequency of repetition of entities between all ontologies. Frequencies have a great role in specifying the statistical model to generate global ontology for the purpose of query expansion in federated queries.

Based on the Zipf law, high ranked entities occur more frequently in different ontologies [6]. Recall that federated queries return a number of documents. Documents associated with these highly ranked entities will come before those associated with low ranked entities. We present the statistical model for ranking entities and use a maximum likelihood of how frequencies are effective in generating global ontology.

Our work relies on finding the largest common subgraph between ontology graphs using Wang algorithm [2]. They use three measurements to find the similarity of subgraphs including node attribute similarity, edge attribute similarity and minimizing degree difference of nodes. Our approach leverages some of these ideas along with some new variations towards the construction of a novel algorithm which best fits the ontology’s semantic networks. In ontology alignment, we consider different thresholds for name similarity of concepts and structural similarity between them. Finally, we determine the best threshold and use it for our multiple ontology matching problems.

The contribution of our paper is as follows:
Ranking Entities from Multiple Ontologies to Facilitate Federated Queries

- Finding the Largest Common Subgraph (LCS) between RDF graphs of multiple ontologies and specifying the overlapped entities between different ontologies.
- Defining a Statistical Model based on the frequency of entities in different data sources.
- Determining appearance probability of each entity in GO using maximum likelihood estimation and rank it based on this probability.

The rest of this paper is organized as follows. In section 2, we present a survey on federated architecture and the problem statement. In section 3, we propose the Largest Common Subgraph (LCS) algorithm for aligning two or more ontologies using the name and structural similarity. Further, we present a statistical model to generate global ontology and use maximum likelihood estimation to calculate the probability of each entity appearing in the global ontology. In section 4, we present experimental results of aligning I’CON conference benchmark tests for pairwise and multiple ontology matching. Finally, in section 5 we present conclusions and future work.

2. Related Works

Different research on ontology ranking has been accomplished. Alani et al. [1] use the analysis of concept structure in AKTiveRank system. They consider relevance of multiple query keywords for ontology ranking. The ranking of ontologies reflects the importance of each relevant vocabulary which is defined by centricity and density of the vocabularies. Centricity presents the distance of a vocabulary to the central level of its hierarchy and density is a weighted summation of the number of its subclasses, siblings, relations and instances. KeWei et al. [3] describe the approach of finding important classes of users. Importance of a class is qualified by the level of classes or having many descendents. It is calculated by a formula using the linear combination of total importance of its direct subclasses and a function of its depth in the class hierarchy. Our ontology ranking approach is more confident and accurate since it defines the important vocabularies based on the frequency of using them in multiple ontologies. It is different from both stated approaches because they consider concept level, density and centricity of a single ontology in their ranking. They do not consider multiple ontologies, hence no ontology alignment issue is addressed.

Our approach for entity ranking relies on a novel efficient algorithm for determining the Largest Common Subgraph (LCS) in the semantic network of ontologies. Wang et al. [2] produce matching pairs for all nodes of two input graphs and create sorted lists for every node. They determine the similarity of two nodes based on degree differences of nodes and their direct neighbors, node attribute similarity and edge attribute similarity. They calculate the degree of each node along with the degree of their neighbors. Also, they determine node attribute similarity which examines how many direct neighbors of each node are the same in two different graphs. Finally, they find out the edge attribute similarity based on the edge values for nodes. They build the maximum subgraph by depth first search traversing the graph. They also compare the algorithm results with two fundamental maximum common subgraph algorithms.
called McGregor’s [4] and Durand-Pasari [5] algorithms. In our method, we use the same structure for matching concepts from the semantic of two ontologies. We compute the edge attribute similarity by finding the cosine similarity of edge types and neighbors. Finally, we use a specific threshold for the average of all similarities and determine the maximum common subgraph.

Our work uses a similarity-based approach in ontology matching for ranking entities including concepts, object properties, data properties and individuals. There are three methods in this regard including HCOME-merge [8], SCM [7], and ASCO [9]. HCOME-merge uses Latent Semantic Analysis (LSA). For each concept, it finds word senses lexicalized by C or its variations by looking up WordNet and expanding its hyponymy. However, it only considers subclass, superclass vicinity of each concept. SCM is based on Vector Space Model. It is limited to match class hierarchies. High-ranking words are extracted from instances and subclasses of a given class to calculate similarity measure. Structural consistency analysis is used to evaluate and adjust mapping. Qu et al. [10] propose to use virtual documents for ontology matching. They determine a collection of weighted words for a URI reference. The collection contains the local and neighbor information and uses traditional vector space model to find the similarity between ontologies.

ASCO uses the available information contained in ontology for discovering mapping between classes. Linguistic similarity is a linear combination of name similarity, label similarity and description similarity. Jaro-Winkler metric is used for finding string similarity. Wordnet is also used to find out the synonyms in the calculation of name similarity. In the latest version of ASCO algorithm, the authors suggest a graph-based algorithm for aligning OWL ontologies. They extract an O-graph such that classes, relations and instances are nodes in the new graph. They use the RDF triples to construct the graph and they use owl:equivalentClass, owl:equivalentProperty and owl:sameAs between object nodes and subject nodes [11]. They use the maximal clique detection approach to find the maximal subgraph between only two ontologies. Therefore, they define an association graph which uses the node compatibility and property compatibility between two graphs. In our work, we use an efficient algorithm to find the common subgraph between multiple ontologies, which is not the case in ASCO method. Our graph covers all the information of the ontologies. First, it finds the matches across two ontologies. Then the result is used to find common entities for remaining ontologies. Furthermore, we define a statistical model to aggregate ontologies based on commonality of entities. The idea of integrating schema vocabularies was proposed by different researchers [12][13][17]. Doan et al. [12] propose a system to mediate schemas that use machine learning techniques. Do et al. [13] propose a platform to combine multiple matches for integrating schemas. The authors focus on using name path, type name, name similarity and children similarity in data sources. They determine a similarity cube to combine different similarities. Our work is different from [12][13] in the sense that we are integrating ontologies using LCS between graphs of ontologies. He et al. [6] observe that when sources proliferate, their aggregate schema vocabulary tends to converge at a relatively small size. They define a statistical model regarding the general distribution of all input schemas. They randomly select some set of partitioned attributes as input.
schema and use maximum likelihood to discover schema model. In our work, we concentrate on the distribution of attributes across different resources and we propose a statistical model based on the frequency of attributes. Our statistical model uses maximum likelihood to find out the probability of each entity and attribute appearing in the global ontology. In the next section, the details of our approach are explained.

3. Ranking Entities for Effective Federated Query Expansion

In federated architecture, different external data sources are examined to find out the response for each query. Federated architecture is considered in different layers including data source layer, mediator layer and application layer. In data source layer, different ontologies with different levels of coverage and accuracy are considered. In mediator layer, an integrated schema called Global Ontology (GO) exists which covers entities of different data sources. In application layer, users’ queries and responses are considered. Integrating schemas to construct the GO has a great role in the correctness of query responses. GO verifies the accuracy of entities and plays a critical role for the result of query expansion. As an example, consider the Culture and Art query of season and location of winter annual festivals in Russia. To answer this query, it is required to consider different Russia ontologies distributed across multiple data sources. All ontologies contain some information relevant to culture and art in Russia. By aligning festival entity across Russia ontologies, we obtain all the information regarding winter festivals in Russia ontologies. Therefore, we are able to find out the correct answer for the query based on common entities. Figure 1 presents the architecture of federated query expansion. We can see that in mediator level, GO helps to determine the correct source for effective federated query expansion among multiple ontologies.

Figure 1: Federated query architecture.
3.1. Problem Statement

Given different ontologies, $O_1$ to $O_n$, all describing semantic information on a particular domain with different level of coverage, the goal is to evaluate commonality of entities (concepts, object properties, data properties, individuals) across them and rank entities to be used for an effective federated query expansion. Challenges exist in different steps. First, it is required to align entities from one ontology to all its correspondences in other ontologies. Second, we need to determine a global schema to contain information on all ontologies called Global Ontology (GO). Finally, we need a mechanism to rank each entity in GO. With regard to the first challenges, entity alignment is done by identifying Largest Common Sub-graph (LCS) among $N$ ontology graphs. These ontologies vary in breadth, depth and the relationship types between their constituent entities. For instance, considering the Russia domain, we need to find LCS between Russia1, Russia 2, Russia A and Russia B ontologies and rank entities respectively. Figure 2 and Figure 3 present a portion of Russia A and Russia B ontologies. As depicted in the figures, different entities exist regarding the festivals. Russia A explains about “Russian_Winter_Folk_Festival” in in Suzdal province while Russia B presents information about the “Russian_Winter_Festival” in Moscow. Therefore, we need to align entities across them for GO.

![Figure 2: Russia A Ontology](image1)

![Figure 3: Russia B Ontology](image2)

To construct the GO, an effective algorithm is required to find LCS among different ontologies. LCS enables us to find out the commonality of entities between ontologies. For this, we need to consider ontologies pair wise and find the LCS. Further, we align other ontologies with the LCS and find out entities which are more common. Based on Zipf’s law, the frequencies of entities across different ontologies are inversely proportional to their ranks [6]. Therefore, GO enable us to rank entities based on commonality of them.
3.2. Proposed Solution

In this section, first, we concentrate on the detail of an effective algorithm to find out commonality of entities across ontologies. Our algorithm determines the LCS between RDF graphs of ontologies based on name and structure similarity of entities. Next, we explain about the construction of global ontology and a statistical model to estimate entity ranks based on their frequencies.

3.2.1. Largest Common Subgraph (LCS) between Ontologies Graphs

In our method, an RDF graph is used to exhibit the structure of ontologies for the purpose of entity ranking. The RDF graph of OWL DL ontology presents necessary elements for describing the semantic structure of ontologies.

**Definition 1 (Ontology Graph):** Ontology graph is a directed, cyclic and connected graph $G = \langle V, E \rangle$, where $V$ include all the entities of ontology and $E$ is a set of all properties between entities.

**Definition 2 (RDF Sentences):** Given an ontology $O$ with corresponding graph, we define RDF sentences as RDF triples of the graph. Sentences are represented as $(s, p, o)$, where $s$, $p$, $o$ are respectively subject, predicate and object. Therefore, $O = \{(s, p, o)\}$.

**Definition 3 (Ontology Vocabulary):** Ontology vocabulary of an RDF graph is all subjects and objects that are RDF URI references and defined in RDF sentences. They cover all entities of an ontology including concepts, object properties, data properties and individuals. They do not belong to the built-ins provided by Ontology Languages.

3.2.2. LCS Matching Algorithm

In order to find out the commonality of ontology vocabularies in different ontologies in a federated system, we propose an efficient algorithm to find the LCS between a pair of ontology graphs. LCS gives us the corresponding entities in two graphs and is used for further matching across other ontology graphs.

**Entity Matching across Multiple Ontologies**

In this section, we present the algorithm to find the LCS for multiple ontologies. First, the LCS for each pair of ontologies is calculated. This new subgraph is counted as a new ontology and is compared with other ontologies to determine the commonality of entities between all of the ontologies.
Algorithm 1: LCS for multiple ontologies

Input: \( O_1, O_2, \ldots, O_n \)

Output: LCS between \( O_1, O_2, \ldots, O_n \)

1: \( \text{MaximumCommonSubgraph} \leq \text{LCS}(O_1, O_2) \)
2: For all ontologies \( i = 3 \) to \( n \) do
3: \( \text{MaximumCommonSubgraph} \leq \text{LCS}(\text{MaximumCommonSubgraph}, O_i) \)
4: End for
5: return \( \text{MaximumCommonSubgraph} \)

In line 1, the algorithm finds the largest common subgraph between two ontologies using an efficient algorithm which is described in the next section. In line 3, it compares the largest subgraph with remaining ontologies by choosing one of them at a time and finds out the new LCS. This process continues for all input ontologies. Thus, by using LCS, all the common entities across several ontologies will be identified. Now we need to construct a global ontology with corresponding rank of each entity. For this, we define a statistical model based on maximum likelihood as explained in the next section.

Pairwise Matching

In this section, we present our algorithm to find LCS between ontology graphs. The algorithm determines the LCS of two ontologies using similarity bounds in the graph of ontologies as follows.

**Step 1:** Given \( O_1 \) and \( O_2 \), the algorithm produces the matching pairs of vocabularies for input ontologies such as concepts \( C_1 \in O_1 \) and \( C_2 \in O_2 \) along with a similarity measurement between them. Matching pairs are placed in a sorted linked list according to the similarity measurement, so that given a particular entity from \( O_1 \), the most similar corresponding concept in \( O_2 \) can be retrieved as the first element of the list. In our algorithm, we define a similarity function as a combination of Name similarity and structural similarities. The algorithm works in the following way: First, we examine all the entities from the first ontology and create a sorted list for them. Second, we compare each of the entities of the first ontology with the second ontology and find out separately their name similarity and structural similarity. Name similarity between entities is measured by the Jaro Winkler distance method [14]. Structural similarity between entities is estimated by type of edges between corresponding entities in sorted linked lists. Structural similarity is calculated by neighbor similarity, and difference of edge types. Finally, the corresponding sorted list for each entity is stored in LCS between two ontologies. Detail of structural similarity measurement is explained below.

Neighbors Similarity

We define a procedure to calculate similarity of concepts from two different ontologies based on their direct neighbors in an ontology graph. The average of similarities between neighbors of two concepts is calculated as a Neighbor Similarity measurement. First, we consider all the neighbors of corresponding concepts from \( O_1 \)
and O₂. Next, we check if the direct neighbors are the same using name match; we increase total similarity value; and finally, we normalize the total similarity value.

**Similarity of Relation Types (SRT)**
Considering each pair of similar entities from linked list like O₁-C₁ and O₂-C₂, we define two separate Relation Type Vectors (RTV) corresponding to C₁ and C₂. Each vector has “n” entries according to the relation types of entity and its neighbors. By checking each relation type for concepts and neighbors, corresponding RTV entries are increased as an indicator for the number of relation types. Using RTV (C₁) and RTV (C₂), we are able to find the cosine similarity of edges for C₁ and C₂. Algorithm 2 presents the detailed steps for calculating SRT as follows.

**Algorithm 2: Similarity of Relation Types (SRT)**

**Input:** Entity C₁ ∈ O₁, Entity C₂ ∈ O₂, Relation Types (C₁, Neighbors), Relation Types (C₂, Neighbors)

**Output:** Overall Neighbor Similarity

1: RTV₁ = [0, 0, ..., 0], RTV₂ = [0, 0, ..., 0]
2: DRT = 0
3: For all ns₁ = neighbors of (O₁-C₁) do
   4: Check relation Type i ((ns₁, O₁-C₁), (ns₁, neighbor))
   5: RTV₁[i] = RTV₁[i]+1
6: End for
7: For all ns₂ = neighbors of (O₂-C₂) do
   8: Check relation Type j ((ns₂, O₂-C₂), (ns₂, neighbor))
   9: RTV₂[j] = RTV₂[j]+1
10: End for
11: SRT(C₁,C₂) ≤= Cosine Similarity(RTV₁, RTV₂)
12: return SRT(C₁,C₂)

In line 3, we find all the relation types for concept C₁ and create the type array for it. In line 4, we find out the type array for C₂ from O₂. In line 11, cosine similarity between two arrays is found and the sorted link list for each concept is updated based on the corresponding total structural similarity calculated by NS and SRT.

**Step 2:** In this step, LCS is built through a series of linked lists by choosing a pair of similar concepts from each linked list. The algorithm defines a priority queue. Whenever an entity is selected from the ontology, all of its children are added to priority queue. The algorithm uses a similarity threshold to perform pruning. If similarity value is less than threshold, that branch is ignored and the algorithm keeps track of concepts with the remaining children in priority queue. Each entity is checked to ensure that it already doesn’t appear in the path.

**Step 3:** The procedure of finding LCS is continued until all paths have been considered. The procedure stops when the threshold requirement satisfies, if a threshold size was specified.
With the LCS of two ontologies, we are now able to use the pairwise LCS for the matching process across other ontologies as explained in Algorithm 1.

3.2.3. Statistical Model and Global Ontology

Using the overlapped analysis between different ontologies from the previous section, we specify a statistical model based on Zipf's law. The frequencies of entities between different ontologies in a specific domain are inversely proportional to their ranks. High ranked entities occur more frequently in different data sources [6]. This observation leads us to specify a statistical model to generate GO using the vocabularies and constraints on them.

Model Structure

We make some assumptions to reasonably define statistical model for creating the GO. First, we assume that GO entities are selected independently. Second, entities are non-overlapping; that is, they don’t have any semantic overlap. Third, attributes are mutually exclusive. We assume that in generating global ontology, no two corresponding attributes will both be selected, since they are representing one entity. Based on our assumptions, we define a statistical model as follows (Figure 4). For each entity in the original ontologies, we specify a node which represents the probability of appearance of the entity in global schema. Vocabulary of the global schema includes the union of all vocabularies in different ontologies.

![Figure 4: Global Ontology Structure](image)

**Definition 4**: Global Ontology (GO) is defined as a statistical structure with a 4-tuple (V, E, Pe, Pa). The vocabulary V is a set of all entities in ontology including concepts, object properties, data properties and individuals as follows.

\[ V = \bigcup_{i=1}^{n} \text{Entity}_i \quad \land \quad \text{Entity}_i \cap \text{Entity}_j = \emptyset \]

Pe is the Entity Probability function, which determines the probability \(\alpha_i\) in GO generating. Pa is the Attribute Probability function which determines the probability \(\beta_j\) for selecting Attribute\(_j\), once its related Entity is already selected. Note that the GO Statistical Model represents a generative distribution of any entity in different ontologies of a same domain. Formally, GO is presented as a subset of attributes with probability annotations. The GO statistical model for figure 4 is as follows:
Ranking Entities from Multiple Ontologies to Facilitate Federated Queries

Each entity can appear in the GO with probability \( \Pr(\text{Entity}|\text{GO}) = \alpha_i \), while each attribute appears in GO with the probability

\[
\Pr(\text{Attribute}_j|\text{GO}) = \beta_{ij} \quad \exists i: \text{Attribute}_j \in \text{Entity}_i \\
0 \quad \text{Otherwise} \tag{1}
\]

The probability of having a set of entities in GO is as follows:

\[
\Pr(\text{Entity}_1, \ldots, \text{Entity}_j|\text{GO}) =
\begin{cases}
0 & \exists j \neq k: \exists i: \text{Attribute}_j \in \text{Entity}_i \land \text{Attribute}_k \in \text{Entity}_i \\
\prod \Pr(\text{Entity}_j|\text{GO}) & \text{Otherwise} \tag{2}
\end{cases}
\]

Putting it together, we can derive the probability of instantiation of any combination of entities and attributes in the GO as stated in equations 2. Let \( n_1 \) = number of corresponding attributes across a particular entity \( i \). For each possible GO, the instantiation equation is as follows:

\[
\Pr(\text{Entity}_1, \ldots, \text{Entity}_k|\text{GO}) = \alpha_1^{n_1} \cdot \alpha_2^{n_2} \cdots \alpha_k^{n_k} \tag{3}
\]

Also, the probability of having all entities from different ontologies in GO is 1 as equation 4 presents.

\[
\sum_{\text{Entity}_i \in \text{GO}} \alpha_i = 1 \tag{4}
\]

For each entity, related attributes can have different origins. In each ontology, it is possible to have a number of attributes that match some attributes of other ontologies. Therefore, in our statistical model, we consider the original ontology for a set of attributes in a specific Entity group. The summation of their probabilities is equal to the summation of the probabilities of other attributes from other original ontologies as equation 5 presents. Also, the total summation of all attributes appearing in the entity group is 1 as equation 6 presents.

\[
\forall \text{Attributes} \in \text{Entity}_i, \sum_{\text{Attribute}_j \in \text{Ontology}_p} \beta_j = \sum_{\text{Attribute}_k \in \text{Ontology}_q} \beta_k \tag{5}
\]

\[
\sum_{\text{Attribute}_j \in \text{Ontology}_p} \beta_j + \sum_{\text{Attribute}_k \in \text{Ontology}_q} \beta_k = 1 \tag{6}
\]

Intuitively, each entity may have different attributes from different ontologies, but the global ontology must have only one instance for each entity and from each ontology only one attribute appears in the GO. Therefore, each entity is identified with a pair of probabilities in GO: first the probability of appearing in its related entity \( (\alpha_i) \) and second the probability of the attribute \( (\beta_j) \).

**Example 1:** Given two ontologies, Animals A and Animals B depicted in figure 5 and figure 6. We find the following alignments by our LCS algorithm.
In this section, we present the result of our experiments on benchmark ontologies of I-\textsuperscript{CON}. Our LCS was implemented in Java. Jena API was used for manipulating
Ranking Entities from Multiple Ontologies to Facilitate Federated Queries

ontologies using their RDF graph [16]. The OWL DL graph representation is derived with a gruff tool [15].

4.1. Datasets

We test our algorithm for both pairwise ontology matching and multiple ontology matching. In pairwise alignment, we use all test cases including the Animals, Computer Networks and People+Pet ontologies. We evaluate our algorithm for multiple comparison for 3 ontologies including Russia 1, Russia 2 and Russia A ontologies.

4.2. Measuring Performance

We report the performance of our approach by using standard information retrieval metrics to assess our results. We focus on the average F-measure for our matching purpose between different ontologies.

4.3. Discussion on Experiment

In our experiments, first we extract the RDF Graph of ontologies by Jena API. We extract different entities of each ontology as presented in table 1.

Table 1: Extracting the Concepts and Properties in different Ontologies

<table>
<thead>
<tr>
<th>Ontologies</th>
<th>Class</th>
<th>Object Properties</th>
<th>Data Properties</th>
<th>Individual Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>RussiaA</td>
<td>151</td>
<td>60</td>
<td>16</td>
<td>157</td>
</tr>
<tr>
<td>RussiaB</td>
<td>103</td>
<td>46</td>
<td>12</td>
<td>125</td>
</tr>
<tr>
<td>AnimalsA</td>
<td>10</td>
<td>12</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>AnimalsB</td>
<td>10</td>
<td>11</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Soccer</td>
<td>265</td>
<td>3</td>
<td>3</td>
<td>124</td>
</tr>
<tr>
<td>Basketball</td>
<td>171</td>
<td>3</td>
<td>2</td>
<td>107</td>
</tr>
<tr>
<td>HotelA</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>HotelB</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Pet+PeopleA</td>
<td>60</td>
<td>14</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>Pet+PeopleB</td>
<td>58</td>
<td>14</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Pet+PeopleA (no instance)</td>
<td>60</td>
<td>14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Pet+PeopleB (no instance)</td>
<td>58</td>
<td>14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Network A</td>
<td>27</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network B</td>
<td>27</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1 presents the number and type of entities related to each ontology from I3CON benchmark: for example Russia A has 384 entities including 151 classes, 60 object
properties, 16 data properties and 157 individuals. In our experiments, we match ontologies using all related entities. Further, we examine ontologies pairwise and define the largest common subgraph between them. Using the LCS algorithm, we find the frequency of entities across ontologies. Our experiments are performed in two different methods to find structural similarity. First, we measure structural similarity using the relation type similarity (RTS) as explained in section 3. We have used the threshold of 0.9 and measure the performance of LCS as shown in figure 8. The X axis represents different matching ontologies and the Y axis represents the f-measure related to our ontology comparison. We compare our result with two other sets of results from I’CON participants in figure 8. In some test cases including the computer network, Pets and Pets (No instance), LCS improves f-measure for matching entities compared to other methods. We calculate f-measure for Network as 0.88, Pets as 0.95 and Pets (No instance) as 0.93.

Next, we measure structural similarity by finding both the relation type and neighbor similarity as shown in figure 9. The X axis presents different ontologies and the Y axis shows different f-measures. Comparison of the three different methods is depicted in figure 9. The average of f-measure in our method is 0.71 with RTS calculation and 0.68 with RTS and neighbor matching.

![Figure 8: Pair wise Ontology Matching using Name Similarity + RTS.](image1)

![Figure 9: Pairwise Ontology Matching using Name Similarity + (RTS and Neighbor).](image2)

We extend our experiments for finding commonality between 3 ontologies. We create new test cases by editing the I’CON Ontologies. We construct Animals C Ontology by editing Animals B and removing two classes “TwoLeggedThing” and “TwoLeggedPerson”. Also, we create Pets C by removing classes including
Ranking Entities from Multiple Ontologies to Facilitate Federated Queries

Publication, corresponding subclasses {magazine, newspaper, broadsheet, quality-broadsheet, tabloid, red_top}, Object properties {reads} and related individuals {The_Guardian, The_Times, The_Sun}. For Animals 3 ontologies and PetsPeople 3 ontologies, LCS achieves 0.94, and 0.91 precision respectively; 0.7, 0.95 recall respectively and 0.76, 0.93 F-measure respectively.

5. Conclusions

In this paper, we have outlined an efficient algorithm that finds the largest common subgraph on the graph of ontologies using name and structural similarity. We focused on ranking entities and constructing a global ontology by extracting the largest common subgraph between ontologies. Largest common subgraph determines the commonality of entities in multiple ontologies. Next, we propose a statistical model which identifies the probability of entities appearing in global ontology based on maximum likelihood. Furthermore, we illustrated and discussed the results of a series of experiments which displayed the result of the ontology matching algorithm over two or more ontologies. Finally, we provided a more in-depth look at some ontology ranking computations between different ontologies. Future efforts regarding global ontology will focus on improvement of ontology matching. We will seek a more accurate and complete common subgraph by background knowledge. We will also explore some pruning heuristic to have better performance on our common subgraph determination.

References