# Partitioning and Local Matching Learning of Large Biomedical Ontologies

Amir Laadhar, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri

# Partitioning and Local Matching Learning of Large Biomedical Ontologies

Amir Laadhar
Institut de Recherche en Informatique de Toulouse (CNRS/UMR 5505)
Toulouse, France
amir.laadhar@irit.fr

Faiza Ghozzi
Université de Sfax, MIRACL
Sakiet Ezzit, Sfax, Tunisia
faiza.ghozzi@isims.usf.tn

Imen Megdiche
Institut de Recherche en Informatique de Toulouse (CNRS/UMR 5505)
Toulouse, France
imen.megdiche@irit.fr

Franck Ravat
Institut de Recherche en Informatique de Toulouse (CNRS/UMR 5505)
Toulouse, France
franck.ravat@irit.fr

Olivier Teste
Institut de Recherche en Informatique de Toulouse (CNRS/UMR 5505)
Toulouse, France
olivier.teste@irit.fr

Faiez Gargouri
Université de Sfax, MIRACL
Sakiet Ezzit, Sfax, Tunisia
faiez.gargouri@isims.usf.tn

## ABSTRACT

Conventional ontology matching systems are not well-tailored to ensure sufficient quality alignments for large ontology matching tasks. In this paper, we propose a local matching learning strategy to align large and complex biomedical ontologies. We define a novel partitioning approach that breakups large ontology alignment task into a set of local sub-matching tasks. We perform a machine learning approach for each local sub-matching task. We build a local machine learning model for each sub-matching task without any user involvement. Each local matching learning model automatically provides adequate matching settings for each local sub-matching task. Our results show that : (i) partitioning approach outperforms existing techniques, (ii) local matching while using a specific machine learning model for each sub-matching task yields to promising results and (iii) the combination between partitioning and machine learning increases the overall result.

## KEYWORDS

Semantic web, Ontology matching, Ontology partitioning, Machine Learning

## 1 INTRODUCTION

Ontologies are the backbone of the semantic web. They enable sharing, reusing and accessing the knowledge resources. Hundreds of large biomedical ontologies such as SNOMED CT, the National Cancer Institute Thesaurus (NCI), and the Foundational Model of Anatomy (FMA) are extensively employed in the biomedical domain [30]. These ontologies are based on diverse modeling views and vocabularies. Mapping these ontologies enables interoperability to share valuable data. The integration of knowledge bases requires efficient matching systems supporting the complex ontologies [9, 18–20, 29]. For instance, the cartesian product between the entities of NCI and SNOMED ontologies results in more than 45 billion comparisons. An agile solution is required, like divide and conquer or parallelization [10]. Ontology mapping becomes a challenging and time-consuming task due to the large volume of ontologies. To cope with these challenges, the traditional all-against-all ontology alignment methods are giving way to more efficient strategies (e.g. the divide and conquer approach) [9]. This approach decomposes a large matching problem into a set of smaller sub-matching tasks or partitions. Each partition focuses on a specific sub-topic of interest. Each partitioning approach is guided by the requirements of the target application domain [4]. The ontology alignment process consists of aligning similar partition-pairs from the two input ontologies. The partitioning process aims to decrease the matching complexity. However, several existing partitioning work result in a low coverage ratio coupled with isolated partitions. Moreover, the state-of-the-art partitioning approaches apply global matching settings (eg. matchers choice, thresholds, weights, etc.) for all extracted partitions. Consequently, these approaches do not consider the characteristics of each local sub-matching context. Therefore, they do not maximize the matching accuracy gain after performing the partitioning process.

In this paper, we take advantage of the partitioning process by delivering an automated local matching tuning for each sub-matching task. Tuning refers to the process of adjusting a matcher for a better functioning in terms of better quality of matching results and better performance [9]. We propose a partitioning approach, based on the hierarchical agglomerative clustering [22]. This approach produces a set of partition-pairs with a sufficient coverage ratio and without producing any isolated partitions. We automatically define the local matching tuning for each partition-pair by leveraging machine learning techniques. The existing approaches define a global machine learning model for the entire ontology matching task. Moreover, they employ the reference mappings to train the machine learning model. Nevertheless, these mappings are not

https://doi.org/10.1145/3297280.3297507

available and difficult to determine. Hence, we automatically derive for each sub-matching task the corresponding training set to build its machine learning model. The training sets are generated without any gold standard or user involvement. In addition, we align each sub-matching-task through its specific local matching parameters defined by its local machine learning model. Thus, every learning model takes into account the local matching context. As far as we know, there is a lack of approaches that perform the local matching learning.

The remainder of this paper is organized as follows. Next section presents the relevant related work. Section 3 introduces the set of formal foundations employed in our approach. Section 4 details our approach for large ontology partitioning and local matching learning. We report and discuss the evaluation results in Section 5. Finally, Section 6 presents our conclusions and perspectives.

## 2 RELATED WORK

In this section, we review the existing approaches related to ontology partitioning and matching learning. Matching learning refers to the use of machine learning techniques for ontology matching [9]. To the best of our knowledge, there is a lack of approaches that associate partitioning with matching learning to generate local matching learning.

**Ontology Partitioning** A line of work matching systems perform an all-against-all comparison between the entities of the input ontologies [10]. However, this strategy is not suitable for dealing with large and complex ontologies. The divide and conquer approach reduces the search space of pairwise ontology matching. Therefore, the matching process is applied to the set of similar partition-pairs between the input ontologies. Some approaches perform the partitioning process to ease the maintenance and the re-usability of a single large ontology [30]. Other lines of work xperform the partitioning process of pairwise ontologies to align the set of identified similar partitions (e.g., [2, 5, 7, 12–14, 27, 32]). Commonly, the partition-based matching strategies follow three tasks:

(1) The extraction of partition sets form the input pairwise ontologies.
(2) The identification of similar partitions between the set of extracted partitions of the input ontologies.
(3) The alignment between the set of identified similar partitions using a global matching tuning.

Recently, Jimenez-Ruiz et al. [16] proposed a divide and conquer approach that partition large ontologies into a set of sub-matching tasks. Jimenez-Ruiz et al. followed two clustering strategies: Naive strategy and neural embedding strategy. COMA++ [5] was one of the first matching systems proposing an ontology partitioning approach. Falcon-AO [13] and Hu et al. [14] employed an agglomerative structural clustering algorithm to divide an ontology into a set of disjoint partitions. Nonetheless, this approach is evaluated through only one pair of ontologies from the web directories matching task of the Ontology Alignment Evaluation Initiative (OAEI[1]). SeeCOnt [2] analyzed each input ontology to derive the root of each partition. SeeCOnt employed a ranking function that assigns each entity of an ontology to a single partition root. However, the number of roots is manually defined. Most of the existing work suffer from the low coverage value of the generated partitions [27]. For instance, the partitioning methods PBM, PAP, and APP obtained a coverage value of 80%, 34%, and 48%, respectively for the FMA-NCI matching task of OAEI [27]. To the best of our knowledge, most of the existing approaches apply the same manual tuning over all the extracted partitions. However, each partition represents a single sub-topic of interest. This sub-topic of interest has its own context and characteristics, which should be taken into account. In this paper, we divide a large ontology alignment to a set of sub-matching tasks called partitions. We align each sub-matching task using its local settings. We automatically determine the local matching settings using a specific machine learning model for each sub-matching task.

**Matching Learning** There have been some remarkable work on supervised matching learning, such as [8, 15, 24, 25]. Machine learning approaches for ontology alignment usually follow two phases [9]:

(1) The training phase: the machine learning algorithm learns the matching settings from a training set.
(2) The classification phase: the global learned matching configuration is applied over the input ontologies to classify the candidate mappings.

A matching learning approach can explore instance, structural, syntactic and semantic features [9]. Eckert et al. [8] built a meta-learner strategy to combine multiple learners. Malform-SVM [15] defines the ontology matching task like a machine learning problem. The training set is built from the reference alignment through a set of element level and structural level features. Nezhadi et al. [24] presented a machine learning approach to aggregate different types of similarity measures. This approach is evaluated through a relatively small bibliographic matching track provided by the OAEI campaign. Yam ++ [25] defined a decision tree model based on a training set that uses different similarity measures. The decision tree model is built from the reference alignment and applied over the matching tasks. However, each matching task has its specificities and a single global model cannot fit the characteristics of every mapping task. State-of-the-art matching learning approaches build a machine learning model from the reference alignment or manually derived for a particular matching task. The reference alignments are not always available. The generated machine learning model align every new matching task. However, each matching task has its characteristics and should be aligned through its adequate model and features. Hence, the generated machine learning model represents a single context that does not fit every new matching context. To maximize the accuracy for aligning partition-pairs, we propose to create a local learning model for each sub matching task rather than using a global model for the whole alignment problem. Moreover, we automatically generate the local training set for each sub matching task. We do not use any reference alignments or user interaction to build the local training sets. Each local training set represents the input of its local machine learning model, which provide adequate settings for each local matching context. To the best of our knowledge, there is a lack of approaches that provide automated local matching learning.

[1] http://oaei.ontologymatching.org/

## 3 FORMAL FOUNDATIONS

In the following, we present a set of definitions that we employ throughout this paper.

*Definition 3.1 (Ontology).* We encode an ontology $O_i$ as a directed acyclic graph (DAG), denoted by $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where:

- $\mathcal{V}_i = \{ e_{i,1},...,e_{i,n_i} \}$. $\mathcal{V}_i$ is a finite set of classes of an ontology.
- $\mathcal{E}_i = \{(e_{i,k}, e_{i,l}) \mid e_{i,k}, e_{i,l} \in \mathcal{V}_i \}$. $\mathcal{E}_i$ is a finite set of edges, where an edge encode the relationship between two classes of an ontology.

*Definition 3.2 (Ontology partition).* An ontology partition $p_{i,k}$ of an ontology $O_i$ is a sub-ontology denoted by $p_{i,k} = (\mathcal{V}_{i,k}, \mathcal{E}_{i,k})$ such as :

- $\mathcal{V}_{i,k} = \{e_{i,k,1},...,e_{i,k,m_k}\}$, $\mathcal{V}_{i,k} \subseteq \mathcal{V}_i$.
- $\mathcal{E}_{i,k} = \{(e_{i,k,x}, e_{i,k,y}) \mid (e_{i,k,x}, e_{i,k,y}) \in \mathcal{V}_{i,k} \exists (e_{i,k,x}, e_{i,k,y}) \in \mathcal{E}_{i,k} \}$, $\mathcal{E}_{i,k} \subseteq \mathcal{E}_i$.

*Definition 3.3 (Set of ontology partitions).* An ontology $O_i$ can be divided into a set of ontology partitions $\mathcal{P}_i = \{p_{i,1},...,p_{i,s_i}\}$.

- $\forall k \in [1..s_i], \mathcal{V}_{i,k} \neq \emptyset$
- $\bigcup_{k=1}^{s_i} \mathcal{V}_{i,k} = \mathcal{V}_i$
- $\forall$ k $\in [1..s_i]$, $\forall$ l $\in [1..s_i]$, $k \neq l$, $\mathcal{V}_{i,k} \cap \mathcal{V}_{i,l} = \emptyset$

*Definition 3.4 (Global matching).* The global matching $\mathcal{GM}_{ij}$ between two ontologies $O_i$ and $O_j$ consists in matching learning between the entities of $\mathcal{V}_i$ and $\mathcal{V}_j$. The global matching $\mathcal{GM}_{ij}$ requires a single machine learning model $\mathcal{G}ml_{ij}$ generated from one global training set $\mathcal{G}ts_{ij}$. The resulted mappings $\mathcal{M}_{ij}$ from the global matching $\mathcal{GM}_{ij}$ are denoted as $\mathcal{M}_{ij} = \{(e_{i,s}, e_{j,t}, r, c)\}$ where:

- $e_{i,s} \in \mathcal{V}_i$ and $e_{j,t} \in \mathcal{V}_j$
- the relationship $r \in \{\equiv, \subseteq, \supseteq\}$
- the confidence value $c \in [0, 1]$

*Definition 3.5 (Local matching).* The local matching $\mathcal{LM}_{ij}$ between two ontologies $O_i$ and $O_j$ is denoted by $\mathcal{LM}_{ij} = \{lm_{ij,1},...,lm_{ij,n}\}$. A single local matching task $lm_{ij,q}$ is computed by $\mathcal{V}_{i,k} \times \mathcal{V}_{i,l}$ respectively from $p_{i,k}$ and $p_{j,l}$, where $p_{i,k} \subseteq \mathcal{P}_i$ and $p_{j,l} \subseteq \mathcal{P}_j$.

## 4 BIOMEDICAL ONTOLOGIES PARTITIONING AND LOCAL MATCHING LEARNING

As depicted in Figure 1, the local matching architecture contains four modules: (i) input ontologies indexing and loading, (ii) input ontologies partitioning, (iii) local matching learning and (iv) output alignment generation and evaluation.

### 4.1 Input Ontologies Indexing and Loading

The first step of the ontology indexation and loading module is the pre-processing task. During this task, we pre-process the lexical annotations of the input ontologies. Thus, we apply the Porter stemming [28] as well as the stop word removal process over the extracted lexical annotations. The structural indexing is responsible for storing all the relationships between entities. The third step is responsible for loading the indexed data structures.

### 4.2 Input Ontologies Partitioning

The novel ontology partitioning approach follows mainly three stages: the ontology partitioning pre-processing, the partitioning algorithm and the identification of similar partition pairs.

*4.2.1 Ontology Partitioning Pre-processing.* We employ the hierarchical agglomerative clustering technique to divide an ontology into a set of partitions. This approach does not expect prior information regarding the number of required partitions. The hierarchical agglomerative clustering algorithm takes as input a list of pairwise structural similarity scores between all the entities of an input ontology. Based on Definition 4.1, we compute the structural relatedness between the entities of one ontology. Definition 4.1 is inspired by Wu and Palmer [31] similarity measure.

*Definition 4.1 (Relatedness between entities).* To compute the degree of relatedness between all the entities in one ontology, we measure their structural similarity. As depicted in Equation 1, for a

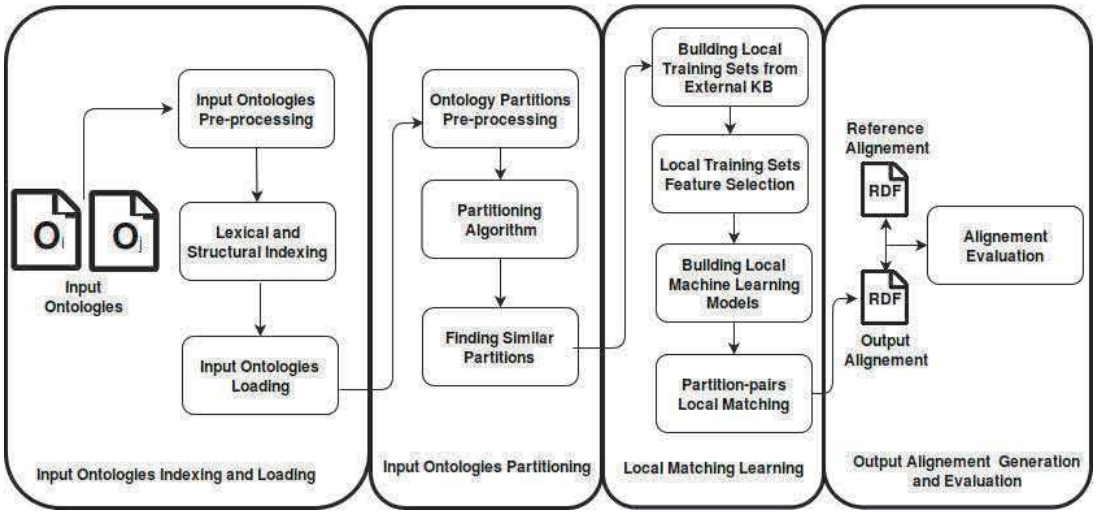


**Figure 1: Architecture Overview**

given two entities $e_{i,x}$ and $e_{i,y}$, *lca* is their lowest common ancestor. Dist($e_{i,x}$,*lca*) represents the shortest distance between $e_{i,x}$ and *lca* in terms of number of edges. Dist($e_{i,y}$,*lca*) denote the distance between $e_{i,y}$ and *lca*. Dist($r_i$,*lca*) is the distance between the root $r_i$ and *lca*.

$$StrcSim(e_{i,x}, e_{i,y}) = \frac{Dist(r_i, lca) \times 2}{Dist(e_{i,x}, lca) + Dist(e_{i,y}, lca) + Dist(r_i, lca) \times 2} \quad (1)$$

According to this structural similarity measure, when two entities are structurally close in one ontology, they are likely belonging to the same partition. Some ontologies do not contain any root element. Therefore, we search for all high-level entities of an ontology. We link these entities to a newly created root entity using a subsumption relationship. Consequently, we solve the issue of the lack of the root element.

**Algorithm 1** Ontology partitioning Algorithm

1: **Input** $\mathcal{G}_i$= ( $\mathcal{V}_i$, $\mathcal{E}_i$), *SplitSize* ▷ (1) Input
2: $\mathcal{D} \leftarrow \emptyset, \mathcal{P}_i \leftarrow \emptyset,\ \mathcal{L}v \leftarrow 0$ ▷ (2) Initialization
3: **for** $x \leftarrow 1$ *to* $|\mathcal{V}_i|$ **do**
4: $p_{i,x} \leftarrow (\{e_{i,x}\}, \emptyset)$
5: $\mathcal{P}_i \leftarrow \bigcup p_{i,x}$
6: **end for**
7: **while** ($\mathcal{L}v <= |\mathcal{V}_i|$) **do** ▷ (3) Dendrogram Construction
8: $(p_{i,l}, p_{i,k}) \leftarrow getMaxSimilarPart(\mathcal{P}_i)$
9: $\mathcal{P}_i \leftarrow \mathcal{P}_i \setminus p_{i,l} \setminus p_{i,k} + Merge(p_{i,l}, p_{i,k})$
10: $\mathcal{D} \leftarrow \mathcal{D} + < p_{i,l}, p_{i,k} >$
11: $\mathcal{L}v \leftarrow \mathcal{L}v + 1$
12: **end while**
13: $\mathcal{P}_{init} \leftarrow getInitCutPartitions(\mathcal{D})$ ▷ (4)Dendrogram Multi-cut
14: $\mathcal{P}_i \leftarrow \emptyset$
15: **for** *each* $p_{i,l}$ *of* $\mathcal{P}_{init}$ **do**
16: **if** ($|V_{i,l}| > SplitSize$) **then**
17: $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup getIterativeCutPartitions(\mathcal{D}, \mathcal{P}_{i,l})$
18: **else**
19: $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{p_{i,l}\}$
20: **end if**
21: **end for**
22: **Return** $\mathcal{P}_i$ ▷ Result

*4.2.2 Partitioning Algorithm.* Algorithm 1 describes the different steps required for partitioning an input ontology $O_i$. Algorithm 1 takes as input an ontology $O_i$ as the graph $\mathcal{G}_i$. The partitioning algorithm splits a single ontology into a set of partitions $\mathcal{P}_i$. As mentioned in Algorithm 1, the partitioning algorithm follows the following four steps:

**(1) Input** The different variables are initialized such as the *Dendrogram* $\mathcal{D}$ and the *Partitions* $\mathcal{P}_i$. The *Dendrogram* $\mathcal{D}$ represents the hierarchical relationship between the partitions of an ontology $O_i$.

**(2) Initialization** Each entity of $\mathcal{V}_i$ is initially considered as a single partition $p_{i,x}$ belonging to the set of partitions $\mathcal{P}_i$ (line 3 to 6). Therefore, $\mathcal{P}_i$ initially contains a set of partitions having a single entity for each one.

**(3) Dendrogram construction**The Dendrogram $\mathcal{D}$ of each input ontology is generated following the hierarchical agloromative approach [22] (line 7 to 12). This approach iteratively identify the two partitions $(p_{i,l}, p_{i,k})$ with the maximum structural similarity (line 8). The maximum structural similarity is computed using the relatedness formula (Definition 4.1) associated with the average linkage clustering technique. The identified partitions $p_{i,l}$ and $p_{i,k}$ are then removed from the set of initial partitions $\mathcal{P}_i$ (line 9). We also merge the partitions $p_{i,l}$ and $p_{i,k}$ into a new partition containing $(p_{i,l}, p_{i,k})$, wish is added to the set of initial partitions $\mathcal{P}_i$ (line 9). Every merged partitions are added to the Dendrogram hierarchy (line 10). The steps from line 7 to line 12 are repeated until building a dendrogram D with a number $\mathcal{L}v$ of levels equal to the number of entities of $\mathcal{V}_i$.

**(4) Dendrogram multi-cut** The generated dendrogram should be cut at a certain level to result in a set of non-isolated partitions $\mathcal{P}_i$. Partitions with only one entity are considered as isolated. These isolated partitions affect the matching accuracy result. For instance, the partitioning of FMA-SNOMED matching task of OAEI using Falcon [13] results in 3352 isolated partitions with an F-Measure 0.485 [27]. A single cut of a dendrogram can either result in a set of isolated partitions or a set of large partitions. To cope with this issue, we perform a multi-cut strategy of the resulted dendrogram (line 13 to 21). Therefore, the non-isolated partitions are derived based on two steps: an initial dendrogram cut (line 13) and a set of iterative cuts (line 14 to 21). The initial cut result in a set of partitions$\mathcal{P}_{init}$. This first cut is defined at a level of the dendrogram $\mathcal{D}$, which do not result in any isolated partitions. To perform this cut, we perform all the possible cuts over the dendrogram until finding the first cut returning non-isolated partitions. This initial cut result in a set of partitions $\mathcal{P}_i$ with no isolated ones. The initially returned partitions $\mathcal{P}_{init}$ may contain large ones. A large partition is identified through its size (SplitSize) in terms of the number of entities (line 16). The SplitSize is set as an input. We iteratively compare the size of the initial partitions $\mathcal{P}_{init}$ to the SplitSize (line 16). If an initial partition $p_{i,l}$ is large, we split this partition into smaller ones. These partitions are cutted based on the same strategy of Algorithm 1 (line 8). The identified partitions are added to the final partition set $\mathcal{P}_i$ (line 17). If the partition $p_{i,l}$ of $\mathcal{P}_{init}$ is smaller than the SplitSize, we add it directly to the final partition set $\mathcal{P}_i$ (line 19). The result of the multi-cut strategy is a set of partitions $\mathcal{P}_i$ for each input ontology $O_i$. In order to demonstrate the multi-cut strategy, we draw in Figure 2 a dendrogram example. This dendrogram is resulted by the hierarchical clustering of an ontology containing 13 entities (x axis). The initial cut is represented by the top dashed line. This cut result in two partitions $\mathcal{P}_{init}$ : the black partition (5 entities) and the partition with the grey colors (9 entities). The initial cut does not result in any isolated partitions. For this example, we adopt a SplitSize of 7 entities. Therefore, the partition with the grey colors should be cut in a level that results in no isolated partition while the maximum size of the resulted partition should be below the SplitSize. Consequently, we perform the second cut which results in the two partitions: the dark grey color (5 entities) and the light grey color (4 entities). Consequently, the multi-cut of the dendrogram result in three partitions depicted with the following three colors: Black (5 entities), light grey (4 entities) and dark grey (5 entities).

In the next step, we determine similar partition-pairs between two sets of ontology partitions.

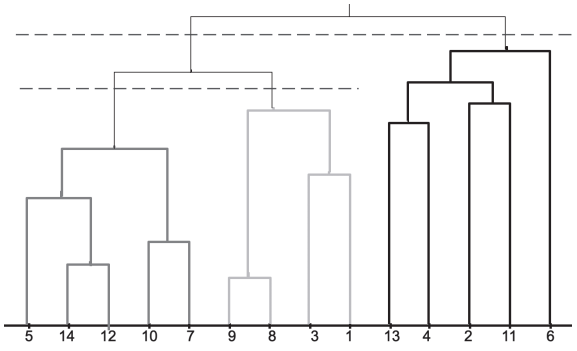


**Figure 2: Multi-cut example**

*4.2.3 Finding Similar Ontology Partitions.* Algorithm 2 takes as input the two sets of partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ of the input ontologies $O_i$ and $O_j$ with a set of anchors $\mathcal{A}_{ij}$ (line 1). Algorithm 2 results in the local matching $\mathcal{LM}_{ij}$. The anchors are denoted as: $\mathcal{A}_{ij}$ = {($e_{i,x}$,$e_{j,y}$) | $e_{i,x} \in \mathcal{V}_i$, $e_{j,y} \in \mathcal{V}_j$ }. The anchors $\mathcal{A}_{ij}$ are defined to identify the set of local matching $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. Since we are dealing with biomedical ontologies, anchors are extracted by cross-searching the input ontologies with the available external biomedical knowledge bases (KB) such as the Unified Medical Language System (UMLS) Metathesaurus [6], Medical Subject Headings (MeSH) [21], Uberon [23], and BioPortal [26]. For instance, UMLS integrates more than 160 biomedical ontologies. In our case, we cross-search the two input ontologies with the Uberon ontology to derive the set anchors $\mathcal{A}_{ij}$. We expand these anchors with a set of exact mappings between the input ontologies using a fast hash-based search method [11]. One generated partition $p_{i,k}$ of $\mathcal{P}_i$ can have multiple anchors with different partitions of $\mathcal{P}_j$. Also one partition $p_{j,k}$ of $\mathcal{P}_j$ can have multiple anchors with different partitions of $\mathcal{P}_i$. Therefore, we iteratively merge the partitions of $\mathcal{P}_i$ having anchors with more than one partition of $\mathcal{P}_j$ (line 3 to 6). We also merge the partitions of $\mathcal{P}_j$ having anchors with partitions of $\mathcal{P}_i$ (line 7 to 10). Hence, we guarantee that there is no anchors overlap between the partitions of $\mathcal{P}_i$ and $\mathcal{P}_j$. Consequently, a single partition of $\mathcal{P}_i$ has anchors with only a single partition of $\mathcal{P}_j$. Therefore, we are able to identify the local matching tasks $lm_{ij,q}$ of $\mathcal{LM}_{ij}$ between the partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ based on the anchors $\mathcal{A}_{ij}$ (line 11 to 17). This technique guarantees a good coverage ratio. Figure 3 shows the proposed approach for partitions merging. As depicted in the example of Figure 3 each ontology has 6 ontology partitions. Each partition contains a set of entities illustrated as nodes. The dashed lines represent the set of anchors between the ontology partitions of the two ontologies $O_i$ and $O_j$. We iteratively merge the ontology partitions of one ontology having anchors with one partition of the other ontology. For example, the ontology partitions P1 and P2 of the ontology $O_i$ are merged into one ontology partition. This merge is illustrated in Figure by a hashed circle. Thus, we are able to identify the set of local matchings $\mathcal{LM}_{ij}$ between the two ontologies. For instance, in Figure 3, we identify four local matchings: $\mathcal{LM}_{ij}$={$lm_{ij,1}$,$lm_{ij,2}$,$lm_{ij,3}$,$lm_{ij,4}$}.

**Algorithm 2** Finding Similar Partitions Algorithm

1: **Input** $\mathcal{P}_i$,$\mathcal{P}_j$,$\mathcal{A}_{ij}$ ▷ Input
2: $Q_i \leftarrow \emptyset, Q_j \leftarrow \emptyset, \mathcal{LM}_{ij} \leftarrow \emptyset$ ▷ Initialization
3: **for** each $p_{j,l}$ of $\mathrm{P}_j$ **do** ▷ $\mathcal{P}_i$ partitions merging
4: $\quad Q_i = \bigcup_{\forall (e_{ikx}, e_{jly}) \in A_{ij}} p_{i,k}$
5: $\quad \mathcal{P}_i \leftarrow \mathcal{P}_i \setminus Q_i \; + \; Merge(Q_i)$
6: **end for**
7: **for** each $p_{i,k}$ of $\mathrm{P}_i$ **do** ▷ $\mathcal{P}_j$ partitions merging
8: $\quad Q_j = \bigcup_{\forall (e_{ikx}, e_{jly}) \in \mathcal{A}_{ij}} p_{j,l}$
9: $\quad \mathcal{P}_j \leftarrow \mathcal{P}_j \setminus Q_j \; + \; Merge(Q_j)$
10: **end for**
11: **for** each $p_{i,k}$ of $\mathcal{P}_i$ **do** ▷ Finding similar partitions between $\mathcal{P}_i$ and $\mathcal{P}_j$
12: $\quad$ **for** each $p_{j,l}$ of $\mathcal{P}_j$ **do**
13: $\quad\quad$ **if** $(\exists (e_{ikx}, e_{jly}) \in \mathcal{A}_{ij})$ **then**
14: $\quad\quad\quad \mathcal{LM}_{ij} \leftarrow \mathcal{LM}_{ij} \bigcup \{(p_{i,k}, p_{j,l})\}$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**
18: **Return** $\mathcal{LM}_{ij}$ ▷ Result

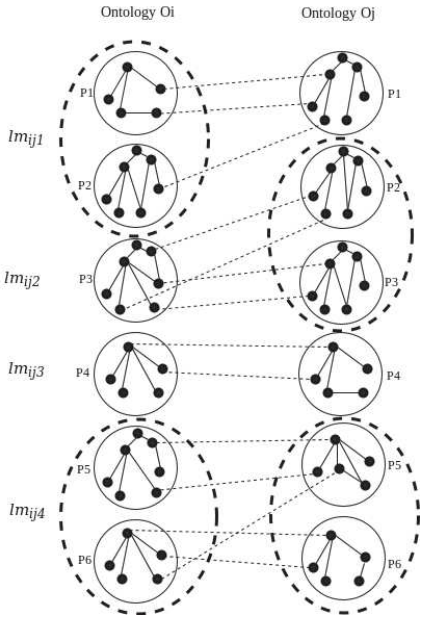


**Figure 3: Partitions merging**

## 4.3 Local Matching Learning

The high complexity of the large biomedical ontologies decreases the matching accuracy. No single similarity measure can effectively treat all the syntactic heterogeneity aspects of a matching task.

Therefore, for each local matching task, we construct its specific machine learning model. The training set of every local learning model is not based on any reference alignments. In the following, we detail the required steps for building the local training sets.

*4.3.1 Building Local Training Sets from External Knowledge Bases.* We automatically construct a supervised training set $ts_{ij,q}$ for each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. These training sets serve as the input for each local machine learning model. Every local training set $ts_{ij,q}$contains a set of features associated with a class attribute denoted by: $ts_{ij,q} = \{f_{i1},..., f_{niq}, c_{qi}\}$. Since we are dealing with a supervised binary classification task, the class label $c_{qi} \in \{0,1\}$. We employ the-state-of-the art syntactic similarity measures[2] as features. Table 1 represents an example of a local matching training set $ts_{ij,q}$ for the ontologies $O_i$ and $O_j$. The labeled data of the training set is usually hard to acquire. The existing works retrieve labeled data either from the reference alignment or by creating it manually. However, the reference alignment commonly does not exist. We automatically generate the local training sets without the manual involvement of the user or any gold standard. We derive the positive mappings samples of $c_{qi}$ by cross-searching the entities of a local matching with the existing biomedical knowledge bases like Uberon. For a given local matching $lm_{ij,q}$, we denote the extracted positive samples by $\mathcal{PS}_{ij,q} = \{(e_{i,q,x}, e_{j,q,y})\}$. We deduce the negative samples from the retrieved positive samples. The negative samples $\mathcal{NS}_{ij,q}$ are determined by computing the difference between the extracted positive sample $\mathcal{PS}_{ijq}$ and the cartesian product of the entities of a local matching $lm_{ij,q}$. Thus, $\mathcal{NS}_{ijq} = (\mathcal{V}_{i,q} \times V_{j,q}) \setminus PS_{ijq}$. Therefore, we result in $m$ negative samples $\mathcal{NS}_{ijq}$, where $m = n(n-1)$ and $n = |\mathcal{PS}_{ijq}|$. Consequently, the training set is not balanced since the number of the negative samples $m$ is higher than the number of positive samples $n$. We undersample the training set by removing all the negative samples having at least one feature with a similarity score equal to zero. Then, we randomly sample the extracted negative samples. Therefore, the training set became balanced ($n$=$m$). The output of this step is a local training set $ts_{ij,q}$ for each local matching task $lm_{ij,q}$. This local training set captures the characteristics of the local matching context.

| Local entities pairs | Feature 1 | ... | Feature n | Class |
|---|---|---|---|---|
| $(e_{i,q,1}, e_{j,q,1})$ | 0.61 | ... | 0.75 | 1 |
| $(e_{i,q,2}, e_{j,q,2})$ | 0.45 | ... | 0.5 | 0 |
| ... | ... | ... | ... | ... |
| $(e_{i,q,x}, e_{j,q,y})$ | ... | ... | ... | ... |

**Table 1: An example of a local matching training set.**

*4.3.2 Feature Selection.* We apply the wrapper feature selection [17] method over the resulted local training sets. This technique selects the subset of the most effective and suitable features for each local training set. Therefore, each local matching task has its specific similarity measures. This method overcomes the issue of the manual choice of similarity measures.

*4.3.3 Building Local Machine Learning Models.* We build a local machine learning model for each local matching task. The purpose of the learning models is to determine the thresholds of the selected similarity measure. Moreover, every learning model defines a weight for each similarity measure. Therefore, every single local matching task has its specific tuning.

*4.3.4 Local Matching of Similar Partition-pairs .* The entities of each local matching task are classified using their specific machine learning model. This local learning model aligns the input entities based on the adequate matching parameters.

## 4.4 Alignments Generation and Evaluation

The generated correspondences for every local matching task $lm_{ij,q}$ are unified to generate the final alignment file for the whole ontology matching task. The alignment file is compared to the reference alignment to evaluate the overall result accuracy.

# 5 EVALUATION

We evaluated the proposed approach according to three different experiments. In Section 5.2, we compare the defined partitioning approach to the existing state-of-the-art works. In Section 5.3, we asses the novel local matching approach while using different machine learning algorithms. In Section 5.4, we compare the results of the global matching to the novel local matching learning approach. Since we are dealing with biomedical ontologies, we are based on the datasets provided by the OAEI of 2017, in particular, the biomedical ontologies matching tracks: Anatomy and LargeBio. All experiments have been implemented in Java using a MacOs operating system with 2.8 GHz Intel I7-7700HQ and 16 GB of internal memory. The runnable Java jar file of the matching system is available upon (request[3]).

## 5.1 Partitioning Approach Evaluation

We evaluated the proposed partitioning approach according to the current partitioning strategies SeeCOnt [2], Falcon [13], Alsayed et al. [3] and Ernesto et al. [16]. All these approaches are evaluated using the biomedical anatomy track of OAEI. Therefore, in Table 2, using the same dataset, our partitioning strategy outperforms the existing state of the art approaches. We achieved an F-Measure of 89% with a coverage ratio of 98.3%. The recall value is significantly higher than most of the existing works due to the achieved coverage ratio. The precision value is lower than Seecont [2], Falcon [13] and Alsayed et al. [3] due to the additional number of wrong alignments discovered compared to the later approaches. The good coverage ratio is achieved because of the performed partitions merging strategy of Algorithm 2. The number of the resulted partitions for the two ontologies of the anatomy track is 57 due to the proposed merging strategy. SeeCOnt did not mention the number of generated partitions for the anatomy dataset. Moreover, the other existing approaches did not define (n/d) the achieved coverage ratio. The partitioning coverage and its ratio are defined respectively in Definition 5.1 and 5.2.

[2]https://git.io/fNvqt

[3]https://goo.gl/FjZGh6

| Approach | F-Measure | Precision | Recall | Partitions number | Coverage Ratio | Run time (mn) |
|---|---|---|---|---|---|---|
| Proposed approach | 0.896 | 0.915 | 0.877 | 57/57 | 98.3 | 8.13 |
| SeeCOnt [2] | 0.863 | 0.951 | 0.789 | n/d | n/d | n/d |
| Ernesto et al. [16] | 0.850 | 0.880 | 0.820 | 5/10 | n/d | 42 |
| Falcon [13] | 0.730 | 0.964 | 0.591 | 139/119 | n/d | 10 |
| Alsayed et al. [3] | 0.753 | 0.975 | 0.613 | 84/80 | n/d | 0.98 |

**Table 2: Anatomy track partitioning results**

*Definition 5.1 (Partitioning coverage).* An alignment $\mathcal{M}_{ij}$ is the set of correspondences between two ontologies $O_i$ and $O_j$. $\mathcal{M}_{ij}$ can be either retrieved from the reference alignment. We state that a single mapping of $\mathcal{M}_{ij}$ is covered by the set of local matching tasks $\mathcal{LM}_{ij}$, if the later mapping is discovered by at least one local matching task $lm_{ij,q}$. The partitioning coverage ratio is defined in the following Definition 5.2.

*Definition 5.2 (Partitioning coverage ratio).* The partitioning coverage ratio of a set of Local matching $\mathcal{LM}_{ij}$ defines the percentage of mappings that cannot be discovered after performing the partitioning process. Consequently, a low coverage ratio leads to a low matching accuracy. $Cg(lm_{ij,q}, \mathcal{M}_{ij})$ denotes the set of mappings $\mathcal{M}_{ij}$ covered by the $lm_{ij,q}$. In the following Equation 2, we present the partitioning coverage ratio between the set of local matchings and the reference mappings $\mathcal{M}_{ij}$.

$$CgRatio(\mathcal{LM}_{ij},\ \mathcal{M}_{ij}) = \frac{|Cg(lm_{ij,q}, \mathcal{M}_{ij}))|}{|\mathcal{M}_{ij}|} \tag{2}$$

## 5.2 Matching Learning Results

In Figure 4, we compared the accuracy of local matching while employing different machine learning algorithms for building the set of local learning models. Since we are performing a supervised classification strategy, we draw the results of the top performing machine learning algorithm for each classification algorithm type. Therefore, we report the achieved results by SVM (function), Naive Bayes (bayes), JRip (decision rules) and C4.5 (decision tree). In Figure 4, we depict the resulted accuracy based on the OAEI 2017 biomedical matching tasks: the Anatomy dataset and the Largebio dataset fragments of FMA-NCI, FMA-SNOMED and SNOMED-NCI. SVM achieved a better accuracy for the local matching learning than the other algorithms. For every local matching task $lm_{ij,q}$, we generated a local training set $ts_{ij,q}$ by cross-searching the local entities with UBERON as an external knowledge base. We are based on the extracted positive samples $\mathcal{PS}_{ijq}$ from UBERON to automatically infer the negative samples $\mathcal{NS}_{ijq}$. We mention that different knowledge bases can be employed to enrich the positive samples of the local training sets. We are based on the wrapper feature selection method to identify the adequate similarity measures for each local matching task $lm_{ij,q}$. Therefore, each local matching task is aligned through a local machine learning model, which defines the local matching tuning.

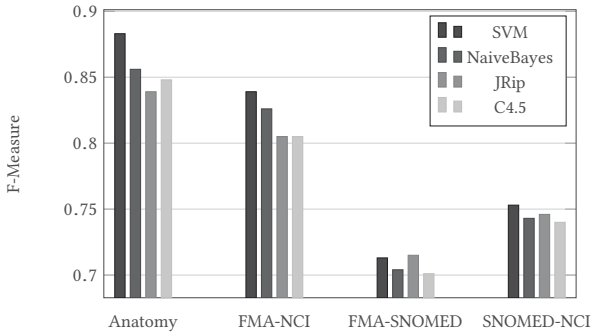


**Figure 4: Comparing different machine learning algorithms for local matching**

## 5.3 Comparing Local Matching to Global Matching

As shown in Figure 5, we conducted a set of experiments in order to compare the proposed local matching learning to the global matching. The global matching $\mathcal{GM}_{ij}$ is the alignment between two input ontologies using a single global machine learning model. This model is generated using one global training set. We automatically build the training set by cross-searching the two input ontologies with UBERON as an external biomedical knowledge base (KB). Then we perform the wrapper feature selection method to select the adequate similarity measures The local matching models are applied locally over the set of similar partition pairs. Global matching and local matching are evaluated using the same reference alignments provided by OAEI. In order to compare our local matching approach to the existing ontology matching systems, we employ the 2017 version of AgreementMakerLight (AML) [11] as a baseline. AML has been consistently the top OAEI matching system for biomedical tracks since the OAEI of 2012. For a fair comparison, we turn on the element level matchers of AML since the local matching learning is based on the element level. Moreover, we compared our approach to the 2017 version of LogMap [1], which has been one of the top matching systems. As depicted in Figure 5, the local matching significantly outperforms the global matching as well as AML. We achieved comparable results to LogMap. We mention that our approach employs only syntactic similarity measures as features for the local matching learning. AML manually defines the employed similarity measure and their associated thresholds. However, our proposed approach is completely automated for local matching of biomedical ontologies. For instance, AML chooses a global syntactic threshold of 0.6 for all the matching task. Nonetheless, our proposed approach derives automatically the adequate threshold value for every employed syntactic similarity measure

in every local matching task context. Thus, we take advantage of the partitioning process to maximize the matching accuracy gain. The automatically generated local matching tuning takes into consideration the matching context in order to increases the overall obtained result.

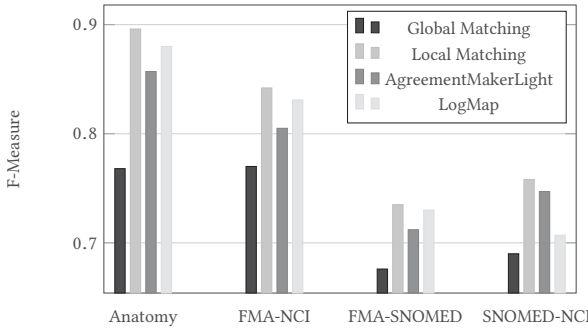


**Figure 5: Comparing local matching to global matching**

## 6 CONCLUSION AND FUTURE WORK

Ontology matching is the key interoperability enabler for the Semantic Web. We have proposed a novel approach for ontology partitioning to deal with large ontology matching tasks. We target not only the reduction of search space but especially the increase of the matching quality using the local matching learning approach. We break down an ontology matching task into a set of local matching tasks. We have proposed a novel approach that performs an automated local matching learning process. This local matching learning combines ontology partitioning with machine learning techniques. The proposed local machine learning model automatically defines the matching tuning of each sub-matching task. Every local machine learning model receives as an input a local training set without employing any gold standard or user involvement. The existing approaches generate a global machine learning model by using the reference alignments. However, the reference alignment usually does not exist and the manual approach is time consuming. We automatically generate the local training sets by exploring the external biomedical knowledge bases. Therefore, the proposed local matching learning model defines the adequate matching tuning parameters for every sub-matching task context. We have shown that the local matching of partitions-pairs outperforms the global matching approach. However, our local matching learning approach is limited to the biomedical domain. In future work, we tend to perform the local matching learning for different domains. We also intend to incorporate structure level features into the local matching learning strategy.

## REFERENCES


[1] Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, et al. 2017. Results of the ontology alignment evaluation initiative 2017. In *OM 2017-12th ISWC workshop on ontology matching*.

[2] Alsayed Algergawy, Samira Babalou, Mohammad J Kargar, and S Hashem Davarpanah. 2015. Seecont: A new seeding-based clustering approach for ontology matching. In *East European Conference on Advances in Databases and Information Systems*. Springer.

[3] Alsayed Algergawy, Sabine Massmann, and Erhard Rahm. 2011. A clustering-based approach for large-scale ontology matching. In *East European Conference on Advances in Databases and Information Systems*. Springer.

[4] Mathieu Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. 2007. Ontology modularization for knowledge selection: Experiments and evaluations. In *International Conference on Database and Expert Systems Applications*. Springer.

[5] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. 2005. Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. Acm.

[6] Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32.

[7] Agnese Chiatti, Zlatan Dragisic, Tania Cerquitelli, and Patrick Lambrix. 2015. Reducing the search space in ontology alignment using clustering techniques and topic identification. In *Proceedings of the 8th International Conference on Knowledge Capture*. ACM, 21.

[8] Kai Eckert, Christian Meilicke, and Heiner Stuckenschmidt. 2009. Improving ontology matching using meta-level learning. In *European Semantic Web Conference*.

[9] Jérôme Euzenat and Pavel Shvaiko. 2007. Ontology matching, vol. 1. Heidelberg, Germany: Springer-Verlag.

[10] Daniel Faria, Catia Pesquita, Isabela Mott, Catarina Martins, Francisco M Couto, and Isabel F Cruz. 2018. Tackling the challenges of matching biomedical ontologies. *Journal of biomedical semantics* 9, 1.

[11] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. 2013. The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer.

[12] Anika Grob, Michael Hartung, Toralf Kirsten, and Erhard Rahm. 2012. GOMMA results for OAEI 2012. In *Proceedings of the 7th International Conference on Ontology Matching-Volume 946*. CEUR-WS. org.

[13] Wei Hu, Yuzhong Qu, and Gong Cheng. 2008. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering* 67, 1.

[14] Wei Hu, Yuzhong Qu, and Gong Cheng. 2008. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering* 67, 1 (2008).

[15] Ryutaro Ichise. 2008. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*. IEEE.

[16] Ernesto Jiménez-Ruiz, Asan Agibetov, Matthias Samwald, and Valerie Cross. 2018. We Divide, You Conquer: From Large-scale Ontology Alignment to Manageable Subtasks. (2018).

[17] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2.

[18] Amir Laadhar, Faiza Ghozzi, Ryutaro Ichise, Imen Megdiche-Bousarsar, Franck Ravat, and Olivier Teste. [n. d.]. Automated Partitioning and Matching Tuning of Large Biomedical Ontologies (poster).

[19] Amir Laadhar, Faiza Ghozzi, Imen Megdiche-Bousarsar, Franck Ravat, Olivier Teste, and Faiez Gargouri. 2017. POMap: An Effective Pairwise Ontology Matching System (short paper). In *International Conference on Knowledge Engineering and Ontology Development (KEOD)*. INSTICC, 1–8.

[20] Amir Laadhar, Faiza Ghozzi, Imen Megdiche-Bousarsar, Franck Ravat, Olivier Teste, and Faiez Gargouri. 2017. POMap results for OAEI 2017 (regular paper). In *Ontology Matching co-located with the 16th International Semantic Web Conference (OM@ISWC'17), Vienna, Austria, 21/10/17-25/10/17*. CEUR Workshop Proceedings.

[21] Carolyn E Lipscomb. 2000. Medical subject headings (MeSH). *Bulletin of the Medical Library Association* 88, 3.

[22] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

[23] Christopher J Mungall, Carlo Torniai, Georgios V Gkoutos, Suzanna E Lewis, and Melissa A Haendel. 2012. Uberon, an integrative multi-species anatomy ontology. *Genome biology* 13, 1.

[24] Azadeh Haratian Nezhadi, Bita Shadgar, and Alireza Osareh. 2011. Ontology alignment using machine learning techniques. *International Journal of Computer Science & Information Technology* 3, 2.

[25] DuyHoa Ngo and Zohra Bellahsene. 2016. Overview of YAM++(not) Yet Another Matcher for ontology alignment task. *Web Semantics: Science, Services and Agents on the World Wide Web* 41.

[26] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. 2009. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research* 37, suppl_2.

[27] Sunny Pereira, Valerie Cross, and Ernesto Jiménez-Ruiz. 2017. On Partitioning for Ontology Alignment. In *International Semantic Web Conference*.

[28] Martin F Porter. 2001. Snowball: A language for stemming algorithms.

[29] Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Michelle Cheatham, and Oktie Hassanzadeh. [n. d.]. OM-2017: Proceedings of the Twelfth International Workshop on Ontology Matching. In *OM 2017*.

[30] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra. 2009. *Modular ontologies: concepts, theories and techniques for knowledge modularization*. Vol. 5445. Springer.

[31] Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

[32] Xingsi Xue and Jeng-Shyang Pan. 2017. A segment-based approach for large-scale ontology matching. *Knowledge and Information Systems* 52, 2.