THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *Date de défense (27/09/2019)* **par :**

Amir LAADHAR

Local Matching Learning of Large Scale Biomedical Ontologies

JURY

Sonia AYACHI GHANNOUCHI	Maitre de conférences habilité, Université de Sousse	Rapporteur
Ladjel BELLATRECHE	Professeur, ENSMAA-Poitiers	Rapporteur
Bernard ESPINASSE	Professeur, Université Aix-Marseille	Examinateur
Faiez GARGOURI	Professeur, Université de Sfax	Directeur
Faiza GHOZZI	Maitre Assistant, Université de Sfax	Co-directeur
Clement JONQUET	Maitre de Conférences habilité, Université de Montpellier	Examinateur
Imen MEGDICHE	Maitre de Conférences, Ecole d'ingénieurs ISIS	Co-directeur
Franck RAVAT	Professeur, Université Toulouse 1	Invité
Olivier TESTE	Professeur, Université Toulouse 2	Directeur

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :
Olivier TESTE et *Faiez GARGOURI*

Rapporteurs :
Sonia AYACHI GHANNOUCHI et *Ladjel BELLATRECHE*

Acknowledgments

First of all, I would like to thank all those who contributed to the preparation and writing of my thesis. Obviously, it is difficult to thank everyone. Thanks to the help of many people, I was able to carry out this thesis to its end.

In the first place, I would like to express my gratitude to all my family, thanks to my father and mother who have been the luminous stars for me all along the road and who have always believed in me. I would like to thank my sister Ahlem and all my family for their sincere love during these years.

My thanks go also to the reviewers, Sonia Ayachi Ghannouchi and Ladjel Bellatreche, to whom I associate the members Bernard Espinasse and Clement Jonquet for honoring me with taking time to read in detail my work and whose kindness, interest and advice to continue and improve my research are invaluable.

I would like to thank my thesis directors, Faiez Gargouri and Olivier Teste for having accepted to supervise me during the preparation of my thesis. In addition to their scientific support and the human side, they have always been there to support me and advise me throughout the development of my work especially during difficult times.

A special thanks goes also to my supervisors Faiza Ghozzi, Imen Megdiche and Franck Ravat for always being there whenever I had questions, and for their patience and immense knowledge. A special thanks to all my friends. It is almost impossible for me to imagine the last three years without them, who gave me moral support during the years of this thesis.

Thanks, finally, to all the members of the SIG team. I deeply appreciate the IRIT lab every single person, for their involvement in my personal well-being.

Abstract

Although a considerable body of research work has addressed the problem of ontology matching, few studies have tackled the large ontologies used in the biomedical domain. We introduce a fully automated local matching learning approach that breaks down a large ontology matching task into a set of independent local sub-matching tasks. This approach integrates a novel partitioning algorithm as well as a set of matching learning techniques. The partitioning method is based on hierarchical clustering and does not generate isolated partitions. The matching learning approach employs different techniques: (i) local matching tasks are independently and automatically aligned using their local classifiers, which are based on local training sets built from element level and structure level features, (ii) resampling techniques are used to balance each local training set, and (iii) feature selection techniques are used to automatically select the appropriate tuning parameters for each local matching context. Our local matching learning approach generates a set of combined alignments from each local matching task, and experiments show that a multiple local classifier approach outperforms conventional, state-of-the-art approaches: these use a single classifier for the whole ontology matching task. In addition, focusing on context-aware local training sets based on local feature selection and resampling techniques significantly enhances the obtained results.

Résumé

Les larges ontologies biomédicales décrivent généralement le même domaine d'intérêt, mais en utilisant des modèles de modélisation et des vocabulaires différents. Aligner ces ontologies qui sont complexes et hétérogènes est une tâche fastidieuse. Les systèmes de matching doivent fournir des résultats de haute qualité en tenant compte de la grande taille de ces ressources. Les systèmes de matching d'ontologies doivent résoudre deux problèmes: (i) intégrer la grande taille d'ontologies, (ii) automatiser le processus d'alignement.

Le matching d'ontologies est une tâche difficile en raison de la large taille des ontologies. Les systèmes de matching d'ontologies combinent différents types de matcher pour résoudre ces problèmes. Les principaux problèmes de l'alignement de larges ontologies biomédicales sont: l'hétérogénéité conceptuelle, l'espace de recherche élevé et la qualité réduite des alignements résultants.

Les systèmes d'alignement d'ontologies combinent différents matchers afin de réduire l'hétérogénéité. Cette combinaison devrait définir le choix des matchers à combiner et le poids . Différents matchers traitent différents types d'hétérogénéité. Par conséquent, le paramétrage d'un matcher devrait être automatisé par les systèmes d'alignement d'ontologies afin d'obtenir une bonne qualité de correspondance.

Nous avons proposé une approche appele "local matching learning" pour faire face à la fois à la grande taille des ontologies et au problème de l'automatisation. Nous divisons un gros problème d'alignement en un ensemble de problèmes d'alignement locaux plus petits. Chaque problème d'alignement local est indépendamment aligné par une approche d'apprentissage automatique. Nous réduisons l'énorme espace de recherche en un ensemble de taches de recherche de corresondances locales plus petites. Nous pouvons aligner efficacement chaque tache de recherche de corresondances locale pour obtenir une meilleure qualité de correspondance.

Notre approche de partitionnement se base sur une nouvelle stratégie à découpes multiples générant des partitions non volumineuses et non isolées. Par conséquence, nous pouvons surmonter le problème de l'hétérogénéité conceptuelle. Le nouvel algorithme de partitionnement est basé sur le clustering hiérarchique par agglomération (CHA). Cette approche génère un ensemble de tâches de correspondance locale avec un taux de couverture suffisant avec aucune partition isolée.

Chaque tâche d'alignement local est automatiquement alignée en se basant sur les techniques d'apprentissage automatique. Un classificateur local aligne une seule tâche d'alignement local. Les classificateurs locaux sont basés sur des features élémentaires et structurelles. L'attribut class de chaque set de donne d'apprentissage training set est automatiquement étiqueté à l'aide d'une base de connaissances externe. Nous avons appliqué une technique de sélection de features pour chaque classificateur local afin de sélectionner les matchers appropriés pour chaque tâche d'alignement local. Cette approche réduit la complexité d'alignement et augmente la précision globale par rapport aux méthodes d'apprentissage traditionnelles.

Nous avons prouvé que l'approche de partitionnement est meilleure que les approches actuelles en terme de précision, de taux de couverture et d'absence de partitions isolées. Nous avons évalué l'approche d'apprentissage d'alignement local à l'aide de diverses expériences basées sur des jeux de données d'OAEI 2018. Nous avons déduit qu'il est avantageux de diviser une grande tâche d'alignement d'ontologies en un ensemble de tâches d'alignement locaux. L'espace de recherche est réduit, ce qui réduit le nombre de faux négatifs et de faux positifs. L'application de techniques de sélection de caractéristiques à chaque classificateur local augmente la valeur de rappel pour chaque tâche d'alignement local.

Mots clés : Alignement d'ontologies, Partitionnement d'ontologies, Web sémantique, Apprentissage Automatique

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Research Context

The semantic web is an extension of the World Wide Web through standards by the World Wide Web Consortium (W3C) [2]. Therefore, humans and machines will understand and explore the available data. In the Semantic Web, an idea of Tim Berners-Lee, the director of the W3C, documents are annotated with additional meta data, such that software agents can interpret the information to capture the semantics. This is not only limited to the use within the WWW but can also be used within several applications. The most simple way to add meta data to information is the use of RDF, which describes relations between different pieces of information. Meta information can also be used not only to better describe information sources but to better understand the questions of the user.

An ontology is a data model that represents a set of entities belonging to a specific domain and the set of relationships between the entities of this domain. Ontologies have many applications. A concept (also called class) is the main component (concerning the information content) of an ontology. It may be described by several attributes, which are concrete data fields. It is also possible to store information directly as instances of concepts. In addition, most ontologies provide extra information on the entities, e.g. data types or comments.

The ontology in Figure 1.1 displays a few pieces of information of the domain “organization of a university” stored in an ontology. “Professor”, “address”, “chair” and “lectures” are concepts or classes. The pale blue fields like “name” or “surname” are called attributes or data type properties. The diamonds represent relations between different concepts; a relation might be an object property or (not shown in this example) a subclass property. Instances are concrete values of the attributes and are displayed in italic strings. An instance called also individual is belonging to the concept “professor” is “Stefan Conrad”, where “Stefan” is the value of the attribute “name” and Conrad the value of “surname”. This instance has a relation to the instance “conrad@cs.uni-duesseldorf.de.

Ontology contains information about concepts, relationships, attributes and individuals. Concepts are abstract groups or types of objects. Relationships describe relations between objects, they include specialization (is-a) relation and meronymy (part-of) relation. Attributes describe concepts they belong to. Individuals are instantiations of concepts, usually representing real-world objects. They contain values (attribute instances) and are connected with relationship instances. The representation process of an ontology aims to make those logical definitions and axioms to be formal, and machine-accessible. There exist many kinds of languages can be used in that task such as OWL, RDF.. different represented languages lead to different syntax in the formal ontologies. Ontology languages are formal languages used to encode ontology specifications. The Web Ontology Language (OWL) [69] is the recent standard for ontology specification in the domain of the semantic web. In OWL terminology, concepts are called classes. A class may be specified as a subclass of another class, thus implementing the specialization relationship. Properties are used to define the content of classes. Properties fall into two categories; object properties represent general relations between two classes, while datatype properties represent

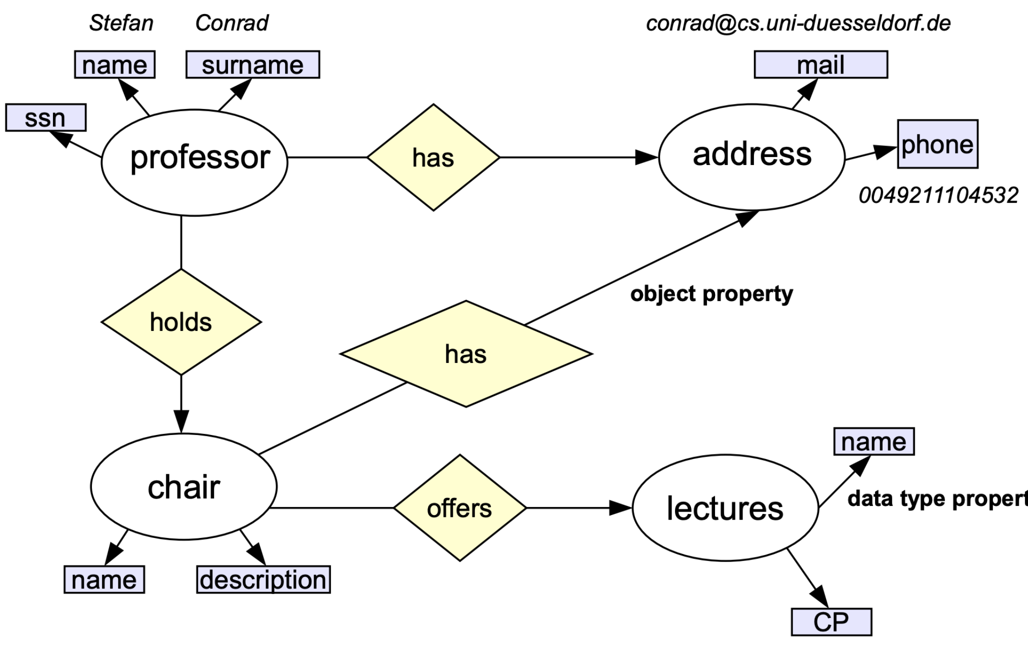


Figure 1.1: Ontology example

attributes. Individuals are still called individuals. OWL builds on other standards, XML Schema datatypes are used, and the RDF/XML syntax [66] is used to exchange OWL ontologies.

The main role of ontology is to describe domain knowledge in a generic and explicit way. For instance, El Hadj Amor et al. [6] presented a new ontology based on a real business process to create semantic relationships between all terms. An ontology provides also an agreed understanding of a domain. The conceptualization process provides a simplified viewpoint of users to the reality (the domain knowledge). However, different users may have different knowledge acquisitions, different backgrounds and understanding of the way this knowledge has to be conceptualized. This leads to different conceptualizations comprising different objects, entities and relations among them. Further, the formalization process describes all the concepts defined in the previous step as an explicit specification. It means that developers explain all entities and their relations as given in the conceptualization phase, by using some specific formal language (e.g. Description Logics). But, different languages provide different abilities to represent logical rules or axioms and different developers may assign different names for constant or predicates. Therefore, at this step even the same conceptualization maybe formalized with different specification.

Life sciences ontologies like biomedicines and biology are produced and managed by ontology developers and research community [7]. The integration and the analysis of these datasets related to a single topic of interest such as the correlation between genotype and phenotype is essential in the knowledge discovery process. Consequently, researchers are publishing these datasets on the internet to make them available to the semantic web community and more specifically to the biomedicines specialists [7, 91]. However, the integration of the biomedical datasets raises many challenges in terms of their integration, management, and representation. These challenges are addressed by the semantic web community in order to resolve the issues related to the large scale of the available biomedical ontologies published on the web. In figure 1.2, we depict the linking open data cloud diagram[1] of March 2019. This web page is a representation of the LOD cloud diagram. Datasets that have been published in the Linked Data format. The dataset currently contains 1,239 datasets with 16,147 links. The pink circles represent the life sciences datasets, which represents one of the biggest sub clouds of the LOD cloud diagram.

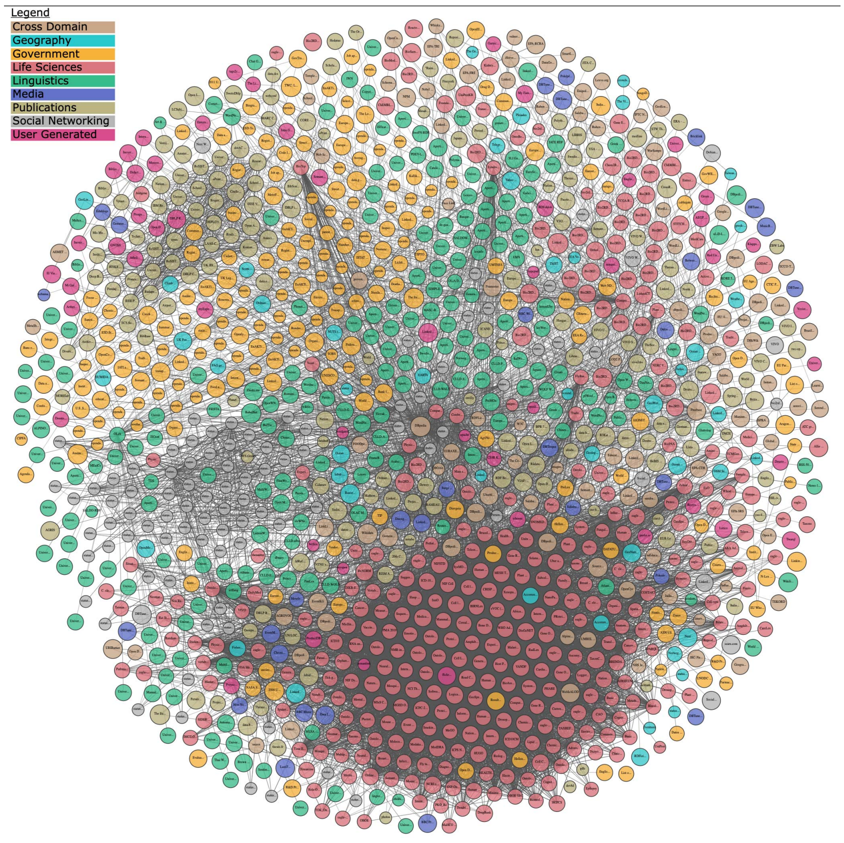


Figure 1.2: Linking open data cloud diagram of March 2019

An ontology can be used in many different application areas to describe and store knowledge. In general, there are many different ontologies describing the same domain. In most cases, the ontologies are not totally equal, because the used vocabulary differs and the coverage of the domain is varying. If two or more ontologies need to be compared correspondences have to be found despite the existing heterogeneity. Ontologies are highly heterogeneous [35]. According to the classification of Jérôme Euzenat and Pavel Shvaiko [35], ontology heterogeneity can be classified into four levels:

- Syntactic: At this level, all forms of heterogeneity depend on the choice of the representation format. This heterogeneity is caused by the use of different models or vocabularies such as XML, RDF and OWL [81].

- Terminological: At this level, all forms of heterogeneity are related to the process of

[1]https://lod-cloud.net/

naming entities that occur in an ontology [35].

- Conceptual: At this level, all forms of heterogeneity have come from the differences in the content of an ontology [35].

- Semiotic Pragmatic: At this level, all the discrepancies are related to the fact that different individual/communities may interpret the same different ways in different context [81].

The heterogeneity at the Syntactic level can be handled by using a transformation tool, which converts ontology from one represented language to another [81]. The heterogeneity at the Semiotic and Pragmatic level is very difficult because they strongly depend on understanding the context of using ontology. Therefore, most of the current ontology matchers focus only on solving the problem of mismatches between entities at the Terminology and Conceptual levels. Dealing with terminological heterogeneity, ontology matching systems employ different matcher based on similarity measures. A similarity measure is a function $f : \mathcal{V}_i \to \mathcal{V}_j$ [0,1] where $\mathcal{V}_i$ is the set of $\mathcal{O}_i$ entities and $\mathcal{V}_j$ is the set of $\mathcal{O}_j$ entities. For each pair of entities (e_i, e_j), a similarity measure computes a real number, generally between 0 and 1, expressing the similarity between the two entities.

Ontology alignment is a process of discovering correspondences (or mappings) between semantically related entities of different ontologies. An ontology matching systems define a confidence value between the discovered correspondences. Usually this confidence value is denoted as n, where $n \in [0..1]$ Due to the high heterogeneity of ontologies, the same concept described in different ontologies may have different representations (e.g., labels, properties or relations with other concepts) [35]. Alignment algorithms primarily use the equivalence relation (=), meaning that the matched objects are the same or are equivalent. Using OWL vocabulary, it is possible to take advantage of the ”owl:equivalentClass”, ”owl:disjointWith” or ”rdfs:subClassOf” relations in order to infer correspondences between classes of two ontologies. These relations correspond to set-theoretic relations between classes: equivalence (=); disjointness ($\perp$) and less general ($\leq$) [35].

Figure 1.3 is extracted from [47]. This figure shows a small ontology network with two ontologies (concepts are represented by nodes and the is-a structures are represented by directed edges) and an alignment (represented by dashed edges) [47]. The alignment consists of 10 equivalence mappings. One of these mappings represents the fact that the concept of bone in the first ontology is equivalent to the concept of bone in the second ontology [47].

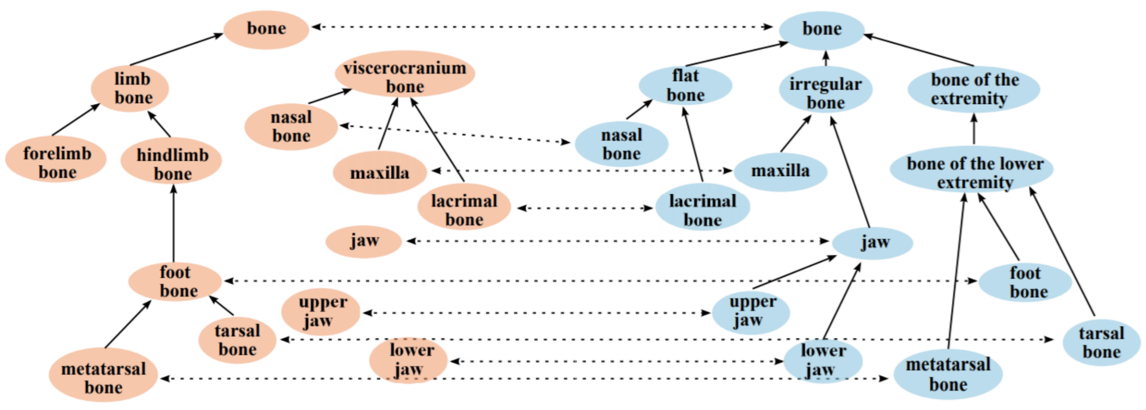


Figure 1.3: Ontology matching example

Ontology matching system is a software system that performs the matching process [73]. The degree of automation is an important characteristic of the matching system.

- Interactive matching is done by a human, who is given hints by the matching system.
- Semiautomatic matching systems present a set of mapping propositions to the user. The user approves them and may include additional mappings manually, this is repeated until the matching is completed.
- Fully automatic matching is done solely by the matching system; it is the only option if user input is not available.

The present research focuses mainly to fully automatic matching. In most scenarios, ontology matching is expected to be done without the presence of a qualified user, therefore fully automatic matching is required.

Let A be an alignment produced by a given ontology matching system and R the reference alignment [8]. We can compute the accuracy of the matching system based on Precision, Recall and F-measure as follows:

$$Precision = \frac{|A \cap R|}{|A|}$$

$$Recall = \frac{|A \cap R|}{|R|}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

1.2 Research Problems

Ontologies have grown increasingly large in real application domains, notably the biomedical domain, where ontologies, such as the Systematized Nomenclature of Medicine and

Clinical Terms (SNOMED CT) with 122464 classes, the National Cancer Institute Thesaurus (NCI) with 150231 classes, and the Foundational Model of Anatomy (FMA) with 104721 classes are widely employed. These ontologies can vastly vary in terms of their modeling standpoints and vocabularies, even for the same domain of interest. To enable interoperability we will need to integrate these large knowledge resources in a single representative resource. This integration can be established through a novel matching process which specifies the correspondences between the entities of heterogeneous ontologies. Existing ontology matching systems have to overcome two major issues when dealing with large ontologies: (i) integrating the large size not yet feasible with a good matching accuracy, (ii) automating the ontology matching process.

1.2.1 Large Ontology Matching

With regards to the first issue, according to the bioportal, large ontologies like NCI and SNOMED respectively contain 137,328 and 327,128 entities. For instance, the traditional alignment approaches perform the Cartesian product between all the entities of NCI and SNOMED ontologies, which produces a set of 137,328327,128 pairwise comparisons. This set corresponds to almost 45 billion candidate alignments. This constructs a huge search space, here the number of correspondences that should be specified between these ontologies is 18,844 according to the LargeBioMed Track from the Ontology Alignment Evaluation Initiative (OAEI), which represents only 4.19% of the candidate alignments. Therefore, the alignment of large ontologies is a complex task. The large size of these ontologies decreases the matching accuracy of ontology matching systems. Large ontologies describing the same domain includes a high conceptual heterogeneity. Ontology developers can construct the same domain ontology but using different conceptual models. As a result, finding mappings between two ontologies became more difficult. Consequently, the matching of large ontologies became error-prone, especially while combining different matchers in order to result in an adequate result. To summarize, the main issues of the alignment of large ontologies are the conceptual heterogeneity, the high search space and the decreased quality of the resulted alignments. Dealing effectively with biomedical ontologies requires a solution that will align large alignment tasks such as ”divide and conquer” or parallelization approaches. The ”divide and conquer” strategy means breaking a problem down into two or more sub-problems until the problem becomes simple enough to be solved directly. The solutions to the sub-problems are then combined to give a final solution for the original problem. When applied to the context of ontology alignment, the ”divide and conquer” approach consists of dividing a large matching task into a set of smaller sub-matching tasks known as partitions or blocks.

1.2.2 Automation of the Matching Process

A single matcher for a large ontology matching task is insufficient to result in good matching quality. Usually, a matcher treats a single type of heterogeneity. Hence, the combination of different matchers and the tuning of each matcher is essential for a better matching accuracy [60]. This process should be automatically defined by ontology matching systems. Therefore, an ontology matching system should not depend on a single configuration of matchers for all the matching problems. The matching tuning process should be automatically defined for every new matching context. Dealing with large ontologies should not preclude a high-quality solution. High quality is affected by the tuning parameters chosen during the alignment process, such as the thresholds, the appropriate matchers and their weights. While dealing with different matching tasks, the main issue is the automation process is the choice of the matching settings. The matching tuning process should be automated in order to reduce the matching process complexity, especially while dealing with large scale ontologies. As a result, the ontology matching process needs to be self-tuned for a better selection of matching settings for each matching problem. This process can improve the ontology matching accuracy. In the case of large ontologies, it is important to have highly-automated, generic processes which are independent of the input ontologies. To achieve quality alignments, ontology matching systems can employ a variety of matchers while managing complex ontologies [72]. The choice of these matchers should depend on the matching context. In the context of large ontologies, the drawback of manual solutions is the level of complexity and the time needed to generate results for such a large problem. Automatic solutions proposed in recent research work apply matching learning techniques based on various machine learning approaches. Matching learning corresponds to the use of machine learning for ontology matching.

1.3 Thesis Contribution

In this Ph.D. thesis, we put forward a novel local matching learning approach that combines ontology partitioning with ontology matching learning. In the following, we describe the main contribution of this thesis.

- A novel partitioning approach based on hierarchical agglomerative clustering [60]. As input, it takes two ontologies and generates as an output a set of local matching tasks. The partitioning approach split a large ontology matching task into a set of sub-matching tasks. The large search space is reduced accordingly to the number of local matching tasks. Therefore, the search space is minimized from the whole ontology matching problem to a set of sub-matching problems. Consequently, the alignment of the two input ontologies can be more effective for each sub-matching task in order to result in a better matching accuracy for the whole matching problem. The proposed partitioning approach is based on a novel multi-cut strategy generating not large partitions or not isolated ones.

- A local matching learning approach in order to fully automate the matching tuning for each local matching task [64]. This automation has to be defined for every new matching context in order to result in a context-independent local matching learning system. This matching system should align each local matching context based on its characteristics. State-of-the-art approaches define a set of predefined matching settings for all the matching contexts. However, the benefit of the local matching learning approach is the use of machine learning methods, which can be flexible and self-configuring during the training process. We apply the proposed matching learning approach locally and not globally. Consequently, we set the adequate matching tuning for each local matching task. Therefore, we result in a better matching quality independently of the matching context. Each local matching task is automatically aligned using its local classifier from its local training set. These local training sets are generated without the use of any reference alignments. Each local classifier automatically defines the matching settings for its local matching task in terms of the appropriate element-level and structural-level matchers, weights and thresholds.

- A prototype called POMap++ implementing all the above contributions. This prototype has participated in OAEI'2017 and OAEI'2018 have got top positions.

The work on the current PhD thesis have lead to the following publications:

1. **Amir Laadhar**, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri. The Impact of Imbalanced training Data on Local matching learning of ontologie. International Conference on Business Information Systems (BIS 2019), Seville, Spain, 26/06/19-28/06/19, june 2019.

2. **Amir Laadhar**, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri. Partitioning and Local Matching Learning of Large Biomedical Ontologies. ACM/SIGAPP Symposium on Applied Computing (SAC 2019), Limassol, Cyprus, 08/04/19-12/04/19, ACM SIGAPP, april 2019.

3. **Amir Laadhar**, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri. OAEI 2018 results of POMap++. International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (OM@ISWC'18 2018), Monterey, CA, US, 08/10/18-12/10/18, CEUR-WS : Workshop proceedings, p. 192-196, 2018.

4. **Amir Laadhar**, Faiza Ghozzi, Ryutaro Ichise, Imen Megdiche, Franck Ravat, Olivier Teste. Partitioning and Matching Tuning of Large Biomedical Ontologies. International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (OM@ISWC'18 2018), Monterey, CA, USA, 08/10/18-12/10/18, CEUR-WS : Workshop proceedings, p. 220-221, 2018.

5. **Amir Laadhar**, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri. POMap results for OAEI 2017. Dans : Ontology Matching co-located with the 16th International Semantic Web Conference (OM@ISWC'17 2017), Vienna, Austria, 21/10/17-25/10/17, CEUR-WS : Workshop proceedings, october 2017.

6. **Amir Laadhar**, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, Faiez Gargouri. POMap: An Effective Pairwise Ontology Matching System. International Conference on Knowledge Engineering and Ontology Development (KEOD 2017), Funchal, Madeira, Portugal, 01/11/17-03/11/17, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, p. 1-8, 2017.

1.4 Research Overview

In this section, we briefly describe the main processes of the proposed contributions as depicted in Figure 1.4. This architecture has two ontologies as the input and alignments as the output. The output is a set of correspondences generated from the two input ontologies.

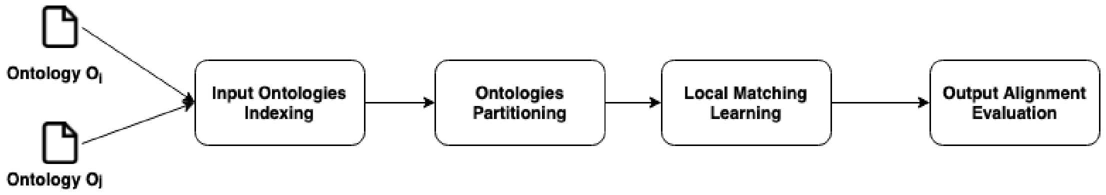


Figure 1.4: Contribution Overview

1. The two input ontologies are pre-processed and indexed in the first module. We applied a set of natural language processes across the annotations for each input ontology. All the annotations and semantic relationships between entities are stored in a data structure.

2. In the second module, the indexed ontologies are then partitioned in order to generate the set of local matching tasks. The partitioning process ensures good coverage of the alignments that should be discovered.

3. In the third module, we automatically build a local classifier for each local matching task. These local classifiers automatically align the set of local matching tasks based on their adequate features.

4. In the fourth module, the generated alignment file stores the set of correspondences located by all the local matching tasks. The correspondences are compared to the reference alignments provided by the Gold Standard to assess the accuracy of local matching.

1.5 Manuscript Outline

The remainder of this thesis is organized as the following:

- *Chapter 2* reviews and compares the existing works related to the alignment of large biomedical ontologies. It includes three main sections. The first one focuses on reviewing the state of the art matching systems. In this section, we classify the related ontology matching systems based on the employed matching techniques. The second section reviews the ontology partitioning approaches, which are classified into two main categories: Partition-based method and anchor-based segmentation method. The third section reviews the ontology matching learning approaches. Each of the two later sections is followed by a discussion in order to review the employed techniques.

- *Chapter 3* starts by presenting the formal foundations employed throughout the chapter. Then, we introduce the ontologies partitioning architecture. Later, we present the three main components of the ontology partitioning approach, which are: input ontologies pre-processing, partitioning algorithm and identification of local matching tasks. Each part is illustrated through examples.

- *Chapter 4* presents the local matching learning approach. We begin by presenting the formal foundations employed in the chapter. Then, we present an architecture presenting the main components of the local matching learning approach. The approach mainly contains three components: the local training sets generation, the balancing of the generated local training sets, the wrapper-based feature selection and the local classification through a set of classifiers.

- *Chapter 5* corresponds to the evaluation of our main contributions. We begin by presenting the Experimental environment. We are based on the biomedical datasets provided by the benchmark of the Ontology Alignment Evaluation Initiative (OAEI). Then, we evaluate the partitioning approach compared to the state-of-the-art methods based on the same dataset. Later, we asses the local matching approach based on different experiments. Each experiment is followed by a discussion.

- *Chapter 6* summarizes the thesis with a discussion about the different results and concludes with some perspectives and future research directions.

Chapter 2

Related Literature

Contents

2.1 Introduction

Large ontologies include a high conceptual heterogeneity, especially for biomedical ontologies. Usually, these ontologies are developed and maintained by different oncologists and researchers. Moreover, the same domain ontology can be constructed based on different modeling views. As a result, the matching process performed by matching systems can be impacted. Finding all the correspondences between two large ontologies became a challenging task. This can affect the matching accuracy, which can be decreased. Furthermore, the alignment of large ontologies results in a huge search space. For instance, the number of candidate correspondences between the ontologies NCI and SNOMED is more than 45 billion entity pairs. In this huge search space, ontologies matching systems try to find the wright correspondences. This process is error-prone especially while combining different matchers. As a result, the accuracy of the matching process can be impacted. To summarize, the main issues of aligning large scale ontologies are the conceptual heterogeneity, the high search space and the decreased quality of the resulted alignments.

Moreover, automating the alignment process of ontologies is presenting an issue for the state-of-the-art matching systems [35]. Furthermore, the alignment automation of large ontologies is representing a challenge for the semantic web community. Each matching task is different from the other matching tasks in terms of its semantic richness and its heterogeneity. Different ontology matching tasks can have different kinds of heterogeneity. Therefore, ontology matching systems should take into account the difference between ontology matching tasks. Moreover, ontology matching systems employ different matchers since a single matcher is not sufficient to cope with the heterogeneity of a large matching task. Different matching tasks require different matching settings such as the set of employed matchers, their weights and the threshold of each matcher. Therefore, an automatic combination of matchers is required especially for large matching tasks. The matching settings of an ontology matching system should not manually be defined. We argue the choice of the automatic approach due to the complexity of the manual approach particularly for large matching setting, which requires multiple matchers. Moreover, each ontology matching task has its own specific matching settings that are different from the other matching tasks. Therefore, the automation process of the matching tuning should be taken into account by ontology matching systems. In Section 2.4 we review the related works for ontology matching based on machine learning techniques.

In Section 2.2, we classify ontology matching systems based on their employed techniques. Since we are proposing an ontology partitioning approach, in Section 2.3, we review the related work for this field. Partitioning approaches are compared and reviewed. Dealing with ontology matching automation, in Section 2.4, we review the state-of-the-art for ontology matching learning.

2.2 Ontology Matching Systems

Different ontology matching systems have been approached by the semantic web community. Several surveys [90, 56, 22, 35, 56, 86, 85] compare the state-of-the-art approaches. Since the appearance of the OAEI (Ontology Alignment Evaluation Initiative) campaign, many ontologies matching systems have been proposed and compared. In this section, we classify and discuss the recent systems proposed by the state-of-the-art. Most of these systems have participated in the OAEI campaign. Classification is guided by the different techniques that can be employed during the matching process.

Aiming to review the ontology matching systems and inspired by the state-of-the-art surveys [90, 35, 86, 85], we classify them under the following groups: (i) basic matching techniques; (ii) scalability techniques and (iii) workflow strategies. Basic matchers are employed by almost matching systems. They are used to discover mappings between the entities of pairwise ontologies. Each ontology matching system can employ different basic matching techniques in order to cope with the high heterogeneity of ontology matching tasks. Scalability matching techniques are employed to align large scale ontologies. Workflow strategies are employed to combine different matchers [95]. For instance, in a sequential matching workflow, each matcher can reuse the generated correspondence of an earlier matcher. In a parallel workflow, matchers can be implemented at the same time in order to combine their results. In Figure 2.1, we draw the classification of the matching techniques as well as the workflow strategies.

New techniques are required to deal with the large size of the ontologies in order to deliver a good matching accuracy. During the last years, the alignment of large ontologies is getting a lot of interest by the semantic web community. In the following, we review the state-of-the-art matching techniques based on the classification depicted in Figure 2.1.

2.2.1 Basic Matching Techniques

Basic matching techniques are also known as individual matchers. These techniques usually implement a similarity measure that explores a single feature of an entity [61]. An ontology matching system can include several basic matchers, which are combined in order to deliver a final set of alignments. According to the survey of Euzenat and Pavel Shvaiko [35], these individual matchers are classified as follows.

2.2.1.1 Terminological methods

Terminological methods compare the strings of the names, labels, and comments of entities in order to align the similar ones. These methods use string similarity measures in order to asses the similarity between two strings. Similarity measures are classified into four categories: (i) Character-based [18] (ii) Edit-distance [68] (iii) Term-based [48] and (iv) Subsequence-based [70]. This method is used by many ontology matching systems such as AML [37], XMap [30] and Lily [108].

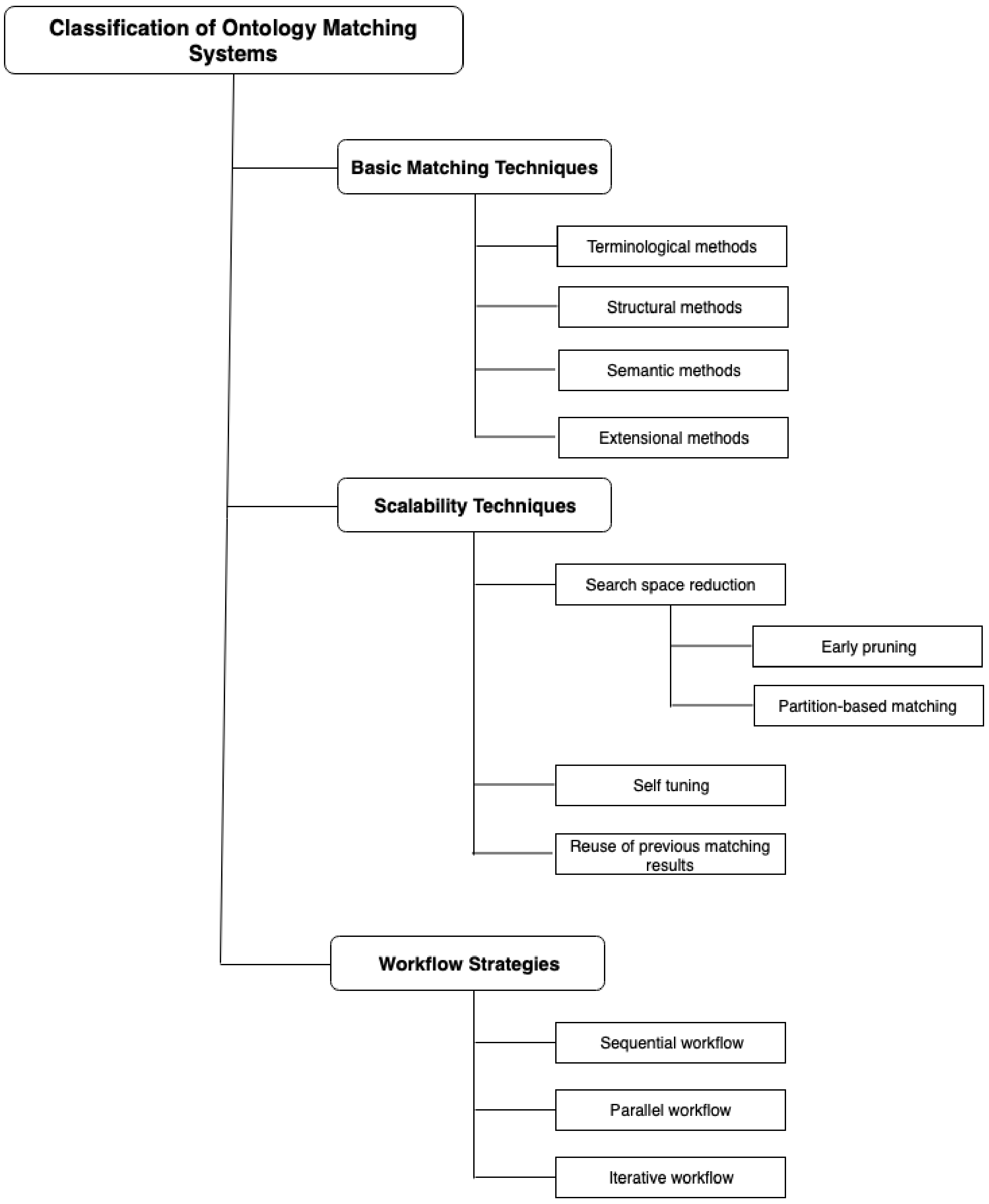


Figure 2.1: Classification of ontology matching systems.

2.2.1.2 Structural methods

Structural methods explore the structural characteristics between the entities of an ontology to derive correspondences. The structural information between two entities can be the subsumption relation, domain, range and restriction property. For instance, if two classes are similar, then their siblings can be somehow similar [35] and if two classes are similar, then their subclasses should be also similar. These methods are divided into two categories: internal and external structure methods [35]. Internal methods employ the domain, range, and cardinality of their properties in order to compute the structural similarity [30]. Some ontology matching systems developed their own structural similarity measures [74]. These measures compute the overlap between two sets of entities taken from super entities, descendant entities or based on the path from the root to the to-be-matched entities. Nevertheless, the structure of ontologies describing the same domain can be different. Therefore, it is not sure that two similar entities from two ontologies can have similar neighbors [35]. Consequently, some ontology matching systems employ filter techniques to find wrong correspondences [37, 52].

2.2.1.3 Semantic methods

Semantic methods explore semantic information encoded in the entities of an ontology. Therefore, if two entities have similar semantic relations, such as the same property, therefore, they can be similar. Description logic employs semantic information in order to discover inconsistent mappings. These inconsistent mappings are removed from the final mappings set. Reasoning can be used to rewrite and expand relations between entities. For example, new semantic relationships between entities can be generated. Therefore, new mapping can be discovered. Another type of using description logic is to transform the resulted mappings to the optimization problem on constraint programming. Annotating an ontology class by new semantic information extracted from background knowledge sources can facilitate the discovery of new mappings. Therefore, the heterogeneity between two ontologies is reduced and new correspondences can be discovered. Therefore, upper-level ontologies can be employed as external knowledge sources in order to align lower level ontologies describing the same domain. DBPedia [10] and YAGO [103] are multi-purpose upper ontology. For biomedical domain, UMLS [16] is a generic medical dictionary. Uberon [78], FMA [38] and GO [24] are specific upper level for bio-medical domain. These upper-level ontologies can employ the Open Biomedical Ontologies (OBO) vocabulary [98] in order to interconnect similar classes of different ontologies. More specifically, the annotation property ”oboInOwl:hasDbXref” is employed by the OBO vocabulary to interconnect the classes of different ontologies describing the same concept. Therefore, the names of each class can be extended by new definitions and names. Many ontology matching systems employ external background knowledge to annotate and align ontologies.

2.2.1.4 Extensional methods

Other methods compute the similarity between classes by comparing their instances. Therefore, the schema matching can be based on the data layer matching. For instance, two classes can be similar if their instances are similar. This kind of methods can be employed when the information about the schema layer is limited. For example, extensional methods can be used when there is no names, labels, and comments for classes. Such methods can be found in several ontologies matching systems live such as ASMOV [49], and AROMA [28].

2.2.2 Workflow Strategies

Basic matching techniques are not able to discover a set of mapping that satisfies user requirements. Each basic matcher usually focuses on a specific type of heterogeneity between the entities of two ontologies. Therefore, ontology matching systems combine several basic matchers in order to deliver a better matching accuracy. The state-of-the-art follows three workflows strategies in order to combine matchers: sequential, parallel and interactive composition. The combination of the later compositions is identified as a hybrid strategy. In the following, we discuss the state-of-the-art workflow strategies.

2.2.2.1 Sequential Workflow

The most usual method to compose basic matchers is the sequential workflow. The output of each matcher is the input of the next matcher. This method is used by many ontology matching systems. Usually, an element-level matcher generates an initial set of correspondences, which are passed to a structural-level matcher. This strategy is employed by Gomma [39], Falcon-AO [44], Similarity Flooding [74]. For instance, in a sequential workflow, we can employ at least two matchers executed sequentially. The first matcher aims to find all the possible candidate mappings, the second matcher discovers all the possible inconsistent correspondences produced between the two matchers. Therefore, there is a dependence between matchers.

2.2.2.2 Parallel Workflow

In the parallel workflow strategy, matchers are independent. Each matcher is executed independently from the other matchers. If each basic matcher is employing a similarity measure, then the similarity scores generated by each candidate matcher are aggregated based on an aggregation operator. The commonly used aggregation operators are the weighted sum and the weighted average. Other matching systems can assign manually the weigh for each matcher COMA [11], ASCO [67]. Other systems proposes adaptive weights AgreementMaker [25] or fuzzy assign weights (OWA) [50]. Some other ontology matching systems aggregate operators via a decision function. Therefore, basic matchers can be combined based on machine learning classifiers [46].

2.2.2.3 Iterative Workflow

The purpose of the iterative workflow strategy is that the matching process is repeated many times until converging to a stop condition. The matching process can be performed by one basic matcher or a combination of several basic matchers. This type of strategy can be used in a part of a system or in the whole system. A typical example of using iterative method in a part of system is similarity propagation method. The principle of the algorithm is that the similarity between two nodes must depend on the similarity between their adjacent nodes [74, 108]. Therefore, at each step of the running algorithm, the similarity value of each pair of entities is re-computed according to the current values of itself and its neighbors. We can find this strategy in the second phase of works in Similarity Flooding [74], OLA [57], Falcon-AO [44], Lily [108]. Similarly, the iterative strategy is also used in constraint-based method. In that way, for each step, the constraint-based method re-calculates the confidence values for every candidate mappings or removes inconsistent mappings. This process will be stopped until the optimization condition is reached. The iterative constraint-based methods can be found in second phase of GLUE [31], CODI [45], LogMap [52].

2.2.3 Scalability Techniques

Large ontologies like NCI and SNOMED respectively contain 137,328 and 327,128 entities. The traditional alignment approach uses the cartesian product between all the entities of NCI and SNOMED ontologies, which produces a set of $137,328 \times 327,128$ pairwise comparisons. This set corresponds to almost 45 billion candidate alignments. The number of correspondences that should be specified between these ontologies is 18,844 according to the LargeBioMed Track from the Ontology Alignment Evaluation Initiative (OAEI) [1], which represents only 4.19% of the candidate alignments. Dealing effectively with biomedical ontologies requires a solution that will align large alignment tasks such as search space reduction (early pruning and partitioning methods), self-tuning and reuse of previous matching results [35, 89].

2.2.3.1 Search Space Reduction

Early pruning This method reduces the search space by heuristically eliminating the set of the candidate correspondences. For instance, Eff2Match [23] employ the top-K entities algorithm in the target ontology according to their context virtual document similarity. This methods heuristically selects candidate mappings. For instance, the ontology matching system QOM [34] proposed a heuristic strategy based on different extracted features such as label, hierarchy, neighbors to select the promising mappings.

Partitioning Methods These methods split two large ontologies into a set of sub-ontologies based on their structural information. The alignment process is performed only between the entities of the sub-ontologies instead between the whole ontologies. Some

[1]http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/

methods are based on anchors in order to extract the set of sub-ontologies. An anchor is a set of entity-pairs determined by a similarity measure or other sophisticated techniques. Each sub-ontology of one ontology have only one corresponding sub-ontology of the second ontology. Then, the alignment process will be performed for each pair of related sub-ontologies. This strategy is proposed by several ontology matching systems such as: Anchor-Prompt [84], AnchorFlood [97], Lily [108] and TaxoMap [42].

2.2.3.2 Self Tuning

Several match systems first analyze the schemas to be matched and determine their linguistic and structural similarity. These similarity characteristics are then used to select matchers or to weigh the influence of different matchers in the combination of matcher results. The Rimom system uses such similarity factors for dynamically selecting matchers for a specific match task. For example, they use string measures for name matching only if the input schemas have highly similar names; otherwise, they rely on thesauri such as Wordnet. Similarly, they apply structural matching only if the input schemas are deeply structured and structurally similar. Falcon-AO uses linguistic and structural similarities to combine matcher results, particularly to optimize individual similarity (cutoff) thresholds. For example, if the linguistic similarity is high, Falcon-AO uses lower thresholds for linguistic matchers so that more of their correspondences are considered [89].

2.2.3.3 Reuse of Previous Matching Results

A promising approach to improve both the effectiveness and efficiency of schema matching is the reuse of previous match results to solve a new but similar match task. An ideal situation for such a reuse is the need to adapt a mapping between two schemas after one of them evolves to a new schema version. By reusing the previous match mapping for the unchanged schema parts, a significant amount of match effort can likely be saved. The reuse of previously determined correspondences and match results may also be applicable in other cases, especially when different schemas share certain elements or substructures, such as standardized address information. Exploiting the reuse potential requires a comprehensive infrastructure, particularly a repository to maintain previously determined correspondences and match results. Furthermore, methods are necessary to determine the schema elements and fragments for which match reuse is applicable. Reuse can be exploited at three mapping granularities: for individual element correspondences, for mappings between common schema fragments, and for complete mappings and schemas [89].

2.2.4 Discussion

Due to the decentralized semantic web, ontologies are highly heterogeneous. As a result, ontology matching becomes a crucial task in semantic web applications. Therefore, ontology matching systems are developed in order to discover correspondences between semantically related entities of ontologies. Ontology matching systems should be efficient, especially while aligning large ontologies due to their high heterogeneity. These systems propose different approaches for the matching that are implemented in ontology matching algorithms exploring different ontology matching techniques. Therefore, in the following Table 2.1, we perform a comparative study of the prominent state-of-the-art matching systems. This comparative study aims to compare the state-of-the-art matching systems to our proposed approach called POMap++. We are based on three main criteria for the comparison: basic matching techniques, scalability techniques, and the workflow strategy. These criteria are presented in the earlier sections of this chapter.

Table 2.1: Comparative of ontology matching systems

Approach	Basic Matching techniques	Scalability techniques	Workflow strategy
Eff2Match [23]	Structural, Terminological	Early pruning	Sequential
QOM [34]	Structural, Terminological	Early pruning	Sequential
CODI [45]	Structural, Terminological, Semantic	No	Iterative
GLUE [31]	Structural, Terminological	No	Iterative
Agreement Maker [26]	Structural, Terminological	No	Parallel
RiMOM [109]	Structural, Terminological, Structural	Parallel	Iterative
Lily [108]	Structural, Terminological	Partitioning and reduction anchors	parallel
OLA [57]	Structural, Terminological	No	Iterative
ASCO [67]	Structural, Terminological	No	Sequential
ASMOV [49]	Structural, Terminological, Semantic	No	Iterative
AROMA [28]	Structural, Terminological	Early pruning	Iterative
AML [37]	Structural, Terminological, Semantic	Hashmap data structure	parallel
LogMap [53]	Structural, Terminological, semantic	logic based	iterative
XMAP [30]	Structural, Terminological, semantic	divide-and-conquer	Parallel
FCAMapX [43]	Structural,Terminological, semantic	formal concept analysis	Iterative
YAM-BIO [8]	Semantic, Terminological, Structural	Candidates correspondences filtering	Sequential
CroMatcher [17]	Semantic, Terminological, Structural	No	Parallel
COMA++ [11]	Terminological, Structural	Partitioning	Parallel
Falcon-AO [44]	Terminological, Structural	Partitioning	Sequential
Anchor Flood [97]	Terminological, Structural	Anchor-segmentation	Iterative
Anchor-prompt [84]	Terminological, Structural	Anchor-segmentation	Sequential
Taxomap [42]	Terminological, Structural	Partitioning	Sequential
Gomma [39]	Semantic,Terminological, Structural	Partitioning	Parallel
Xue and Pan [114]	Terminological, structural	Partitioning	Sequential
KEPLER [55]	Terminological, Semantic	Partitioning	Sequential
Ngo and Bellahsene [82]	Semantic, Terminological, Structural	Candidates correspondences filtering	Sequential
Ryutaro Ichise [46]	Semantic, Terminological, Structural	no	n/d
Nezhadi et al. [80]	Semantic, Terminological, Structural	No	Sequential
Eckert et al. [33]	Semantic, Terminological, Structural	No	n/d
Wang et al. [107]	Terminological, Structural	Candidate selection	Sequential
F. Cruz et al. [27]	Semantic, Terminological, Structural	No	Parallel
Nkisi-Orji et al. [83]	Semantic, Terminological, Structural	No	Sequential
POMap++ [63]	**Terminological, Structural**	**Partitioning**	**Parallel**

Firstly, we classify existing approaches based on their use of basic matching techniques, which can use the following matching techniques: terminological, structural and semantic. Next, for the second criterion, we state the employed techniques in order to deal with the alignment of large scale ontologies. We argue the use of this criterion since our approach deals with large scale ontologies. The final criterion corresponds to the employed workflow strategy, which can be in sequential, in parallel or iterative. We were able to add an additional criterion that states the automation of the approaches. However, many ontology matching systems claim to be automated, but in reality, these systems introduce a manual tuning process. Usually, a predefined manual tuning is defined for ontology matching tasks after the evaluation of the generated correspondences. Even, ontology matching learning systems lack automation in terms of the generation of the training data. The majority of the existing matching systems build the training data either manually or extract it from the reference alignments. However, reference alignment does not usually exist for the majority of ontology matching tasks. Moreover, even a single automated generated training set is not rich enough to include all the possible patterns for all the ontology matching tasks. Therefore, the learned classifier can contain wrong assumptions that can result in wrong correspondences for the ontology matching tasks. We can deduce from the performed comparative study, that there is a lack of matching systems that combine scalability techniques to a fully automated ontology matching approach. Nevertheless, it is highly required to automatically align large ontology matching tasks. As a result, our proposed approach, POMAp++, combines large scale ontologies matching to a fully automated matching learning approach. This approach is called local matching learning, which divide a large scale ontology matching into a set of local matching tasks aligned automatically using machine learning techniques. In the next section, we review the existing ontology partitioning approaches.

2.3 Ontologies partitioning

Some of the matching systems perform an all-against-all comparison between the entities of the input ontologies [36]. However, this strategy is not suitable for dealing with large and heterogeneous 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 [101]. Partitioning and modularization techniques have been also employed by the semantic web community in order to solve some tasks such as (e.g., ontology visualization [102, 3], ontology reuse [54], ontology debugging [104], ontology classification [92]. Partitioning has also used [51]. Another line of work perform the partitioning process of pairwise ontologies to reduce the complexity of large ontology matching task (e.g., [11, 44, 4, 39, 114, 21, 88]. The alignment of the large scale ontologies is one of the main issues in the ontology matching field. The size of the input ontologies impacts the resulted matching quality in terms of effectiveness and efficiency. There is a very high conceptual heterogeneity in large ontologies. This heterogeneity results in a difficulty in finding all the possibles mappings between ontologies. Therefore, ontologies matching systems can be affected due to the large search space that can be generated. The discovery of the mapping in huge search space is time-consuming. Consequently, it may lead to wrong alignments.

In this section, we focus on reviewing the approaches of large ontologies matching. Notably, we review large scale ontology matching methods in particular partition-based method in Section 2.3.1. Then, we review the anchor-based segmentation method and discussed in Section 2.3.2. Later, in Section 2.3.3, we will discuss the strength and weaknesses of the state-of-the-art methods.

2.3.1 Partition-based Methods

This method is similar to the divide and conquer approach. The partition-based methods break down a big matching problem into a set of sub-problems. The sub-problems correspond to the matching of the set of sub-ontologies [101].

Commonly, partition-based matching strategies follow two tasks:

1. The partitioning of the input ontologies can be performed by applying a clustering algorithm. The entities of each cluster are semantically related to each other [101, 51]. Whereas, the entities of different clusters are weakly related. We assume that partitions represent different sub-domains for each ontology.

2. The identification of similar partitions-pairs is accomplished through the comparison between the partitions of the two ontologies [101]. This comparison is made by computing the similarity or based on a set of anchors between the set of partitions. The intuition is if two partitions of two ontologies are similar, then they describe the same sub-domain.

Recently, Jimenez-Ruiz et al. [51] proposed a divide and conquer approach that partition large ontologies into a set of sub-matching tasks based on two clustering strategies: (i) Naive strategy and (ii) neural embedding strategy. Jimenez-Ruiz et al have performed a comprehensive evaluation of both strategies which suggests that the obtained divisions are suitable in practice in terms of both coverages. However, the naive and the neural embedding strategies require the size of the number of matching subtasks or clusters as input. The (required) matching subtasks have to be known beforehand if, for example, the matching tasks are to be run in parallel in a number of available CPUs [51] .

COMA++ [11] was one of the first matching systems proposing an ontology partitioning approach. COMA++ is a schema matching tool based on a parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for the evaluation of the different matchers [35]. COMA contains 6 elementary matchers, 5 hybrid matchers, and a reuse-oriented matcher. Most of them implement string-based techniques, such as affix, n-gram edit distance (Sect. 5.2.1); others share techniques with Cupid, e.g., thesaurus lookup. An original component, called reuse-oriented matcher, tries to reuse previously obtained results for entire new schemas or for their fragments. Schemas are internally encoded as directed acyclic graphs, where elements are the paths [35].

Falcon-AO [44] Falcon present a divide-and-conquer approach to ontology matching. The approach operates in three phases: (i) partitioning ontologies, (ii) partitions matching, and (iii) discovering alignments. The first phase starts with a structure-based partitioning to separate entities (classes and properties) of each ontology into a set of small clusters [35]. Partitioning is based on structural proximities between classes and properties, e.g., how close the classes are in the hierarchies of rdfs:subClassOf relations, and on an extension of the ROCK agglomerative clustering algorithm [35]. Then it constructs blocks out of these partitions. In the second phase, the partitions from distinct ontologies are matched based on anchors. The anchors are discovered by matching entities with the help of the SMOA string comparison technique [35]. Finally, the third phase combines two matchers between the matched partition pairs via sequential composition [35]. Nonetheless, this approach is evaluated through only one pair of ontologies from the web directories matching task of the OAEI.

TaxoMap [42] offers two partitioning methods refining the Falcon-AO method by attempting to introduce more dependency between the two partitions. The main difference between these methods is the order of extraction of anchors in order to discover the sub-matching tasks [35]. The first algorithm, Partition-Anchor-Partition (PAP), first partitions one of the ontologies based on its own structure; then it computes anchors between the ontologies and partitions the second ontology starting at the anchors [35]. The second method, Anchor-Partition-Partition (APP), is more adapted to unstructured ontologies. It partitions the two ontologies by starting from the anchors that are found in these ontologies, and for the second ontology, groups of anchors that are in the same partitions. In both cases, partitions are then paired together based on anchors [35].

SeeCOnt [4] introduced a partitioning approach that analyses each input ontology to derive the root of each partition. This approach employed a ranking function that assigns each entity of an ontology to a single partition root. Using this technique improve the matching efficiency by reducing the search space and produce acceptable matching results. However, the downside of this technique is that it requires an expert to determine the maximum size of each cluster. Moreover, the number of roots is manually defined.

Most of the existing work suffer from the low coverage value of the generated partitions [88]. For instance, the partitioning methods Falcon-AO , PAP, and APP obtained a coverage value of 80%, 34%, and 48%, respectively for the FMA-NCI matching task of OAEI [88]. Pereira et al. [88] performed a study with the PBM method of Falcon-OA, and the PAP and APP methods of TaxoMap. The results in terms of coverage with the largebio tasks were very low, which directly affected the results of the evaluated systems [88]. We are motivated by these obtained negative result to propose our partitioning approach of this thesis.

2.3.2 Anchor-based Segmentation Method

This method is different from the partition-based method in terms of the generation of partitions. The partition-based method can generate anchors in order to find similar partitions between the two input ontologies. Whereas, the anchor-based segmentation method generates anchors in order to construct the set of partitions or called also segments. The anchor-based segmentation method is also known as a dynamic selection of candidates. Therefore, this method iteratively updates the set of candidate correspondences. Candidate correspondences are generated by exploring the structural characteristics of entities in order to remove the possible wrong candidate mappings. This method is proposed by Anchor PROMPT [84] and Anchor Flood [97] matching systems.

The main steps of the anchor-based segmentation method are listed in the following:

1. The first step is the generation of the initial set of anchors between the two input ontologies. This can be accomplished through a fast similarity measure or the exact matching.

2. The neighbors of each anchor is explored in each iteration. Anchors are considered as the set of aligned entities. These anchors represent the root of each segment. For each selected anchors the algorithm updates the entities belonging to its segment. This update is made by adding the neighboring entities of an aligned anchor. The neighboring entities correspond to the superclasses, subclasses, and siblings. The intuition is that if two entities are similar, their neighbors can be also similar.

3. An alignment process is performed between the set neighbors in order to produce a new set of anchors. These anchors are employed in the next iterations.

4. The iterative process is repeated until there is no more entity to explore. The output is a set of aligned entities between the segments of the two input ontologies.

Anchor-Flood [97] starts with anchors in the same way as above. It then compares the neighborhoods (or partitions) of both anchors, i.e., the set of entities connected to the anchors two levels away (parents, grandparents, children, grandchildren, siblings, etc.) [35]. The algorithm only compares entities from two such anchored partitions, starting from the anchors and spreading to the neighborhood until all entities are reached [35]. The pairs of entities to match are those reachable from the same type of operation (ascending, descending, sibling)

Anchor-Prompt [84] is a sequential matching algorithm that takes as input two ontologies, internally represented as graphs and a set of anchors, which are identified with the help of string-based techniques, such as edit distance, user-defined distance or another matcher computing linguistic similarity. Then the algorithm refines them by analyzing the paths of the input ontologies limited by the anchors in order to determine terms frequently appearing in similar positions on similar paths. Finally, based on the frequencies and user feedback, the algorithm determines matching candidates [35].

The anchor-based segmentation suffers while dealing with large scale ontologies. For instance, the FMA (78,989 classes) and NCI NCI.owl (66,724 classes) have in common only 2898 correspondence that should be discovered. Consequently, during the alignment process, the structural similarity value computed for each pair of concepts between the two segments is small. As a result, new correspondences may not be discovered during the next iteration. This issue results in a loss of the number of candidate alignments, which affect the recall value. For example, anchor-Flood obtained a not high Recall (0.682) in the OAEI 2008 Anatomy track.

2.3.3 Discussion

Large biomedical ontologies are developed by different oncologists and researchers. Therefore, these ontologies are characterized by high conceptual heterogeneity. Conceptual heterogeneity corresponds to the mismatches in the conceptualization of two ontologies modeling the same domain. During the conceptualization process, the different entities of an ontology are ordered according to a defined hierarchical structure. This hierarchical structure is defined by the ontology developer. Hence, the same domain ontology can be constructed based on different modeling views. Moreover, two similar entities of two ontologies can be associated with different semantic relationships and properties. Moreover, large biomedical ontologies include a high search space. An ontology matching systems tries to discover the right correspondences in the huge search space. This process is complex and time-consuming especially while combining different matchers. As a result, many wrong false positive and false negative correspondences can be generated. Therefore, the matching accuracy can be decreased due to the complexity of the task. Ontology matching systems suffer to cope with the later issues: the conceptual heterogeneity of large ontologies, the huge search space and the decreased resulted from correspondences.

In order to cope with the later issues, in particular, the large scale only matching, Erhard Raham [89] reviewed four of the promising state-of-the-art techniques: Reduction of search space (early pruning and partition-based matching), parallel matching, self-tuning matching and reuse of previous matching results. We are following the partition-based matching since all the other techniques can be integrated into this matching technique. For instance, we can perform partitions matching using a parallel matching workflow. As a result, sub-matching tasks can be aligned simultaneously in a parallel matching workflow. Moreover, we can integrate early pruning or self-pruning with partition-based matching. The main advantage of the partitioning approaches is that they can be used independently of the matching approach. Therefore, we can apply different matching approaches for the extracted partition-pairs. Therefore, we propose a partitioning approach in order to align large scale ontologies. In the following Table 2.2, we perform a comparative study of the state-of-the-art approaches for ontology partitioning. We are based on the following criteria for comparison: coverage ratio, number of sub-matching tasks, isolated partitions, and the employed dataset. Jimenez-Ruiz et al. [51] argued that is essential to evaluate the efficiency of partitioning approaches. The coverage of the matching subtask aims at providing guarantees about the preservation of the (potential) outcomes of the original matching task (i.e., information loss) [51]. It indicates if the relevant ontology alignments in the original matching task can still be computed with the matching subtasks [51]. The second comparison criteria correspond to the number of sub-matching tasks generated after performing the partitioning approach. This number can be automatically or manually defined. Unfortunately, there is a large variety of the possible number of sub-matching tasks that can be defined. However, the adoption of an automatic approach becomes increasingly necessary and should be considered by partitioning approaches. Another criterion is checking if partitioning approaches generate isolated partitions. Isolated partition is called isolated blocks by Pereira et al. [88]. Isolated partitions are partitions with only one entity. These partitions can result in either source or target ontologies by the partitioning approaches. Therefore, isolated partitions need an extra process to be aligned and not lost after performing the partitioning. The last criterion is employed dataset. We consider this criterion since some partitioning approaches claim to be effective however they employ small datasets. However, partitioning is most efficient for large datasets.

Table 2.2: Comparative of partitioning approaches

Approach	Coverage ratio	Number of submatching tasks	Isolated Partitions	Dataset
Jimenez-Ruiz et al. [51]	High	Manually defined	n/d	LargeBio and Phenotype
COMA++ [11]	n/d	n/d	n/d	Anatomy, Benchmark , Directory, Food
Falcon-AO [44]	Very low	Manually defined	high	Russia12 and TourismAB
Anchor Flood [97]	n/d	Automatic	n/d	Benchmark, Anatomy
Anchor-prompt [84]	n/d	Automatic	n/d	UMD and CMU
Taxomap [42]	Very low	n/d	high	Benchmark, Anatomy and Directory
Gomma [39]	n/d	n/d	n/d	Anatomy, Benchmarks and Library, Conference and Multifarm
Xue and Pan [114]	n/d	Semi-automatic	n/d	Bibliographic benchmarks, Anatomy, Library and LargeBio
KEPLER [55]	n/d	Manually defined	low	Anatomy, Conference, Multifarm, LargeBio and Phenotype
POMap++ [64]	**High**	**Automatic**	**No**	**Anatomy and LargeBio**

We can deduce from Table 2.2 that almost the partitioning approaches do not give importance to the coverage ratio. However, Jimenez-Ruiz et al [51] stated that this criterion is considered as primary in the evaluation of the partitioning approaches. Only the approach proposed by Jimenez-Ruiz et al [51] results in a high coverage ratio. However, this approach defines manually the number of sub-matching tasks. Only the anchor-prompt and Anchor-Flood approaches automatically generate. However, these approaches do not state the resulted coverage ratio. Moreover, these approaches partition relatively small ontologies. Dealing with isolated partitions, only the ontology matching system KEPLER cope with this issue. It generates a low number of isolated partitions. However, these approaches fail to align large ontology matching tasks. Therefore, our partitioning approach integrated into POMap++ focuses on addressing the issues concerning achieving a good average ratio, automatically defining the number of sub-matching tasks and without generating isolated partitions. Moreover, we are based on the dataset of large biomedical ontologies. We mention that almost the state-of-the-art method made no restriction on the size of the generated partitions. Therefore, unbalanced partitions may be produced, where some partitions are big and the other partitions are quite smaller. This may affect the accuracy of the ontology matching algorithm if it uses structural information. To the best of our knowledge, most of the existing approaches apply the same matching tuning over all the extracted sub-matching tasks. However, each sub-matching task represents a single sub-topic of interest, which has its own context and characteristics that should be taken into account during the matching process. The main advantages of the Anchor-based Segmentation methods, which are employed by Anchor-flood and Falcon-AO, is their memory efficiency. These methods suffer when the size of the input ontologies is very large like the Foundational Model of Anatomy ontology (FMA.owl - 78,989 classes) and the National Cancer Institute Thesaurus ontology (NCI.owl - 66,724 classes). Unlike related work, we divide a large ontology alignment task to a set of sub-matching tasks called local matching tasks with a good coverage ratio and without resulting in any isolated partitions or large partitions. We distinguished that the number of sub-matching tasks is automatically defined. We achieve these results due to the proposed approach for the iterative partitions merging. Then, we align each sub-matching task using its specific local settings. We automatically determine the local matching tuning using a specific machine learning classifier for each sub-matching task.

2.4 Ontology Matching Learning

Matching learning corresponds to the use of machine learning techniques for ontology alignment. Supervised matching learning is the task of automatically inferring a function f from training data. The learned function $f : x \Rightarrow y$ maps the input x to an output y [35]. The learned function is called a classifier for the learned function. A classifier is a function that assigns a class label to a data point. The input x is composed of a set of attribute values that describe the object to classify, while the output y is the class in which the object x will be classified by the learned classifier. The training data is a set of objects already classified (containing both attributes and a class value), while the test data is the set of objects to classify. A supervised machine learning algorithm analyzes the training data and produces a classifier that will be used to classify the test data objects.

In this section, we review the state of the art matching systems employing machine learning techniques in their matching process workflow. In Section 2.4.1, we will review the ontology matching learning systems. In Section 2.4.1.1, we will focus on the resampling of the training data for the proposed ontology matching systems. Later, in Section 2.4.1.2, we will focus on the use of feature selection methods for state-of-the-art ontology matching learning approaches.

2.4.1 Ontology Matching Learning Systems

There have been some relevant works dealing with matching learning [46, 80, 82, 33]. Machine learning approaches for ontology alignment usually follow two phases [35]: the training phase and the classification phase.

1. In the training phase, the machine learning algorithm learns the matching settings from a training set. This training set is usually created from the reference alignments of the same matching task.

2. In the classification phase, the generated classifier is applied over the input ontologies to classify the entity pairs of the input ontologies (matches or not).

Eckert et al.([33] built a meta-learner strategy to combine multiple learners. Malform-SVM [?] constructed a matching learning classifier from the reference alignments through a set of element level and structural level features. Nezhadi et al. [80] presented a machine learning approach to aggregate different types of string similarity measures. The latter approach is evaluated through a relatively small bibliographic matching track provided by the OAEI benchmark. Yam++ [82] defined a decision tree classifier based on a training set with different similarity measures. The decision tree classifier is built from the reference alignments. Existing matching learning approaches build their machine learning classifiers from the reference alignments or derive it manually from a particular matching task [35]. Wang et al. [107] proposed a method for enriching entities in an ontology with external definition and context information, and use this additional information for ontology alignment. We develop a neural network architecture capable of encoding additional information when available. Wang et al. showed that the addition of external data results in an F1-score of 0.69 on the Ontology Alignment Evaluation Initiative (OAEI) largebio SNOMEDNCI subtask [107]. Nkisi-Orji et al. [83] introduced a random forest classifier approach for ontology alignment which relies on word embedding for determining a variety of semantic similarities features between concepts. Specifically, Nkisi-Orji et al. [83] combined string-based and semantic similarity measures to form feature vectors that are used by the classifier model to determine when concepts match. This approach eliminates the need for learning the aggregation weights of a composition of similarity measures [83]. However, this approach is tested only based on small and medium-sized ontology matching tasks. Moreover, the training set is generated from the reference alignments without employing any resampling or feature selection.

Matching Learning which is considered as a classification problem. It is also concerned by different problems related to machine learning such as imbalanced data sets or the choice of the best features. Imbalanced data sets usually reflect an unequal distribution of classes within a dataset. Typically, they are composed of two classes: the majority (negative) class and the minority (positive) class. These type of sets suppose a new challenging problem since standard classification algorithms usually consider a balanced training set and this supposes a bias towards the majority class. Machine Learning algorithms tend to produce unsatisfactory classifiers when faced with imbalanced datasets. Feature selection is the process of selecting a subset of relevant features for use in model construction. It enables the machine learning algorithm to train faster. It reduces the complexity of a model and makes it easier to interpret. It also improves the accuracy of a model if the right subset is chosen. In the next subsections, we review the works dealing with (i) balancing training datasets and (ii) feature selection.

2.4.1.1 Balancing Training Sets

Classification problems often suffer from data imbalance across classes. This is the case when the size of instances from one class is significantly higher or lower relative to the other classes. A small difference often does not matter [76]. However, if there is a modest class imbalance in training data like 4:1, it can cause misleading classification accuracy. Imbalanced data refers to classification problems where we have an unequal number of instances for different classes. Most machine learning classification algorithms are sensitive to imbalanced training data. An imbalanced training data will bias the prediction classifier towards the more common class. This happens because machine learning algorithms are usually designed to improve accuracy by reducing the error. Different methods have been proposed in the state-of-the-art to handle imbalanced training data [29]. The common approaches [19] to generate a balanced dataset from an imbalanced one are undersampling, oversampling, and their combination:

- Undersampling approach balances a dataset by reducing the size of the abundant class. This method is used when the quantity of data is sufficient [76]. By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class. Undersampling is used when the amount of collected data is sufficient.

- Oversampling approach tries to balance a dataset by increasing the size of rare samples. Rather than removing the abundant samples, new rare samples are generated [76]. Oversampling is employed when the amount of data collected is insufficient.

- Performing a combination of oversampling and undersampling can yield better results than either in isolation [76].

Generally, oversampling is preferable because the undersampling can result in the loss of important data. Under sampling is suggested when the amount of data collected is larger than ideal and can help data mining tools to stay within the limits of what they can effectively process. Combining oversampling and undersampling can result in better results than either in isolation [76]. Despite the importance of the resampling process, existing matching learning approaches do not take into consideration the issue of the imbalanced dataset.

2.4.1.2 Feature Selection

Feature selection Feature selection methods can be classified into three categories [96]: Filters, Wrappers, and Embedded Methods. Filter methods are independents from the classification algorithm as they are run once a time before classification. These methods depend only on the intrinsic properties of a dataset [96]. Wrapper methods are dependent on the classifier. The subset of best features selected through the wrapper methods is validated through the global process of machine learning. Embedded methods are similar to wrapper methods with the advantage to be far less computationally intensive [96] than the wrapper methods.

Feature selection techniques are less investigated while dealing with ontology matching learning. A line of work is based on a set of predefined features [107, 27, 83], another line of work employs feature selection techniques [12, 115]. [83] proposed a matching learning classifier based on 23 features. [107] feed a neural network classifier with 32 features covering commonly used measures in the literature. A global classifier is built based on all the 32 features. Both approaches [83, 107] do not mention if the training set is balanced or not. Automatchm [12] performs a database schema matching using a probabilistic Bayesian learning strategy. Automatchm reduces the size of the dictionary mediating structure, by three feature selection methods: information gain, mutual information and likelihood ratio. This approach is based on a global matching process focusing on database schema matching. [27] proposed an ontology matching learning approach, based on a multi-class classifier, that employs five predefined configurations selected from the Agreement Maker ontology matching system ([26]. This approach aims to select the best configuration for each candidate alignment of an ontology matching task. However, this approach is limited to only five matching configurations. Moreover, the training set is built based on the reference alignments. The feature selection process is in somehow pre-established.

2.4.2 Discussion

A single matcher to align a large ontology matching task is not enough to result in good matching accuracy. As a result, ontology matching systems employ different matchers in order to align an ontology matching task. The combination of different matchers should be automated in order to choose the adequate matcher, the weight and the threshold of each matcher. When matchers are combined, each matcher can have a weight different than the other matchers. Moreover, each simgle matcher can be be associated with a parameter called threshold. This threshold is especially used by terminological matchers to trim the result of the similarity measures. Each ontology matching task has its specific matching settings. These matching settings should be automatically defined. The manual tuning process is time-consuming, especially for combining many matcher for large ontology matching task. Automating the alignment process of large ontologies is representing an issue for the existing approaches. As a result, in the following Table 2.3, we compare the state-of-the-art ontology matching systems based on matching learning approaches.

Table 2.3: Comparative of matching learning approaches

Approach	Training set generation	Features type	Feature selection	Resampling	Machine learning algorithm
Ryutaro Ichise [46]	n/d	Terminological, Semantic, Structural	No	No	SVM
Nezhadi et al. [80]	Manual	Terminological, Semantic, Structural	No	No	K-Nearest Neighbours, Decision Tree, SVM, AdaBoost
Ngo and Bellahsene [82]	Manual	Terminological, Semantic	No	No	Decision Tree
Eckert et al. [33]	From reference alignments	Ontology profiling, Terminological, Semantic, Structural	No	No	Decision Tree, Naive Bayes
Wang et al. [107]	Semi-automatic	Terminological	No	No (partial)	Neural network
F. Cruz et al. [27]	n/d	Ontology profiling	No	No	K-Nearest Neighbours, Naive Bayes, Neural Network, C4.5
Nkisi-Orji et al. [83]	From reference alignments	Terminological, Semantic	No	No	Random forest
POMap++	**Automatic**	**Terminological , Structural**	**Yes**	**Yes**	**Random forest**

Comparison of Table 2.3 is based on the following criteria: The method for the generation of the training set, the type of the employed features, the use of feature selection methods, the resampling of the training set and the employed machine learning algorithms.

Nezhadi et al. [80] proposed an ontology matching learning approach. Nezhadi et al. [80] claim that the main problem is how to generate a good training set. It should be a highly representative subset of the problem's data. In addition, the larger it is, the higher the quality of the classification will be. An important aspect is the distribution of the classes in the training set. It should reflect the nature of the data to be classified and contain an adequate number of examples for every class [80]. Therefore, training data should be balanced in terms of balancing the number of positive samples and the number of negative samples. We can balance the class distribution using resampling techniques. Consequently, we consider resampling as an important compassion criterion in Table 2.3. Ngo and Bellahsene [82] proposed an ontology matching learning approach based on a decision tree algorithm. Ngo and Bellahsene [82] stated that the performance of the matching learning strongly depends on the training data, i.e., if the training data are not available or not suitable, the matching quality is poor. Hence, we claim that the training set generation method is crucial for the accuracy of matching learning. Therefore, we consider this criterion in comparing the state-of-the-art matching learning systems. Matching learning approaches combine different matchers. Each matcher is considered as a feature of the training data. Hence, we investigate the use of feature selection techniques for the training data. Every matching task has its own specific type of heterogeneity. Therefore, the ontology matching systems should select adequate matchers. This process can be ensured using feature selection techniques. Feature selection is the process of selecting the subset of the most relevant features from the set of features. There are four main reasons for feature selection:

- To simplify the model by reducing the number of parameters.
- To decrease the training time.
- To reduce over-fitting by enhancing generalization.
- To avoid the curse of dimensionality

Hence, we consider feature selection as a criterion to classify the ontology matching learning approaches. Moreover, we investigate the type of the employed features as well as the employed machine learning algorithms by the existing work. Despite the importance of resampling and feature selection, there is a lack of work that considers these two techniques while performing the ontology matching learning process. Moreover, most of the existing work extract the training set manually from the reference alignments. Reference alignments do not usually exist especially while dealing with new matching tasks. Also, the manual generation of the training data is time-consuming and complex in particular for generating training data for large matching tasks, where the training data should be rich by a variety of patterns.

Our approach POMap++ proposes a most generic way to perform the matching learning. We automatically generate a local training set for each sub-matching task. We do not use any reference alignments or user interactions to build the local training sets. Therefore, we are able to include adequate patterns in the training data in order to result in better matching accuracy. We divide a large ontology matching task into a set of local matching tasks aligned independently based on machine learning techniques. Each local machine learning classifier is based on a local training set, which provides adequate matching settings for each sub-matching task. Local training sets are balanced via resampling techniques in order to deliver a better matching accuracy. Moreover, we employ feature selection to identify for each local matching task the appropriate combination of features, which represent the adequate combination of matchers.

2.5 Conclusion

The alignment of large ontologies is considered as one of the most difficult issues to solve in the ontology alignment field. The high heterogeneity of large scale of ontologies impacts the effectiveness and the efficiency of ontology matching systems, which face a difficulty to find adequate correspondence in a large search space. We argue this issue due to the high conceptual heterogeneity of ontologies. Consequently, we have reviewed in Section 2.3 the state of the art ontology matching systems through four criteria: basic matching techniques, scalability techniques, and workflow strategies. This review aims to give an overview of the existing ontology matching systems and to position our approach. We have deduced that existing approaches lack the automation of the alignment of large. We have identified the following approached that deal with large scale ontology matching: Reduction of search space (early pruning and partition-based matching), parallel matching, self-tuning matching and reuse of previous matching results. Partitioning based approaches can be combined with the other large scale matching approaches. Consequently, in Section 2.4, we have reviewed the state-of-the-art related to large scale ontology matching systems, more specifically the systems based on partitioning approaches. We deduced that the existing work for ontology partitioning suffers from a low coverage rate, a manual definition of the number of sub-matching tasks and isolated partitions. Hence, we target these issues by our proposed approach for ontology partitioning.

A single matcher to align large scale ontologies is not enough to result in good matching accuracy. Therefore, ontology matching systems try to combine different matchers in order to result in better accuracy. The manual tuning is error-prone, especially while dealing with different large scale matching tasks. The combination of different matchers should be automated for each large ontology matching task. This automated tuning process can be ensured by machine learning approaches. Consequently, in Section 2.4, we have reviewed the different approaches for ontology matching learning. We have concluded that the existing work lacks the automation of the training data. Training data are usually extracted manually or from reference alignments, which usually do not exist for the majority of the existing matching tasks. Moreover, we have deduced that state-of-the-art approaches do not perform any resampling of the training data or feature selection. Resampling aims to increase the efficiency of the training data. Feature selection methods aim to select adequate matchers.

Our proposed approach is called local matching learning, which combines a fully automated ontology matching to ontology partitioning. The proposed matching learning approach is fully automated compared to the existing work. The ontology partitioning approach copes with earlier stated problematic. Henceforth, in the next chapters, we will introduce the contributions in order to address the issues deduced from the state-of-the-art.

Chapter 3

Ontologies Partitioning Approach

Contents

3.1 Introduction

The alignment of large ontologies is considered as a cumbersome task in the field of ontology matching. The large size of the input ontologies strongly affects the performance of ontology matching systems. Large ontologies usually include a high conceptual heterogeneity. In other words, each ontology developer or researcher build the sane domain ontology but using different conceptual models. The conceptual heterogeneity can impact the effectiveness and efficiency of the ontology matching system. Therefore, it will be more difficult to discover all the mappings between two ontologies. As a result, the accuracy of the resulted alignments by the matching system can be decreased. Moreover, the matching of large ontologies results in a huge search space, where the matching system tries to find the right correspondences. This task is time-consuming, in particular, while employing a set of matchers that should be combined to return the adequate result. Therefore, the efficiency of the matching system will be affected. To sum up, the main issues of the alignment of large biomedical ontologies are the conceptual heterogeneity, the high search space and the decreased quality of the resulted alignments.

In this chapter, we propose a novel ontology partitioning approach in order to align large ontologies matching tasks. This approach divides a large ontology matching task into a set of sub-matching tasks. The large search space is reduced from the whole ontology matching problem to a set of sub-matching problems. Consequently, in the alignment process, we can align effectively each sub-matching task in order to result in better matching accuracy. The proposed partitioning approach is based on a novel multi-cut strategy generating not large partitions or not isolated ones. As a result, we can overcome the issue of conceptual heterogeneity. In Section 3.2, we introduce the formal notations employed in this chapter. Then, we present the proposed ontology partitioning approach, which follows mainly three stages: (i) input ontologies partitioning pre-processing, introduced in Section 3.3, (ii) partitioning algorithm, introduced in Section 3.4 and (iii) identification of local matching tasks is presented in Section 3.5.

3.2 Formal foundations

We introduce definitions that we will employ throughout this chapter.

3.2.1 Ontology and Ontology Alignment

Ththe ontology term is borrowed from philosophy, where an ontology is a systematic account of Existence [40]. For knowledge-based systems, what “exists” is exactly that which can be represented [40]. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects and the describable relationships among them are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names are meant to denote, and formal axioms that constrain the interpretation and well-formed use of these terms [35]. An ontology can have semantic relationships between their classes. Each class can be associated with different names, labels, and comments.

Definition 1 (Ontology) *We encode an ontology $\mathcal{O}_i$ as a directed acyclic graph (DAG), denoted by:*

$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$

such that:

- $\mathcal{V}_i = \{ e_{i,1},...,e_{i,n_i} \}$ *is a finite set of entites of the ontology* $\mathcal{O}_i$, $n_i = \mid \mathcal{V}_i \mid$. *Each entity represent a class that can be associated to different names, labels and comments.*

- $\mathcal{E}_i = \{(e_{i,k},e_{i,l}) \mid e_{i,k},\ e_{i,l} \in \mathcal{V}_i \}$ *is a finite set of edges, where an edge encodes the relationship between two entities of the ontology* $\mathcal{O}_i$. *An edge can refer to the subsumption or an object property.*

An alignment $\mathcal{A}_{ij}$ is the set of correspondences between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$. $\mathcal{A}_{ij}$ can be either retrieved from the set of reference alignment. We state that a single mapping of $\mathcal{A}_{ij}$ 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 6.

Definition 2 (Ontology Alignment) *Ontology Alignment $\mathcal{A}_{ij}$ between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ results in the set of Alignment:*

$\mathcal{A}_{ij} : <e_l,\ e_q,\ n,\ p>$

such that:

- $e_l \in \mathcal{V}_i$ *and* $e_q \in \mathcal{V}_j$

- *n is the matching confidence value between the entities e_l and e_q respectively from $\mathcal{O}_i$ and $\mathcal{O}_j$, where $n \in [0..1]$*

- *p is the semantic relationship between e_l and e_q.*

Moreover, the ontology matching process may need some parameters, which correspond to the matching tuning configuration and some resources. These resources may be the external knowledge resources that can be employed during the matching process.

3.2.2 Local Matching and Ontology Partitioning

In the following, we provide the characteristics of local matching and ontology partitioning.

Definition 3 (Local matching) *The local matching $\mathcal{LM}_{ij}$ between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ is denoted by:*

$\mathcal{LM}_{ij}=\{lm_{ij,1},...,lm_{ij,n}\}$.

A single local matching task $lm_{ij,q}$ correspond to the matching learning between the entities of $\mathcal{V}_{i,k}$ and $\mathcal{V}_{j,l}$, respectively from two ontology partitions $p_{i,k} \subseteq \mathcal{P}_i$ and $p_{j,l} \subseteq \mathcal{P}_j$. Each ontology matching task is aligned based on its specific classifier in order to define the local-based matching settings.

Each ontology is independently partitioned into a set of distinct partitions. Partitions of one ontology do not overlap in terms of entities and semantic relationships. The entities of a single ontology partition belong to the set of entities of its ontology $\mathcal{O}_i$. The relationships between the entities of a single partition are included in the set of relationships of its corresponding ontology $\mathcal{O}_i$.

In the following, we introduce the formal definition of an ontology partition:

Definition 4 (Ontology partition) *An ontology partition $p_{i,k}$ from an ontology $\mathcal{O}_i$ is a sub-ontology denoted by:*

$p_{i,k} = (\mathcal{V}_{i,k}, \mathcal{E}_{i,k})$

such that:

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

The partitioning process of each ontology results in a set of partitions for each ontology. Each partition contains at least one entity. Therefore, we do not result in any isolated partitions. The total number of the relationships and entities of the partitions set is equal to the total number of the relationships and entities of its corresponding ontology. For a given ontology $\mathcal{O}_i$, the entities and the semantic relationships of each partition are distinct from the entities and the semantic relationships of the other partitions. The set of partitions of each ontology are disjoint. In the following, we provide the formal foundation of the cited characteristics of a set of ontology partitions.

Definition 5 (Properties of a set of ontology partitions) *An ontology $\mathcal{O}_i$ can be divided into a set of ontology partitions:*

$\mathcal{P}_i =\{p_{i,1},..,p_{i,s_i}\}$

such that:

- $\forall k \in [1..s_i], \mathcal{V}_{i,k} \neq \emptyset$: *each partition includes at least one entity.*
- $\bigcup_{k=1}^{s_i} \mathcal{V}_{i,k} = \mathcal{V}_i$: *all the entities of an ontology $\mathcal{O}_i$ are covered by the set of partitions.*
- $\forall\ k \in [1..s_i]$, $\forall\ l \in [1..s_i]$, $k \neq l$, $\mathcal{V}_{i,k} \cap \mathcal{V}_{i,l} = \emptyset$: *the partition set are disjoint, there is no partitions that share the same entity or the semantic relationship.*

Definition 6 (Partitioning coverage ratio) *The partitioning coverage ratio of the Local Matching $\mathcal{LM}_{ij}$ defines the set of mappings that can be discovered after performing the partitioning process*

$$\mathcal{C}gRatio(\mathcal{LM}_{ij},\ \mathcal{A}_{ij}) = \frac{|\mathcal{C}g(lm_{ij,q}, \mathcal{A}_{ij}))|}{|\mathcal{A}_{ij}|} \tag{3.1}$$

In the following, we introduce the different steps for the ontology partitioning approach. These steps are depicted as an architecture in Figure 3.1.

3.3 Ontologies Partitioning Architecture

In this section, we present the main steps of the proposed approach for ontology partitioning. In Figure 3.1, we depict the main steps of the partitioning approach. The approached strategy is similar to the divide and conquer method [35]. The divide and conquer method reduces the search space for finding alignments. Therefore the search space is reduced from the Cartesian product of all the entities of the two input ontologies to the Cartesian product of entities of the partition-pairs. A partitions-pair between two ontologies correspond to a local matching task. The partitioning approach splits a big matching problem (i.e., matching large ontologies) into smaller problems (i.e., matching sub-ontologies).

There are two main classes of clustering methods: hierarchic methods and non-hierarchic methods. Hierarchical clustering methods produce a set of clusters nested hierarchical structure. Therefore, hierarchical methods are more flexible than non-hierarchical methods in terms of the relationship between the generated clusters. Hierarchical clustering methods are classified by top-down and bottom-up methods. Despite top-down methods, bottom-up methods do not require any prior information regarding the desired number of partitions. Therefore, the later methods are considered more flexible than top-down clustering methods.

We are based on the hierarchical agglomerative clustering approach while delivering a large ontology into a set of partitions. The hierarchical agglomerative clustering does not need any user involvement [77]. In other words, the number of clusters is not manually defined in advance for each ontology matching task. The hierarchical agglomerative clustering method divides the input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ into a set of partitions, respectively $\mathcal{P}_i$ and $\mathcal{P}_j$. Then, we iteratively identify the set of similar partitions between the two partition sets $\mathcal{P}_i$ and $\mathcal{P}_j$. Similar partitions represent the set of our local matching tasks $\mathcal{LM}_{ij}$. Hierarchical clustering algorithms are either top-down or bottom-up. Bottom-up algorithms treat each entity as a singleton cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all entities. Bottom-up hierarchical clustering is therefore called Hierarchical Agglomerative Clustering or HAC. The HAC approach requires the choice of two main parameters: the linkage strategy and the distance similarity measure. Different linkage criterions have been proposed by the state-of-the-art to measure the distance or dissimilarity between two clusters. Linkage strategies include several strategies such as complete-linkage clustering, single-linkage clustering, average linkage clustering, and Centroid linkage clustering. Depending on the linkage criterion, several variants of hierarchical clustering can be defined; in particular, for the complete-linkage and the single-linkage clustering algorithms the distance between any two clusters $p_{i,l}$ and $p_{i,k}$ are respectively defined as the maximum and minimum Euclidean distance between all pairs of entities in $p_{i,l}$ and $p_{i,k}$. We have evaluated several linkage strategies and our experiments conclude the choice of the average linkage strategy. The average linkage strategy results in a better matching accuracy during the local matching learning process.

The input of this architecture is two indexed ontologies and the output is a set of local matching tasks. Each local matching task is automatically aligned based on machine learning techniques. This alignment process will be detailed in Chapter 4.

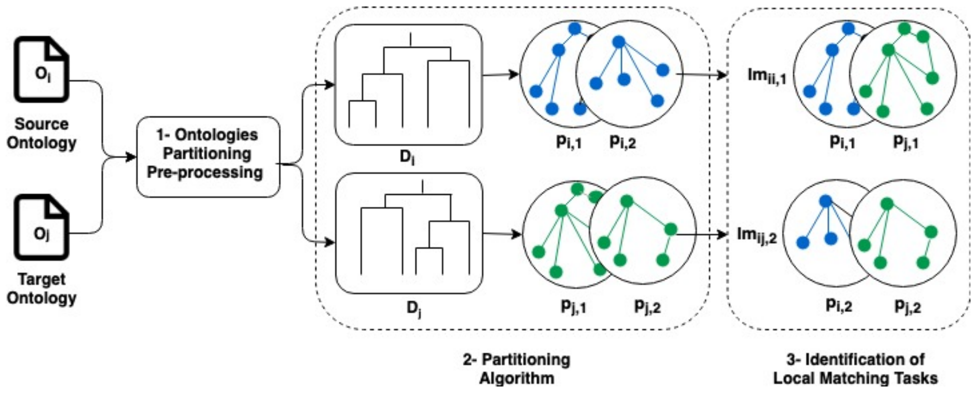

Oi
Source Ontology
Oj
Target Ontology
1- Ontologies Partitioning Pre-processing
Di
Dj
Pi,1
Pi,2
Pj,1
Pj,2
2- Partitioning Algorithm
lmij,1
lmij,2
Pi,1
Pj,1
Pi,2
Pj,2
3- Identification of Local Matching Tasks

Figure 3.1: Ontologies Partitioning Architecture

In the following, we briefly describe the main steps of the partitioning approach. All these steps will be detailed in the next sections of this chapter.

1. **Ontologies partitioning pre-processing**: The input of this step is two indexed ontologies. The output is two pre-processed ontologies. The pre-processing consists of computing the structural relatedness between all the entities of each input ontology. This structural relatedness is required by the hierarchical agglomerative clustering method in order to generate a dendrogram. In our case, the employed similarity measure is the structural relatedness, which will be introduced in the next section. The computation of the structural relatedness requires the existence of a root entity. As a result, we create a novel root entity in case of its absence in each input ontology.

2. **Partitioning algorithm**: The partitioning algorithm has as input two pre-processed ontologies and results in a set of partitions of each ontology. The partitioning algorithm employs the hierarchical agglomerative clustering method. We propose a novel cut strategy of multi-cut strategy in order to result in a set of non-isolated partitions and non-large partitions. Isolated partitions are partitions with only one entity. These isolated partitions decrease matching accuracy result. Therefore, this

issue should be taken into consideration by the partitioning process. Hence, by overcoming this issue, we can increase the matching quality. Large partitions can also affect the accuracy of the matching process. The constructed classifiers for large partitions are less effective than for the smaller ones. For instance, the resulted accuracy of the matching process of two ontologies with one huge partition-pair and one small partition-pair is lower than the matching process of several partitions pairs with no large partition-pair. We also employ the average linkage strategy in order to compute the similarity between two partitions to build the dendrogram of one ontology.

3. **Identification of local matching tasks**: this step has as input the set of partitions of each ontology and results in the set of local matching tasks between the two input ontologies. We identify the set of local matching tasks based on a set of anchors. Anchors are a set of initial correspondences between the two input ontologies. These anchors are employed to identify the set of local matching tasks without being included in the set of final alignments.

In the next sections, we explain in depth the workflow of each step of the partitioning process.

3.4 Input Ontologies Partitioning Pre-processing

In order to measure the similarity between clusters, we use the function *StrcSim* presented in equation 3.2. This function computes the relatedness between the entities of an input ontology. It ensures that the more structurally close two entities, the higher relatedness score they have and the higher the probability that they will belong to the same partition. This formula is inspired by [113] similarity measure.

Definition 7 (Relatedness between entities) *To compute the degree of relatedness between all the entities in one ontology, we measure their structural similarity. Given two entities $e_{i,x}$ and $e_{i,y}$, we have defined **lca** as their lowest common ancestor. We denote the relatedness between entities by:*

$$StrcSim(e_{i,x}, e_{i,y}) = \frac{Dist(r_i, lca) \times 2}{Dist(e_{i,x}, lca) + Dist(e_{i,y}, lca) + Dist(r_i, lca) \times 2} \tag{3.2}$$

Such that:

- *$Dist(r_i,lca)$ is the distance between the root r_i and lca.*
- *$Dist(e_{i,x},lca)$ represents the shortest distance between the entity $e_{i,x}$ and lca in terms of number of edges,*
- *$Dist(e_{i,y},lca)$ denote the distance between the entity $e_{i,y}$ and lca,*

According to this structural similarity measure, when two entities are structurally close in one ontology, they are likely belonging to the same partition. In the case when the root entity (r_i) and the lowest common ancestor (lca) are representing exactly the same entity, we consider the distance $Dist(r_i, lca) = 1$ rather than 0 because the division by 0 is undefined.

In Figure 3.2, we depict the computation of the relatedness between two entities of an ontology $\mathcal{O}_j$. Based on Equation 3.3, the relatedness between the entities $e_{i,9}$ and $e_{i,6}$ gives the following equation:

$$StrcSim(e_{i,9}, e_{i,6}) = \frac{Dist(r_i, lca) \times 2}{Dist(e_{i,9}, lca) + Dist(e_{i,6}, lca) + Dist(r_i, lca) \times 2} \tag{3.3}$$

Such that:

$r_i = e_{i,1}$, lca $= e_{i,7}$, Dist($e_{i,9}$,lca) = 2, Dist($e_{i,6}$,lca) = 1, Dist(r_i,lca) = 2

Consequently, StrcSim ($e_{i,9}$,$e_{i,6}$) = 0.57. We can deduce according to the obtained relatendess score between $e_{i,9}$ and $e_{i,6}$ that these entities are relatively close to each other.

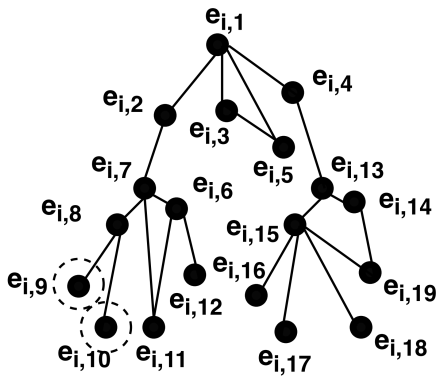


Figure 3.2: Relatedness between entities example

In equation 3.4, we compute the structural relatedness between the entities $e_{i,9}$ and $e_{i,10}$:

$$StrcSim(e_{i,9}, e_{i,10}) = \frac{Dist(r_i, lca) \times 2}{Dist(e_{i,9}, lca) + Dist(e_{i,10}, lca) + Dist(r_i, lca) \times 2} \tag{3.4}$$

Such that: $r_i = e_{i,1}$, lca $= e_{i,8}$, Dist($e_{i,9}$,lca) = 1, Dist($e_{i,10}$,lca) = 1, Dist(r_i,lca) = 3. As a result, StrcSim($e_{i,9}$,$e_{i,10}$) = 0.75. In this example, $e_{i,9}$ and $e_{i,10}$ are more structurally related than $e_{i,9}$ and $e_{i,6}$ since StrcSim ($e_{i,9}$,$e_{i,6}$) < StrcSim ($e_{i,9}$,$e_{i,10}$). Therefore, $e_{i,9}$ and $e_{i,10}$ are more probably to belong to the same cluster of the dendrogram $\mathcal{D}$ representing the ontology $\mathcal{O}_i$.

Particular use case: Some ontologies do not contain any root element. As a result, it is not possible to apply the relatedness equation. Therefore, we search for all high-level entities. These entities do not have any semantic relationship with an ancestor entity. The identified high-level entities of an ontology are linked to a newly created root entity using a subsumption relationship. The ontology depicted in Figure 3.3 (a) do not have any root entity. As a result, we create a root entity $e_{i,0}$. This root entity is connected with a subsumption relationship to the entities $e_{i,1}$ and $e_{i,2}$. Consequently, it is possible to apply the relatedness equation over the ontology in order to generate its corresponding dendrogram.

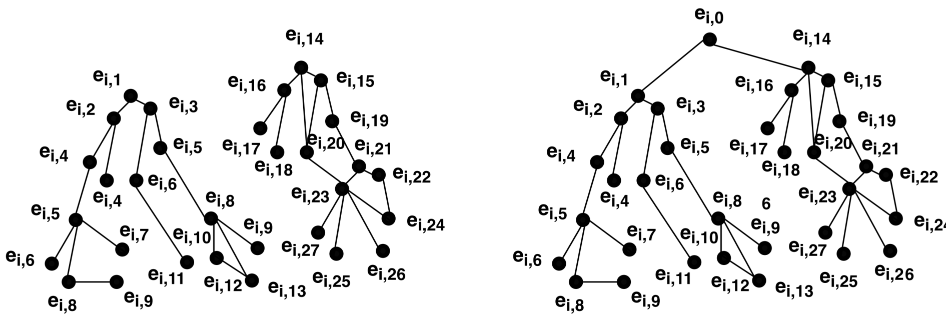


Figure 3.3: (a) An ontology with no root entity (b) The ontology after adding the root entity

3.5 Partitioning Algorithm

In this section, we describe the different steps required for partitioning an input ontology $\mathcal{O}q$. These steps are depicted in Algorithm 1. For each input ontology, the goal of this algorithm is the construction of a dendrogram and to perform its multi-cut in order to result in a set of partitions.

Algorithm 1 takes as input the ontology $\mathcal{O}_i$. The partitioning algorithm splits an input ontology $\mathcal{O}_i$ into a set of partitions $\mathcal{P}_i$. The different variables of Algorithm 1 (line 2) are initialized such as the *Dendrogram* $\mathcal{D}$ and the set of *Partitions* $\mathcal{P}_i$. The *Dendrogram* $\mathcal{D}$ and $\mathcal{P}_i$ are initially initialized to an empty set. These empty sets will be filled in the following parts of Algorithm 1. The *Dendrogram* $\mathcal{D}$ represents the hierarchical relationship between the partitions of an ontology $\mathcal{O}_i$. In the following, we discuss the steps of the partitioning algorithm:

3.5.1 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.

3.5.2 Dendrogram Construction

The agglomerative hierarchical clustering algorithm available in this program module builds a cluster hierarchy that is commonly displayed as a tree diagram called a dendrogram. They begin with each partition in a separate partition. At each step, the two

Algorithm 1 Ontology partitioning Algorithm

```
1: Input $\mathcal{O}_i$= ( $\mathcal{V}_i$, $\mathcal{E}_i$), $SplitSize$                ▷ (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:        $\mathcal{P}_i \leftarrow \mathcal{P}_i \bigcup getIterativeCutPartitions(\mathcal{D}, \mathcal{P}_{i,l})$
18:    else
19:        $\mathcal{P}_i \leftarrow \mathcal{P}_i \bigcup \{p_{i,l}\}$
20:    end if
21: end for
22: Return $\mathcal{P}_i$                ▷ Output
```

partitions that are most similar are joined into a single new partition. As depicted in Figure 3.4, the horizontal axis of the dendrogram represents the distance or dissimilarity between partitions. The vertical axis represents the entities and partitions. The advantage of a dendrogram is its simple interpretation.

The Dendrogram $\mathcal{D}$ of each input ontology is generated by the hierarchical agglomerative approach hac (line 7 to 12). This approach iteratively identify the two partitions $(p_{i,l}, p_{i,k})$ from $\mathcal{P}_i$ with the maximum structural similarity (line 8). The maximum structural similarity is measured using the relatedness formula (Definition 7) coupled 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). Then, we merge these partitions $p_{i,l}$ and $p_{i,k}$ into a new partition containing $(p_{i,l}, p_{i,k})$, whish is added to the set of partitions $\mathcal{P}_i$ (line 9). Every merged partitions are added to the Dendrogram hierarchy $\mathcal{D}$ (line 10). The instractions from line 7 to line 12 are repeated until building a dendrogram $\mathcal{D}$ with a number $\mathcal{L}v$ of levels, which is equal to the number of entities $\mathcal{V}_i$ of $\mathcal{O}_i$.

3.5.3 Dendrogram multi-cut

Agglomerative hierarchical clustering can be represented by a dendrogram. The dendrogram should be at a certain level results in a set of partitions. Cutting at another level results in another set of partitions. To obtain a given partitioning of the data, the dendrogram has to be cut at a certain height. Entities that remain interconnected after the cut will be considered part of the same partition.

The generated dendrogram hierarchy $\mathcal{D}$ should be cut at a certain level to result in a set of non-isolated partitions $\mathcal{P}_i$. There is no standard solution that can result in a perfect dendrogram cut for every application domain. Every application domain has its own preferred cut strategy. Dealing with the hierarchical agglomerative clustering of large ontologies, there is a lack of work that proposes the adequate cut strategy for the generated dendrogram.

Partitions with only one entity are considered as isolated. These isolated partitions negatively impact the matching accuracy result. For instance, the partitioning of FMA-SNOMED matching task of OAEI using Falcon Falcon results in 3352 isolated partitions with an F-Measure 0.485 ernesto.

We depict in Figure 3.4 an example of a normal cut of a dendrogram $\mathcal{D}$ representing an ontology $\mathcal{O}_i$. This ontology is encoded as a directed acyclic graph, where:

$\mathcal{O}_i = (\mathcal{V}_i, \mathcal{E}_i)$

such that:

$\mathcal{V}_i = \{ e_{i,1}, e_{i,2}, e_{i,3}, e_{i,4}, e_{i,5}, e_{i,6}, e_{i,7}, e_{i,8}, e_{i,9}, e_{i,10}, e_{i,11}, e_{i,12}, e_{i,13}, e_{i,14}, e_{i,15}, e_{i,16} \}$

The performed cut results in a set of isolated partitions. This random cut generates the following partitions set $\mathcal{P}_i$ for the ontology $\mathcal{O}_i$:

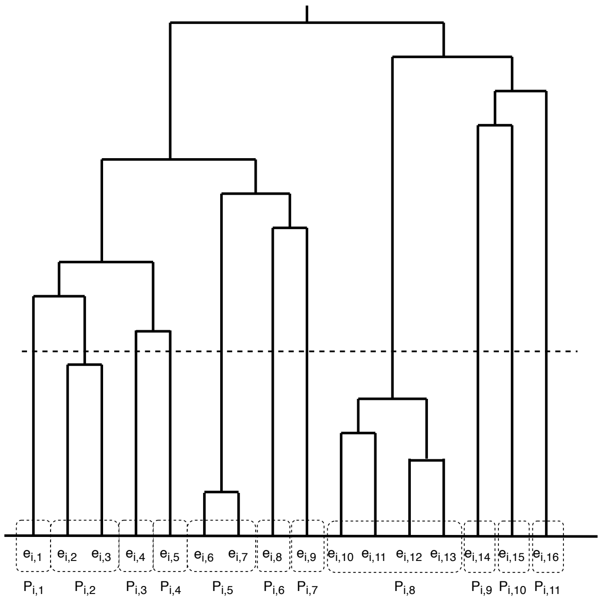
$\mathcal{P}_i = \{p_{i,1}, p_{i,2}, p_{i,3}, p_{i,4}, p_{i,5}, p_{i,6}, p_{i,7}, p_{i,8}, p_{i,9}, p_{i,10}, p_{i,11}\}$

Such that

- $p_{i,1} = (\mathcal{V}_{i,1}, \mathcal{E}_{i,1})$, where $\mathcal{V}_{i,1} = \{e_{i,1,1}\}$, $\mathcal{V}_{i,1} \subseteq \mathcal{V}_i$
- $p_{i,2} = (\mathcal{V}_{i,2}, \mathcal{E}_{i,2})$, where $\mathcal{V}_{i,2} = \{e_{i,2,2}, e_{i,2,3}\}$, $\mathcal{V}_{i,2} \subseteq \mathcal{V}_i$
- $p_{i,3} = (\mathcal{V}_{i,3}, \mathcal{E}_{i,3})$, where $\mathcal{V}_{i,3} = \{e_{i,3,4}\}$, $\mathcal{V}_{i,3} \subseteq \mathcal{V}_i$
- $p_{i,4} = (\mathcal{V}_{i,4}, \mathcal{E}_{i,4})$, where $\mathcal{V}_{i,4} = \{e_{i,4,5}\}$, $\mathcal{V}_{i,4} \subseteq \mathcal{V}_i$
- $p_{i,5} = (\mathcal{V}_{i,5}, \mathcal{E}_{i,5})$, where $\mathcal{V}_{i,5} = \{e_{i,5,6}, e_{i,5,7}\}$, $\mathcal{V}_{i,5} \subseteq \mathcal{V}_i$
- $p_{i,6} = (\mathcal{V}_{i,6}, \mathcal{E}_{i,6})$, where $\mathcal{V}_{i,6} = \{e_{i,6,8}\}$, $\mathcal{V}_{i,6} \subseteq \mathcal{V}_i$
- $p_{i,7} = (\mathcal{V}_{i,7}, \mathcal{E}_{i,7})$, where $\mathcal{V}_{i,7} = \{e_{i,7,9}\}$, $\mathcal{V}_{i,7} \subseteq \mathcal{V}_i$
- $p_{i,8} = (\mathcal{V}_{i,8}, \mathcal{E}_{i,8})$, where $\mathcal{V}_{i,8} = \{e_{i,8,10}, e_{i,8,11}, e_{i,8,12}, e_{i,8,13}\}$, $\mathcal{V}_{i,8} \subseteq \mathcal{V}_i$
- $p_{i,9} = (\mathcal{V}_{i,9}, \mathcal{E}_{i,9})$, where $\mathcal{V}_{i,9} = \{e_{i,9,14}\}$, $\mathcal{V}_{i,9} \subseteq \mathcal{V}_i$
- $p_{i,10} = (\mathcal{V}_{i,10}, \mathcal{E}_{i,10})$, where $\mathcal{V}_{i,10} = \{e_{i,10,15}\}$, $\mathcal{V}_{i,10} \subseteq \mathcal{V}_i$

- $p_{i,11} = (\mathcal{V}_{i,11},\mathcal{E}_{i,11})$, where $\mathcal{V}_{i,11} = \{e_{i,11,16}\}$, $\mathcal{V}_{i,11} \subseteq \mathcal{V}_i$

The performed random cut generated 6 isolated partitions ($p_{i,1}$, $p_{i,3}$, $p_{i,4}$, $p_{i,6}$, $p_{i,7}$, $p_{i,9}$, $p_{i,10}$ and $p_{i,11}$).



e_{i,1} e_{i,2} e_{i,3} e_{i,4} e_{i,5} e_{i,6} e_{i,7} e_{i,8} e_{i,9} e_{i,10} e_{i,11} e_{i,12} e_{i,13} e_{i,14} e_{i,15} e_{i,16}
P_{i,1} P_{i,2} P_{i,3} P_{i,4} P_{i,5} P_{i,6} P_{i,7} P_{i,8} P_{i,9} P_{i,10} P_{i,11}

Figure 3.4: Isolated partitions cut example

Moreover, a single cut of a dendrogram $\mathcal{D}$ can result in a set of large partitions. In order to demonstrate this use case, we perform another random cut of the same dendrogram $\mathcal{D}$. This random cut is depicted in Figure 3.5. The later cut results in the set of the following partition set $\mathcal{P}_i$ for the ontology $\mathcal{O}_i$:

$\mathcal{P}_i = \{ p_{i,1}, p_{i,2}, p_{i,3}\}$

Such that

- $p_{i,1} = (\mathcal{V}_{i,1},\mathcal{E}_{i,1})$, where $\mathcal{V}_{i,1} = \{e_{i,1,1}, e_{i,1,2}, e_{i,1,3}, e_{i,1,4}, e_{i,1,5}, e_{i,1,6}, e_{i,1,7}, e_{i,1,8}, e_{i,1,9}\}$, $\mathcal{V}_{i,1} \subseteq \mathcal{V}_i$

- $p_{i,2} = (\mathcal{V}_{i,2},\mathcal{E}_{i,2})$, where $\mathcal{V}_{i,2} = \{e_{i,2,10}, e_{i,2,11}, e_{i,2,12}, e_{i,2,13}\}$, $\mathcal{V}_{i,2} \subseteq \mathcal{V}_i$

- $p_{i,3} = (\mathcal{V}_{i,3},\mathcal{E}_{i,3})$, where $\mathcal{V}_{i,3} = \{e_{i,3,14}, e_{i,3,15}, e_{i,3,16}\}$, $\mathcal{V}_{i,3} \subseteq \mathcal{V}_i$

We can deduce from the generated partitions that the first partition $p_{i,1}$ is relatively large compared to the total size of the ontology $\mathcal{O}_i$. Therefore, the partitioning process is not well balanced and the search space is not well reduced.

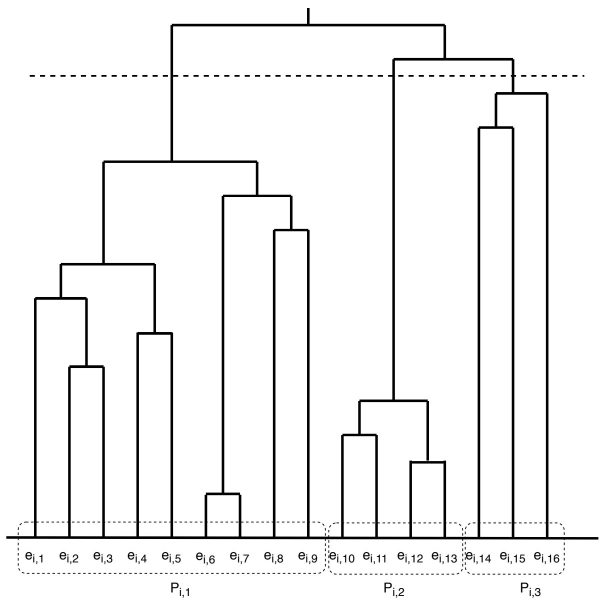

$e_{i,1}$ $e_{i,2}$ $e_{i,3}$ $e_{i,4}$ $e_{i,5}$ $e_{i,6}$ $e_{i,7}$ $e_{i,8}$ $e_{i,9}$
$e_{i,10}$ $e_{i,11}$ $e_{i,12}$ $e_{i,13}$
$e_{i,14}$ $e_{i,15}$ $e_{i,16}$
$P_{i,1}$
$P_{i,2}$
$P_{i,3}$

Figure 3.5: Large partitions cut example

To cope with the issue of isolated partitions and large partitions, we propose a multi-cut strategy of the resulted dendrogram $\mathcal{D}$ (line 13 to 21) in Algorithm 1. Therefore, the non-isolated partitions $\mathcal{P}_i$ 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 certain level of the dendrogram $\mathcal{D}$, which do not result in any isolated partitions. To perform this cut, we conduct all the possible cuts over the dendrogram $\mathcal{D}_i$ until finding the first cut returning a set of non-isolated partitions $\mathcal{P}_{init}$. This initial cut results 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). After performing a set of experiments, we set the SplitSize as 50% of the size of an input ontology $\mathcal{O}_i$. Therefore, a large partition is identified if it contains more than 50% of the entities of an ontology. We iteratively compare the size of the initially generated partitions $\mathcal{P}_{init}$ to the SplitSize (line 16). If an initial partition $p_{i,l}$ is large (line 16), we split this partition $p_{i,l}$ into smaller ones. These partitions are split based on the same strategy as 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 $\mathcal{O}_i$.

We draw in Figure 3.6 an example of a dendrogram $\mathcal{D}$ in order to demonstrate the multi-cut strategy. This dendrogram is resulted by the hierarchical clustering of an ontology $\mathcal{O}_i$ composed of 16 entities. The initial cut is represented by the top dashed line. The multi-cut strategy is performed after a set of iterations. The first iteration corresponds to an initial cut of the dendrogram $\mathcal{D}$. The following iteration of the first one represents the cut process of the resulted partitions.

For this initial iteration, we adopt a SplitSize of 10 entities, which is approximately equal to 50% of the total number of entities of the input ontology $\mathcal{O}_i$. Therefore, the dendrogram $\mathcal{D}$ should be cut in a level that should not result in isolated partition while the size of the resulted partition should be below the SplitSize. This cut result in a set of non-isolated partitions $\mathcal{P}_{init}$:

$\mathcal{P}_{init} = \{ p_{i,1}, p_{i,2}\}$, such that

- $p_{i,1} = (\mathcal{V}_{i,1},\mathcal{E}_{i,1})$, where $\mathcal{V}_{i,1} =\{e_{i,1,1}, e_{i,1,2}, e_{i,1,3}, e_{i,1,4}, e_{i,1,5}, e_{i,1,6}, e_{i,1,7}, e_{i,1,8}, e_{i,1,9}\}$, $\mathcal{V}_{i,1} \subseteq \mathcal{V}_i$

- $p_{i,2} = (\mathcal{V}_{i,2},\mathcal{E}_{i,2})$, where $\mathcal{V}_{i,2} =\{e_{i,2,10}, e_{i,2,11}, e_{i,2,12}, e_{i,2,13}, e_{i,2,14}, e_{i,2,15}, e_{i,2,16}\}$, $\mathcal{V}_{i,2} \subseteq \mathcal{V}_i$

Then, we perform a second cut of the denrogram $\mathcal{D}$. The next cut is performed over the resulted partitions of $\mathcal{P}_{init}$. Partition $p_{i,2}$ of $\mathcal{P}_{init}$ is not considered as a large partition since it contains a number of entities less than the SplitSize, which is 50% of the total number of entities of the input ontology $\mathcal{O}_i$. However, the total number of entities of $p_{i,1}$ is over the SplitSize. Therefore, the next cut is only performed over the partition $p_{i,1}$ of $\mathcal{P}_{init}$. This second cut result in two new partitions. Therefore, the partition set $\mathcal{P}_{init}$ is updated and contains the following partitions :

$\mathcal{P}_{init} = \{ p_{i,1}, p_{i,2}, p_{i,3}\}$

Such that:

- $p_{i,1} = (\mathcal{V}_{i,1},\mathcal{E}_{i,1})$, where $\mathcal{V}_{i,1} =\{e_{i,1,1}, e_{i,1,2}, e_{i,1,3}, e_{i,1,4}, e_{i,1,5}\}$, $\mathcal{V}_{i,1} \subseteq \mathcal{V}_i$

- $p_{i,2} = (\mathcal{V}_{i,2},\mathcal{E}_{i,2})$, where $\mathcal{V}_{i,2} =\{e_{i,2,6}, e_{i,2,7}, e_{i,2,8}, e_{i,2,9}\}$, $\mathcal{V}_{i,2} \subseteq \mathcal{V}_i$

- $p_{i,3} = (\mathcal{V}_{i,3},\mathcal{E}_{i,3})$, where $\mathcal{V}_{i,6} =\{e_{i,3,10}, e_{i,3,11}, e_{i,3,12}, e_{i,3,13}, e_{i,3,14}, e_{i,3,15}, e_{i,3,16}\}$, $\mathcal{V}_{i,3} \subseteq \mathcal{V}_i$

There is no additional cut of the dendrogram since the size of all the resulted partitions is under the SplitSize. Consequently, the resulted partitions of $\mathcal{P}_{init}$ are not large and without any isolated partitions.

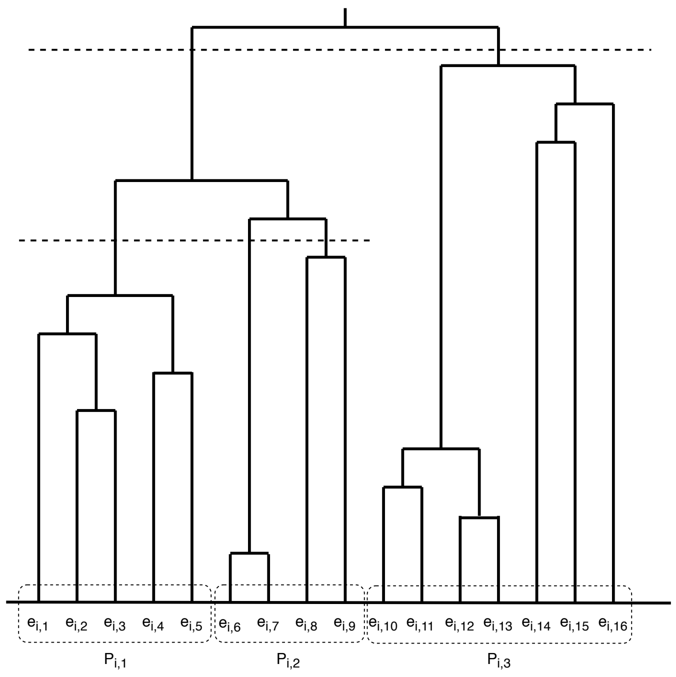

$e_{i,1}$ $e_{i,2}$ $e_{i,3}$ $e_{i,4}$ $e_{i,5}$ $e_{i,6}$ $e_{i,7}$ $e_{i,8}$ $e_{i,9}$ $e_{i,10}$ $e_{i,11}$ $e_{i,12}$ $e_{i,13}$ $e_{i,14}$ $e_{i,15}$ $e_{i,16}$
$P_{i,1}$
$P_{i,2}$
$P_{i,3}$

Figure 3.6: Multi-cut example

In the next step, we show how to determine the local matching tasks $\mathcal{LM}_{ij}$ between two sets of ontology partitions $\mathcal{P}_i$ and $\mathcal{P}_j$.

3.6 Identification of local matching tasks

Algorithm 2 presents the different steps in order to generate a set of local matching tasks from the set of partitions generated by Algorithm 1. Algorithm 2 takes as input the two sets of partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ resulted by the partitioning of the input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ coupled with a set of anchors $\mathcal{A}_{ij}$ (line 1). Algorithm 2 results in the set of local matching tasks $\mathcal{LM}_{ij}$.

The anchors are denoted as:

$\mathcal{A}_{ij} = \{(\mathrm{e}_{i,x},\mathrm{e}_{j,y}) \mid \mathrm{e}_{i,x} \in \mathcal{V}_i,\ \mathrm{e}_{j,y} \in \mathcal{V}_j\ \}$.

The anchors $\mathcal{A}_{ij}$ are employed to identify the set of local matching tasks $\mathcal{LM}_{ij}$. Since we are aligning ontologies, anchors are extracted by cross-searching the two input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ with the available external knowledge bases (KB) such as the Unified Medical Language System (UMLS) Metathesaurus UMLS, Medical Subject Headings (MeSH) mesh, Uberon UBERON, and BioPortal BioPortal in the biomedical donain. For instance, UMLS integrates more than 160 biomedical ontologies. We expand the extracted anchors with a set of exact mappings between the two input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ using a hash-based search method [37]. One partition $p_{i,k}$ of $\mathcal{P}_i$ can have multiple anchors with different partitions of $\mathcal{P}_j$. Similarly, one partition $p_{j,k}$ of $\mathcal{P}_j$ can have multiple anchors with different partitions of $\mathcal{P}_i$. Consequently, we iteratively merge the partitions of $\mathcal{P}_i$ having anchors with more than one partition of $\mathcal{P}_j$ (line 3 to 6). Similarly, we merge the partitions of $\mathcal{P}_j$ having anchors with partitions of $\mathcal{P}_i$ (line 7 to 10). Therefore, we guarantee that there is no anchors overlap between the partitions of $\mathcal{P}_i$ and $\mathcal{P}_j$ respectively from the input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$. Consequently, a local matching task $lm_{ij,q}$ correspond to a single partition of $\mathcal{P}_i$ having anchors only with a single partition of $\mathcal{P}_j$. Therefore, we are able to identify the set of local matching tasks $\mathcal{LM}_{ij}$ between the set of 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. The coverage ratio defines the percentage of reference mappings that can be discovered after performing the partitioning process.

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 P$_j$ **do**                         ▷ $\mathcal{P}_i$ partitions merging
4:     $Q_i = \bigcup_{\forall(e_{ikx},e_{jly})\in A_{ij}} p_i,_k$
5:     $\mathcal{P}_i \leftarrow \mathcal{P}_i \setminus \mathcal{Q}_i \;+\; Merge(\mathcal{Q}_i)$
6: **end for**
7: **for each** $p_{i,k}$ of P$_i$ **do**                         ▷ $\mathcal{P}_j$ partitions merging
8:     $\mathcal{Q}_j = \bigcup_{\forall(e_{ikx},e_{jly})\in \mathcal{A}_{ij}} p_j,_l$
9:     $\mathcal{P}_j \leftarrow \mathcal{P}_j \setminus \mathcal{Q}_j \;+\; Merge(\mathcal{Q}_j)$
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:     **for each** $p_{j,l}$ of $\mathcal{P}_j$ **do**
13:         **if** $(\exists(e_{ikx}, e_{jly}) \in \mathcal{A}_{ij})$ **then**
14:             $\mathcal{LM}_{ij} \leftarrow \mathcal{LM}_{ij} \bigcup \{(p_{i,k}, p_{j,l})\}$
15:         **end if**
16:     **end for**
17: **end for**
18: **Return** $\mathcal{LM}_{ij}$                                  ▷ Output
```

Figure 3.7 gives an example of the proposed approach for partitions merging. As depicted in the example of Figure 3.7, two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ are partitioned into 6 partitions resulted from Algorithm 1. Each partition contains a set of entities illustrated as nodes. The dashed lines represent the set of anchors $\mathcal{A}_{ij}$ between the partitions of the two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$. The first ontology $\mathcal{O}_i$ is encoded as a directed acyclic graph with 36 nodes, where:

$\mathcal{O}_i$= $(\mathcal{V}_i$,$\mathcal{E}_i)$

Such that

$\mathcal{V}_i = \{\ e_{i,1},\ e_{i,2},\ e_{i,3},\ e_{i,4},\ e_{i,5},\ e_{i,6},\ e_{i,7},\ e_{i,8},\ e_{i,9},\ e_{i,10},\ e_{i,11},\ e_{i,12},\ e_{i,13},\ e_{i,14},\ e_{i,15},\ e_{i,16},\ e_{i,17},\ e_{i,18},\ e_{i,19},\ e_{i,20},\ e_{i,21},\ e_{i,22},\ e_{i,23},\ e_{i,24},\ e_{i,25},\ e_{i,26},\ e_{i,27},\ e_{i,28},\ e_{i,29},\ e_{i,30},\ e_{i,31},\ e_{i,32},\ e_{i,33},\ e_{i,34},\ e_{i,35},\ e_{i,36}\}$

The second ontology $\mathcal{O}_j$ is also encoded as a directed acyclic graph with 36 nodes, where:

$\mathcal{O}_j = (\mathcal{V}_j, \mathcal{E}_j)$

such that:

$\mathcal{V}_i = \{ e_{j,1}, e_{j,2}, e_{j,3}, e_{j,4}, e_{j,5}, e_{j,6}, e_{j,7}, e_{j,8}, e_{j,9}, e_{j,10}, e_{j,11}, e_{j,12}, e_{j,13}, e_{j,14}, e_{j,15}, e_{j,16}, e_{j,17}, e_{j,18}, e_{j,19}, e_{j,20}, e_{j,21}, e_{j,22}, e_{j,23}, e_{j,24}, e_{j,25}, e_{j,26}, e_{j,27}, e_{j,28}, e_{j,29}, e_{j,30}, e_{j,31}, e_{j,32}, e_{j,33}, e_{j,34}, e_{j,35}, e_{j,36}\}$

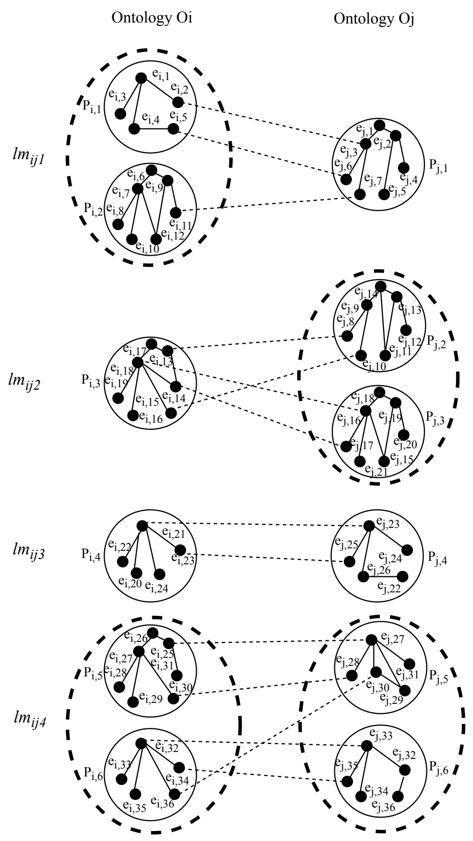


Figure 3.7: Partitions merging example

After partitioning the ontology $\mathcal{O}_i$, the set of partitions $\mathcal{P}_i$ resulted by Algorithm 1 is depicted in Figure 3.5 and denoted by $\mathcal{P}_i$, where:

$\mathcal{P}_i = \{ p_{i,1}, p_{i,2}, p_{i,3}, p_{i,4}, p_{i,5}, p_{i,6}\}$

such that

- $p_{i,1} = (\mathcal{V}_{i,1}, \mathcal{E}_{i,1})$, where $\mathcal{V}_{i,1} = \{e_{i,1,1}, e_{i,1,2}, e_{i,1,3}, e_{i,1,4}, e_{i,1,5}\}$, $\mathcal{V}_{i,1} \subseteq \mathcal{V}_i$
- $p_{i,2} = (\mathcal{V}_{i,2}, \mathcal{E}_{i,2})$, where $\mathcal{V}_{i,2} = \{e_{i,2,6}, e_{i,2,7}, e_{i,2,8}, e_{i,2,9}, e_{i,2,10}, e_{i,2,11}, e_{i,2,12}\}$, $\mathcal{V}_{i,2} \subseteq \mathcal{V}_i$
- $p_{i,3} = (\mathcal{V}_{i,3}, \mathcal{E}_{i,3})$, where $\mathcal{V}_{i,3} = \{e_{i,3,13}, e_{i,3,14}, e_{i,3,15}, e_{i,3,16}, e_{i,3,17}, e_{i,3,18}, e_{i,3,19} \}$, $\mathcal{V}_{i,3} \subseteq \mathcal{V}_i$
- $p_{i,4} = (\mathcal{V}_{i,4}, \mathcal{E}_{i,4})$, where $\mathcal{V}_{i,4} = \{e_{i,4,20}, e_{i,4,21}, e_{i,4,22}, e_{i,4,23}, e_{i,4,24}\}$, $\mathcal{V}_{i,4} \subseteq \mathcal{V}_i$
- $p_{i,5} = (\mathcal{V}_{i,5}, \mathcal{E}_{i,5})$, where $\mathcal{V}_{i,5} = \{e_{i,5,25}, e_{i,5,26}, e_{i,5,27}, e_{i,5,28}, e_{i,5,29}, e_{i,5,30}, e_{i,5,31}\}$, $\mathcal{V}_{i,5} \subseteq \mathcal{V}_i$

- $p_{i,6}$ = ($\mathcal{V}_{i,6}$,$\mathcal{E}_{i,6}$), where $\mathcal{V}_{i,6}$ = {$e_{i,6,32}$, $e_{i,6,33}$, $e_{i,6,34}$, $e_{i,6,35}$, $e_{i,6,36}$}, $\mathcal{V}_{i,6} \subseteq \mathcal{V}_i$

The partitioning of the ontology $\mathcal{O}_j$ results in the set of partitions $\mathcal{P}_j$ after applying Algorithm 1. This set of partitions is depicted in Figure 3.5 and denoted by $\mathcal{P}_j$, where:
$\mathcal{P}_j$ = { $p_{j,1}$, $p_{j,2}$, $p_{j,3}$, $p_{j,4}$, $p_{j,5}$, $p_{j,6}$}
such that

- $p_{j,1}$ = ($\mathcal{V}_{j,1}$,$\mathcal{E}_{j,1}$), where $\mathcal{V}_{j,1}$ ={$e_{j,1,1}$, $e_{j,1,2}$, $e_{j,1,3}$, $e_{j,1,4}$, $e_{j,1,5}$, $e_{j,2,6}$, $e_{j,2,7}$}, $\mathcal{V}_{j,1} \subseteq \mathcal{V}_j$
- $p_{j,2}$ = ($\mathcal{V}_{j,2}$,$\mathcal{E}_{j,2}$), where $\mathcal{V}_{j,2}$ = {$e_{j,2,8}$, $e_{j,2,9}$, $e_{j,2,10}$, $e_{j,2,11}$, $e_{j,2,12}$, $e_{j,3,13}$, $e_{j,3,14}$}, $\mathcal{V}_{j,2} \subseteq \mathcal{V}_j$
- $p_{j,3}$ = ($\mathcal{V}_{j,3}$,$\mathcal{E}_{j,3}$), where $\mathcal{V}_{j,3}$ ={ $e_{j,3,15}$, $e_{j,3,16}$, $e_{j,3,17}$, $e_{j,3,18}$, $e_{j,3,19}$, $e_{j,4,20}$, $e_{j,4,21}$ }, $\mathcal{V}_{j,3} \subseteq \mathcal{V}_j$
- $p_{j,4}$ = ($\mathcal{V}_{j,4}$,$\mathcal{E}_{j,4}$), where $\mathcal{V}_{j,4}$ ={$e_{j,4,22}$, $e_{j,4,23}$, $e_{j,4,24}$, $e_{j,5,25}$, $e_{j,5,26}$}, $\mathcal{V}_{j,4} \subseteq \mathcal{V}_j$
- $p_{j,5}$ = ($\mathcal{V}_{j,5}$,$\mathcal{E}_{j,5}$), where $\mathcal{V}_{j,5}$ = {$e_{j,5,27}$, $e_{j,5,28}$, $e_{j,5,29}$, $e_{j,5,30}$, $e_{j,5,31}$}, $\mathcal{V}_{j,5} \subseteq \mathcal{V}_j$
- $p_{j,6}$ = ($\mathcal{V}_{j,6}$,$\mathcal{E}_{j,6}$), where $\mathcal{V}_{j,6}$ = {$e_{j,6,32}$, $e_{j,6,33}$, $e_{j,6,34}$, $e_{j,6,35}$, $e_{j,6,36}$}, $\mathcal{V}_{j,6} \subseteq \mathcal{V}_j$

Following Algorithm 2 (line 11 to line 17), we iteratively merge the ontology partitions of the ontology $\mathcal{O}_i$ having anchors with one partition of the ontology $\mathcal{O}_j$. We also follow the same iterative strategy in order to merge the ontology $\mathcal{O}_j$ having anchors with one partition of the ontology $\mathcal{O}_i$. For instance, the ontology partitions $p_{i,1}$ and $p_{i,2}$ of the ontology $\mathcal{O}_i$ are merged into one partition since they both have anchors to the partition $p_{j,1}$ of the ontology $\mathcal{O}_j$. As depicted in Figure 3.7, the extracted Anchors between the two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ are denoted by $\mathcal{A}_{ij}$, where:

$\mathcal{A}_{ij}$ = {(e$_{i,2}$,e$_{j,3}$), (e$_{i,5}$,e$_{j,7}$), (e$_{i,11}$,e$_{j,6}$), (e$_{i,13}$,e$_{j,8}$), (e$_{i,14}$,e$_{j,10}$), (e$_{i,16}$,e$_{j,16}$), (e$_{i,18}$,e$_{j,17}$), (e$_{i,21}$,e$_{j,23}$), (e$_{i,23}$,e$_{j,25}$),(e$_{i,25}$,e$_{j,30}$), (e$_{i,30}$,e$_{j,27}$), (e$_{i,32}$,e$_{j,28}$), (e$_{i,34}$,e$_{j,35}$), (e$_{i,36}$,e$_{j,33}$)}, e$_{i,x} \in \mathcal{V}_i$, e$_{j,y} \in \mathcal{V}_j$ }.

The merge of partitions is illustrated in Figure 3.7 by a hashed circle. After performing the merge process, we are able to identify the set of local matching tasks $\mathcal{LM}_{ij}$ between the two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$. For instance, in Figure 3.7, we identify the following four local matching tasks:

$\mathcal{LM}_{ij}$={$lm_{ij,1}$, $lm_{ij,2}$, $lm_{ij,3}$, $lm_{ij,4}$}.

3.7 Conclusion

The matching of large ontologies is a complex task especially while employing a set of matchers. Consequently, the matching accuracy of matching systems is decreased. Large ontologies have a highly conceptual heterogeneity. Moreover, large ontologies produce a huge search space in which matching systems. In this chapter, we have proposed a novel approach to partition a large ontology matching task into a set of sub-matching tasks called local matching tasks. A big matching problem is splitted into a set of smaller problems. Therefore, the ontology matching process is only performed between the set of local matching tasks rather than the whole ontologies. This divide and conquer strategy aims to reduce the search space for the discovery of alignments in large ontologies. Differentiated from the state-of-the-art approaches, the introduced partitioning approach does not result in any isolated partitions or large sub-matching tasks.

The proposed partitioning approach is based on the hierarchical agglomerative clustering method and follows three stages: (i) input ontologies partitioning pre-processing (ii) partitioning algorithm and (iii) identification of local matching tasks. During the first stage, we construct the dendrogram for each input ontology. This dendrogram is constructed based on the relatedness of equation 3.1 and the average linkage strategy. In the second stage, the dendrogram for each ontology is splitted into a set of partitions based on a novel multi-cut strategy. This multi-cut strategy does not result in any large partitions. In the third stage, we identify the set of local matching tasks based on novel partitions merging approach. In the next chapter, we introduce the local matching learning approach in order to automatically align the set of local matching tasks.

Chapter 4

Local Matching Learning Approach

Contents

4.1 Introduction

Using a single matcher for the whole ontology matching task is usually insufficient. Therefore, an automatic combination of these matchers and the tuning of each matcher is essential for a better matching quality. Hence, the settings of an ontology matching system should not depend on a set of pre-selected contexts. The matching settings should be automatically adapted for every new matching context, which can not be included in the predefined set of settings. The main issue concerns the choice of the matching settings (eg. the weight of each matcher, the threshold) while dealing with different matching contexts. This choice should be automated in order to reduce the matching process complexity, especially while dealing with large scale ontologies. Thus, the matching process should be self-tuned in order to derive the adequate matching settings,e.g. similarity measure and threshold, for each matching context. This process can improve the ontology matching accuracy.

The goal of local matching learning is to perform an effective matching of large ontologies. This matching is based on machine learning techniques in order to fully automate the whole process. Hence, a large ontology matching task is to divide into a set of local matching tasks. Each local matching task is considered as a matching context that should be aligned independently from the other local matching tasks. The local matching learning automatically selects the adequate matchers for each local matching task. Usually ontology matching systems combine different types of matchers to align pairwise ontologies [94, 14, 13]. Ideally, matchers should complement each other in order to reduce the generated matching quality [71]. An element level matcher can be based on a similarity measure, which derives a similarity score between the entities of the pairwise ontologies. Structural matchers are based on structural features extracted from the relation between entities of a single ontology. These matchers should be automatically defined for each matching context. We automatically combine element-level and structural-level matcher using machine learning techniques.

In this chapter, we propose a local matching learning approach to fully automate the matching tuning for each local matching task. This automation should be automatically defined for each matching context in order to result in a context-independent local matching learning system. This local matching learning system should align each ontology matching task based on the specificities of the local matching tasks. Since we are dealing with a large ontology matching task, we apply the proposed matching learning approach locally and not globally. Therefore, we define adequate matching tuning for each local matching task. State-of-the-art approaches define a single matching tuning for all the matching contexts, otherwise, they define a set of matching tuning for a set of preselected context. However, the benefit of the local matching learning approach is the use of machine learning methods, which can be flexible and self-configuring during the training process. Therefore, we result in a better matching quality independently of the matching context. After identifying the set of local matching tasks $\mathcal{LM}_{ij}$ as introduced in the last chapter, we perform the local matching learning of each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. The local matching learning automatically discovers the alignments between the entities of each local matching task. In section 4.2, we present the formal notation employed throughout this chapter. The local matching approach follows four stages: (i) Local training sets generation (Section 4.3) (ii) Resampling the generated local training sets (Section 4.4) (iii) Wrapper-based local feature selection of the local training sets (Section 4.5) and (iv) Local-based classification (Section 4.5).

4.2 Formal foundations

4.2.1 Global Matching Versus Local Matching

Global matching perform the matching learning between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ using a single classifier. This classifier defines a single matching tuning for a large ontology matching task. However, the local matching learning aligns two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ based on a set of classifiers. Each classifier focuses on a local matching task by defining its matching tuning.

In the following we provide the definition of global matching. Local matching is defined in definition 3.

Definition 8 (Global matching) *The global matching $\mathcal{GM}_{ij}$ between two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ consists in learning correspondences between all the entities of $\mathcal{V}_i$ and $\mathcal{V}_j$. The global matching $\mathcal{GM}_{ij}$ requires a single machine learning classifier $\mathcal{G}ml_{ij}$ generated from one global training set $\mathcal{G}ts_{ij}$.*

4.2.2 Local-based Training Data

For each local matching $lm_{ij,q}$ of $\mathcal{LM}_{ij}$, we automatically generate a local training set denoted by $ts_{ij,q}$. A local training set $ts_{ij,q}$ is built automatically to generate its specific classifier. A classifier defines the matching tuning of a matching task. Existing works lack the automation of the generation of the training data. State-of-the-art methods generate a single training data for all the possible matching contexts. However, each matching context has its unique characteristics and should be aligned based on its adequate classifier. The automatic generation of the training data results in the appropriate classifier for each local matching task. Therefore, the alignment of local matching tasks based can result in better quality alignments. In the following, we define the local-based training data.

Definition 9 (Local-based training Set) *A set of local machine learning classifiers is generated from a set of local training sets $\mathcal{T}s_{ij}$, denoted by:*

$\mathcal{T}s_{ij} = \{ts_{ij,1},\dots,ts_{ij,q},\dots\}$.

Each local training set $ts_{ij,q}$ contains a set of features $\{f_{ijq,1},\dots, f_{ijq,r}\}$ associated with a prediction class attribute c_{ijq} (match/not match). Each feature corresponds to a matcher. We denote a local training set by:

$ts_{ij,q} = \{f_{ijq,1},\dots, f_{ijq,r}, c_{ijq}\}$, *such that $c_{ijq} \in \{0,1\}$, such that c_{ijq} is an integer representing the class attribute of the binary classification task.*

Table 4.1 represents an example of a local training set.

Table 4.1: An example of a local training set.

Local matching entity pairs	Feature 1	...	Feature r	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
...	...	...	...	...
($e_{i,q,x}$,$e_{j,q,y}$)	...	...	...	...

4.2.3 Local-based Classifier

Existing works generate a single classifier for all the ontology matching tasks. A classifier defines the matching settings of an ontology matching task. However, each matching task has its specificity that should be taken into consideration by the classifier during the alignment process. We build a local-based classifier in order to align the entities of a local matching task. This classifier is automatically generated based on its local-based training set.

Definition 10 (Local-based classifier) *A local matching $\mathcal{LM}_{ij}$ is performed using a set of local machine learning classifiers $\mathcal{L}cl_{ij}$ denoted by:*

$\mathcal{L}cl_{ij} = \{lcl_{ij,1},..., lcl_{ij,q}\}$.

Every local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$ uses a single local training set $ts_{ij,q}$ to generate its local classifier $lcl_{ij,q}$. An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category. In our case, a classifier classifies a candidate correspondence to be aligned or not.

4.3 Local Matching Learning Architecture Overview

Each matching context should be automatically aligned based on its characteristics. Capturing all the characteristics of a matching task by a single matcher is really difficult. Usually, each matcher focuses on a single type of heterogeneity. As a result, the automatic combination of the available matchers is required. The choice of matchers should be automated especially for the alignment of large ontologies. As a result, matching settings should be set automatically. Thus, the matching process should be self-tuned in order to derive the appropriate matching tuning for each matching context. Usually, state-of-the-art approaches generate a single classifier for each ontology matching task. However, each ontology matching task has its specific context different from the other matching context. Moreover, existing approaches construct a single classifier based on training data similar to the matching context that should be aligned. Therefore, this single classifier is less appropriate for all matching context. In our case, we build a classifier for each local matching task taking into consideration its context.

In this section, we discuss the main features of the proposed local matching learning approach. In Figure 4.1, we depict the main steps of the local matching learning approach. This approach automatically aligns the set of local matching tasks based on machine learning techniques. Local matching tasks are generated by the partitioning approach, which is presented in Chapter 3. The local matching learning approach has as an input the set of local matching tasks and results in a single output alignment file for the ontology matching task. This output alignment file contains the set of generated mapping for each local matching task. The local matching process is composed of five steps. This approach aligns each local matching task based on its unique classifier. Classifiers are automatically generated from an external knowledge base.

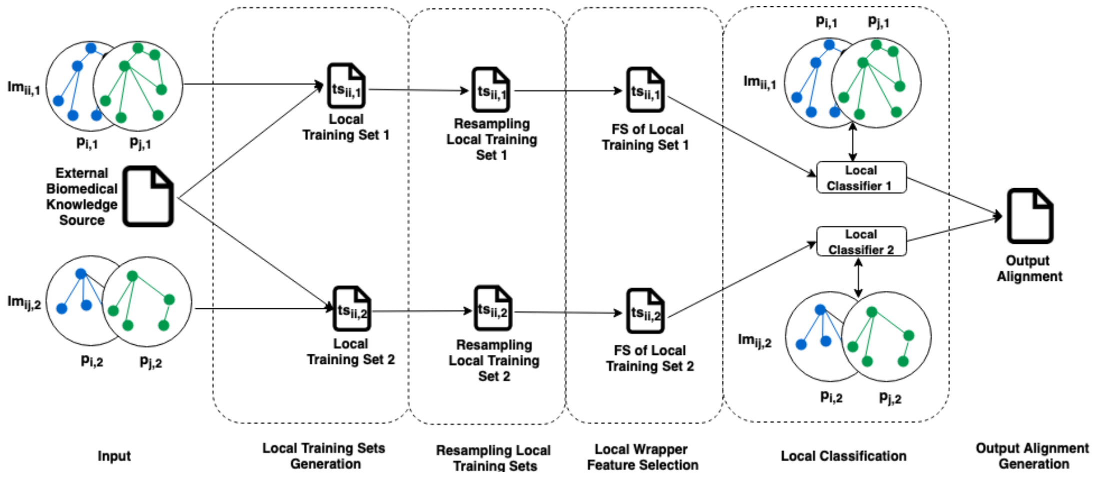


Figure 4.1: Local Matching Learning Architecture Overview

In the following, we give an overview of the different steps of the local matching learning approach.

- **Local training sets generation** The input of this step is a set of local matching tasks generated by the ontologies partitioning approach. The output is a training set for each local matching task. We generate training sets for each local matching task based on an external biomedical knowledge base. As a result, training sets of different local matching tasks are not similar in terms of their positive samples and negative samples. Therefore, the generated classifiers based on the later training sets are flexible and self-configuring in terms of the matching tuning of local matching task. State-of-the-art approaches define a single static training set for all the ontology matching task, which results in a single matching setting for different matching context. As a result, matching accuracy can be decreased [82].

- **Re-sampling local training sets** The generated local training sets are not balanced. The number of negative samples is higher than the number of positive samples. Therefore, we apply resampling techniques in order to generate a better classifier for each local matching task.

- **Local wrapper feature selection** During this step we perform the feature selection for the resampled local training sets. This process aims to include only adequate features for the local classifiers. Therefore, each local matching task is aligned based on its adequate features. Each feature corresponds to a matcher. Therefore, only the best features for each local matching context are selected. Each local matching task can be aligned based on its unique set of features. Consequently, the accuracy of the generated alignments can be improved.

- **Local classification** This step has as an input the local training sets for the local matching tasks. The output is a set of alignments for each local matching task. During this step, we build a classifier for each local matching task. Each classifier aligns its corresponding local matching task.

- **Output alignment generation** During this step, the generated alignments for each local matching task are unified in order to result in a single alignment file. This alignment file contains the correspondences between the two input ontologies.

In the next sections, we explain in depth the workflow of each step of the local matching learning approach.

4.4 Local Training Sets Generation

Each local matching task has its own specific context. Therefore, a local matching task should be aligned based on its adequate matching settings, such as the weight of each matcher and its threshold. Related work generates a single static training set in order to build a classifier for large ontology matching tasks. However, each ontology matching learning task has its own characteristics and should be aligned accordingly. To cope with this issue, a local based classifier should be built for each local matching task. Therefore, we automatically construct a local training set $ts_{ij,q}$ for each local matching task $lm_{ij,q}$. A local training set serves as the input for its local classifier. Differentiated from the existing work, we automatically generate the local training sets without any manual user involvement or any reference alignments.

To summarize, for a given local matching $lm_{ij,q}$, we generate:

- Positive samples $PS_{ij,q} = \{(e_{i,q,x}, e_{j,q,y})\}$, such as the total number of these positive samples is $\mathcal{N} = |PS_{ij,q}|$.

- Negative samples $NS_{ij,q} = \{(\mathcal{V}_{i,q} \times \mathcal{V}_{j,q}) \setminus PS_{ij,q}\}$. Negative samples correspond to all the entity-pairs of a local matching task minus the entity-pairs of the already generated positive samples. Therefore, the total number of negative samples is $\mathcal{M}$ $= \mathcal{N}(\mathcal{N}\text{-}1)$.

4.4.1 Generating Positive Samples

Labeled data for the class attribute c_{ijq} are usually hard to acquire. Existing work construct the labeled data either from the reference alignments or by creating it manually [82]. However, the reference alignments commonly do not exist. We derive the positive mappings samples (the minority class) of the class attribute c_{ijq} by combining the results of two methods: cross-searching and cross-referencing of external knowledge bases for the same domain of application, in particular, the biomedical domain for this Ph.D. thesis. This combination allows the labeling of the local training sets to cover a wide range of biomedical ontologies. In the following, we detail the cross-searching, the cross-referencing and the exact matching methods. All the positive samples generated by the three later methods are unified into a single positive sample set $PS_{ij,q}$. We remove redundant instances from the positive samples set $PS_{ij,q}$. Therefore, the set of positive samples may contain samples from the three approached methods. This process enriches the training data in order to build a better classifier for a better alignment. The lack of positive samples may result in wrong assumptions generated by the classifiers.

4.4.1.1 Generating Positive Samples via Cross-searching

Cross-searching employs external biomedical knowledge sources as a mediator between local matching tasks in order to extract **bridge alignments**. A bridge alignment is extracted if a similar annotation is detected between an entity of the ontology and two entities of a local matching task. We consider bridge alignments as the set of positive samples.

In Algorithm 3, we depict the different steps of the cross-searching method for generating positive samples from external biomedical knowledge bases.

Algorithm 3 Cross-searching Algorithm

```
1: Input $p_{i,l}$, $p_{j,k}$                                      ▷ (1) Input
2: $PS_{ij,q} \leftarrow \emptyset$                                ▷ (2) Initialization
3: sourcePartition = $p_{i,l}$.getNames()
4: targetPartition = $p_{j,k}$.getNames()
5: externalBKS = Uberon.getNames()
6: for each externalName of externalBKS do                ▷ (3) Positive samples generation
7:     for each sourcePName of sourcePartition do
8:         if sourcePartition.contains(externalName) then
9:             for each targetPName of targetPartition do
10:                if targetPartition.contains(externalName) then
11:                $PS_{ij,q}$.add(sourcePName.getClass, targetPName.getClass)
12:                end if
13:            end for
14:        end if
15:    end for
16: end for
17: Return $PS_{ij,q}$                                              ▷ Output
```

The Cross-searching Algorithm has as input the two partition $p_{i,l}$ and $p_{j,k}$ corresponding to their local matching task $lm_{ij,q}$. The output of Algorithm 3 is the generated set of positive samples $PS_{ij,q}$ for their local training set. As mentioned in Algorithm 3, the cross-searching algorithm follows the next three steps:

Cross-searching Algorithm Initialization

We initialize the set of positive samples $PS_{ij,q}$ as an empty set. We also extract the annotations of the source $(p_{i,l})$ and target $(p_{j,k})$ partitions. These annotations are respectively assigned to the variables *sourcePartition* and *targetPartition.* Moreover, we extract the annotations of the employed external biomedical knowledge base. The extracted annotations are assigned to the variable *externalBKS.* In the next step, we employ all the extracted annotations in order to generate bridge alignments between $p_{i,l}$ and $p_{j,k}$ for their local matching task $lm_{ij,q}$.

Positive Samples Generation

During this step, we perform a nested for loop in order to extract bridge alignments between the local matching task $lm_{ij,q}$ and the external biomedical knowledge base. In our case, we employ UBERON as an external knowledge base for the biomedical domain. Bridge alignments are considered as the set of positive samples $PS_{ij,q}$. Therefore, the algorithm performs a nested loop of names of the external knowledge base and the partitions $p_{i,l}$ from $\mathcal{O}_i$ and $p_{j,k}$ from $\mathcal{O}_j$. If the annotations (names) of the external knowledge base contain the same annotation (*externalName*) in both partitions $p_{i,l}$ and $p_{j,k}$, then, we align via a bridge alignment the two classes of the later partitions. This bridge alignment is added to the set of positive samples $PS_{ij,q}$ (line 11).

4.4.1.2 Generating Positive Samples via Cross-referencing

We can also perform the labeling of the raining data via cross-referencing. This method enriches the already discovered positive samples by the cross-searching by new labels. Extracted positive samples by the cross-searching method can maybe not have a sufficient number to build a training set. Therefore, we extract more positive samples using the cross-referencing method, we are based on the vocabulary of the Open Biomedical Ontologies (OBO) [98]. OBO is an effort to create controlled vocabularies for shared use across different biological and medical domains [98]. The OBO ontology library forms the basis of the OBO Foundry, a collaborative experiment involving a group of ontology developers who have agreed in advance to the adoption of a growing set of principles specifying best practices in ontology development. These principles are designed to faster interoperability of ontologies within the broader OBO framework and also to ensure a gradual improvement of quality and formal rigor in ontologies. The OBO library operates to design ways to meet the increasing needs of data and information integration in the biomedical domain [98].

The classes of different biomedical ontologies are interconnected via the annotation property ”oboInOwl:hasDbXref” based on the OBO vocabulary [98]. Consequently, we are based on one of the OBO ontologies, which is UBERON. We employ Uberon as an external biomedical knowledge source in order to derive positive samples for each local training set. Uberon is an integrated cross-species ontology covering anatomical structures and includes relationships to taxon-specific anatomical ontologies. Indeed, we explore the OBO annotation property "oboInOwl:hasDbXref", which is mentioned in almost every class of Uberon. This property references the classes URI of external biomedical ontologies. We align every two classes of a given local matching task in case if one of their classes are both referenced in a single Uberon class.

The UBERON ontology includes references to different biomedical ontologies (via annotation property "oboInOwl:hasDbXref"). For instance, the class UBERON_0001275 ("pubis") of Uberon references the FMA class 16595 ("pubis") and NCI class C33423 ("pubic bone"). Therefore, the later entities construct a positive sample of its local training set.

In Algorithm 4, we depict the different steps of the cross-referencing method for generating positive samples from external biomedical knowledge bases. The difference between the cross-referencing method and the cross-searching method is that the earlier method does not explore the annotation property "oboInOwl:hasDbXref", which reference other biomedical ontologies. The cross-searching method explores only the value of the names of the external knowledge base annotations. Therefore, the returned positive samples of the two methods are different. This difference compliments the set of positive samples by a set of alignments not discovered by cross-reference method.

Algorithm 4 Cross-referencing Algorithm

```
1:  Input $p_{i,l}$, $p_{j,k}$                                        ▷ (1) Input
2:  $PS_{ij,q} \leftarrow \emptyset$                                   ▷ (2) Initialization
3:  set externalBKS = ontology.getClasses(Uberon)
4:  set sourcePartitionURIs = partition.getURIs($p_{i,l}$)
5:  set targetPartitionURIs = partition.getURIs($p_{j,k}$)
6:  for each externalClass of externalBKS do              ▷ (3) Positive samples generation
7:      sourceOBO = ontology.getOBODBxref(externalClass, sourcePartitionURIs)
8:      targetOBO = ontology.getOBODBxref(externalClass, targetPartitionURIs)
9:      for each sourceURI of sourcePartitionURIs do
10:         if sourceURI.equals(sourceOBO) then
11:             for each targetURI of targetPartitionURI do
12:                 if sourceURI.equals(targetOBO) then
13:                 $PS_{ij,q}$.add(sourceURI.getClass, targetURI.getClass)
14:                 end if
15:             end for
16:         end if
17:     end for
18: end for
19: Return $PS_{ij,q}$                                                 ▷ Output
```

As mentioned in Algorithm 4, the cross-referencing algorithm follows the following steps:

Cross-referencing Algorithm Input and Output Similar to the cross-searching Algorithm, the Cross-referencing Algorithm has as input the two partition $p_{i,l}$ and $p_{j,k}$ corresponding to their local matching task $lm_{ij,q}$. The output of Algorithm 4 is the generated set of positive samples $PS_{ij,q}$ for their local training set.

Cross-referencing Algorithm Initialization

We initialize the set of positive samples $PS_{ij,q}$ as an empty set. We extract the classes of the external biomedical ontology (line 3), these classes contain the OBO vocabulary. The OBO annotation property "oboInOwl:hasDbXref" of the external biomedical ontology references the URIs of other biomedical ontologies. Therefore, we extract the URIs of the source partition and the target partition (line 4 and line 5). In the next step, we employ all the extracted data in order to generate bridge alignments between $p_{i,l}$, $p_{j,k}$ and the external biomedical knowledge base.

Positive Samples Generation During this step, we perform a nested loop in order to extract bridge alignments between the local matching task $lm_{ij,q}$ and the external biomedical knowledge base. Therefore, the algorithm respectively perform a nested loop of the external knowledge base, the first partition $p_{i,l}$ and the second partition $p_{j,k}$ of their local matching task $lm_{ij,q}$. A class of the external knowledge base may contain different OBO annotation properties "oboInOwl:hasDbXref", referencing different classes of other biomedical ontologies. If one class of the *externalBKS* reference contain two annotation properties respectively referencing a class from the first partition $p_{i,l}$ and another class from the second partition $p_{j,k}$, then we consider the two referenced classes as a bridge alignment discovered by the cross-referencing method. The discovered bridge alignment is added to the set of positive samples (line 13).

4.4.1.3 Generating Positive Samples via Exact Matching

We can also generate positive samples between two partitions $p_{i,l}$ and $p_{j,k}$ of a local matching task $lm_{ij,q}$ based on the exact matching method. Exact matching can add positive samples, which are not discovered by the cross-searching and the cross-referencing. In OAEI, StringEquiv is a matcher employed as a baseline in order to compare the accuracy of the ontology matching systems. In Algorithm 5, we depict the different steps of the exact matching method. Similar to cross-searching and cross-referencing, the exact matching algorithm has as an input the two partition $p_{i,l}$ and $p_{j,k}$. The output of the exact matching algorithm is the set of positive samples $PS_{ij,q}$ for its local training set.

As mentioned in Algorithm 5, the exact matching algorithm follows the following steps:

Exact Matching Algorithm Initialization We extract the annotations of the source ($p_{i,l}$) and target ($p_{j,k}$) partitions. These annotations are respectively assigned to the variables of source ($p_{i,l}$) and target ($p_{j,k}$) partitions. Moreover, we initialize the positive samples $PS_{ij,q}$ as an empty set.

Positive Samples Generation We perform a nested loop between the annotations of the source and target partitions. If we found that the content of a single annotation is exactly the same between the classes of the later partitions, then we add it to the list of positive samples $PS_{ij,q}$.

Algorithm 5 Exact-matching Algorithm

```
1: **Input** $p_{i,l}$, $p_{j,k}$                                        ▷ (1) Input
2: $PS_{ij,q} \leftarrow \emptyset$                                      ▷ (2) Initialization
3: sourcePartition = $p_{i,l}$.getNames()
4: targetPartition = $p_{j,k}$n.getNames()
5: **for each** sourceName of sourcePartition **do**             ▷ (3) Positive samples generation
6:     **for each** targetName of targetPartition **do**
7:         **if** sourceName.equals(targetName) **then**
8:         $PS_{ij,q}$.add(sourceName.getClass, targetName.getClass)
9:         **end if**
10:    **end for**
11: **end for**
12: **Return** $PS_{ij,q}$                                                ▷ Output
```

4.4.2 The Algorithm for the Generation of Negative Samples

We extract the set of Negative samples $\mathcal{NS}_{ij,q}$ by computing the difference between the set of extracted Positive samples $\mathcal{PS}_{ijq}$ and the set representing the cartesian product of its entities. After extracting the entities for positive samples $PS_{ij,q}$ and negative samples $NS_{ij,q}$, for each local training set, we compute the features for each entity pair of the positive samples $PS_{ij,q}$ as well as the negative samples $NS_{ij,q}$. In the following Algorithm 6, we depict the different steps for the generation of negative samples.

Algorithm 6 Negative Samples Generation Algorithm

```
1: **Input** $p_{i,l}$, $p_{j,k}$, $PS_{ij,q}$                            ▷ (1) Input
2: $i \leftarrow 0$
3: $NS_{ij,q} \leftarrow \emptyset$                                      ▷ (2) Initialization
4: **while** ( i <= $PS_{ij,q}$.geSize) **do**                    ▷ (3) Negative samples generation
5: sourcePS1 ← PS.getSourceClass(i)
6: $j \leftarrow 0$
7:     **while** ( j <= $PS_{ij,q}$.geSize) **do**
8:         sourcePS2 ← PS.getSourceClass(j)
9:             **if** (sourcePS1.notEqual(sourcePS2)) **then**
10:            targetPS ← PS.getTargetClass(j)
11:            $NS_{ij,q}$.add(sourcePS1, targetPS)
12:            **end if**
13:    **end while**
14: **end while**
15: **Return** $NS_{ij,q}$                                                ▷ Output
```

4.4.3 Local Training Set Generation Example

In order to demonstrate an example of a generation of training data, in Figure 4.2, we extracted the following positive samples $PS_{ij,q}$ based on the cross-searching method:

$PS_{ij,q} = \{(e_{i,3},e_{j,8}), (e_{i,2},e_{j,2}), (e_{i,1},e_{j,1})\}$.

The number of extracted positive samples is 3. Therefore, $\mathcal{N} = 3$. These labeled entities are generated by cross-searching the two partitions and an external biomedical knowledge base. This external knowledge base should contain the OBO vocabulary, more specifically the annotation property "oboInOwl:hasDbXref". In our case, we are based on UBERON as a mediating ontology. We deduce that the number $\mathcal{M}$ of negative samples $\mathcal{NS}_{ij,q}$ is equal to the following:

$\mathcal{M} = \mathcal{N}\ (\mathcal{N}\text{-}1) = 6$

Therefore, the generated local training set for this example has 9 instances, corresponding to the sum of 3 positive samples and 6 negative samples. We can deduce that the number of negative samples is higher than the number of positive samples ($\mathcal{M}>\mathcal{N}$). Consequently, we apply resampling techniques over the local training set in order to balance the number of samples.

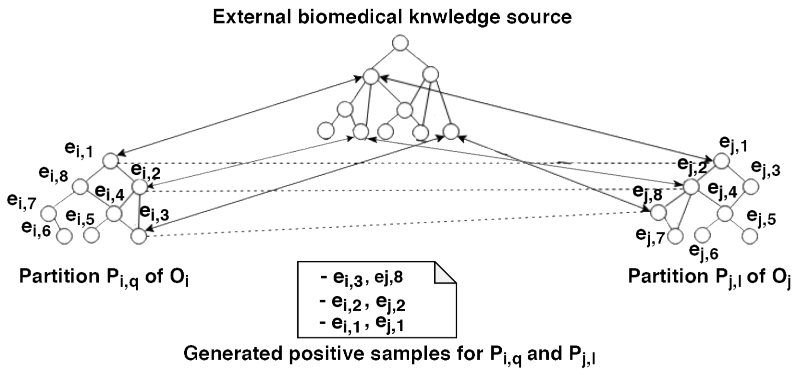


Figure 4.2: Positive samples extraction for a local training set extraction

4.5 Balancing Generated Local Training Sets

Imbalanced training sets are one of the main issues occurring while dealing with ontology matching learning. Imbalanced data typically refers to a classification problem where the number of observations per class is not equally distributed [76]. Current matching learning work does not consider the resampling process to resolve the problem of imbalanced matching learning training data. Nonetheless, resampling is essential to deliver better matching learning accuracy.

Classification problems often suffer from data imbalance across classes. This is the case when the size of instances from one class is significantly higher or lower relative to the other classes. A small difference often does not matter [76]. However, if there is a modest class imbalance in training data like 4:1, it can cause misleading classification accuracy. Imbalanced data refers to classification problems where we have unequal samples for different classes. For instance,

Most of machine learning classification algorithms are sensitive to imbalanced training data. An imbalanced training data will bias the prediction classifier towards the more common class. This happens because machine learning algorithms are usually designed to improve accuracy by reducing the error. Thus, they do not take into account the class distribution of classes. Different methods have been proposed in the state-of-the-art for handling the data imbalance [29]. The common approaches [19] to generate a balanced dataset from an imbalanced one are undersampling, oversampling, and their combination:

- Undersampling approach balances the dataset by reducing the size of the abundant class by keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class;
- Oversampling approach is used when the quantity of data is insufficient. It tries to balance dataset by increasing the size of rare samples. Rather than removing abundant samples, new rare samples are generated;
- Performing a combination of oversampling and undersampling can yield better results than either in isolation.

Most of the state-of-the-art matching learning approaches neglect the problem of the imbalanced dataset. Existing work does not give any importance to resampling. However, the resampling method can strongly affect the obtained accuracy by the matching learning strategy.

Local training sets are not balanced since the number of the negative samples $\mathcal{M}$ is higher than the number of positive samples $\mathcal{N}$. Therefore, we initially undersample each local training set $ts_{ij,q}$ by a heuristic method which consists of removing all the negative samples (majority class) having at least one element level feature equal to zero. The result of this initial treatment is not sufficient to balance the local training data, since the number of negative samples $NS_{ij,q}$ (majority class) still higher than the number of positive samples $PS_{ij,q}$ (minority class). Hence, an additional sampling method is required to result in a balanced local training sets, we employ the state-of-the-art resampling methods to undersample of the majority class $PS_{ij,q}$, oversample the minority class $NS_{ij,q}$ or combining both of the later techniques. In chapter 5, we conduct a comparative study of applying resampling methods over imbalanced training data. We express r as the ratio of the size of the minority class to the majority class. This ratio should be equal to 1 in order to have a similarity between the number of positive samples and the number of negative samples. The ratio r is denoted by:

$r = \|\mathcal{N}\| / \|\mathcal{M}\|$.

The output of this step is a balanced local training set $ts_{ij,q}$ for each local matching task $lm_{ij,q}$. For instance, if a local matching process $\mathcal{LM}_{ij}$ is composed of three local matching tasks: $lm_{ij,1}$ $lm_{ij,2}$ and $lm_{ij,3}$, we respectively result in three local training sets $ts_{ij,1}$ $ts_{ij,2}$ $ts_{ij,3}$.

4.6 Wrapper-based Local Feature Selection

The local matching $\mathcal{LM}_{ij}$ approach splits a large ontology matching problem into a set of smaller local matching tasks $lm_{ij,q}$. Each local matching task focuses on a specific subtopic of interest. Therefore, it should be aligned based on its set of adequate matchers. We classify matchers into structural level matchers and element level matchers. Each matcher corresponds to a feature in a local training set $ts_{ij,q}$. Element level matchers measure correspondences by analysing entities in isolation without taking into consideration the existing relationships between these entities. Whereas, structure level matchers considers the ontological neighborhood of entities in order to determine their similarity. We employ wrapper feature selection in order to determine the suitable element level and structure level features for each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. Consequently, each local matching task is aligned based on its specific set of features. Each feature represents a single matcher. In the following, we present the set of element-level features and structural-level features. These set of features are employed as input for each local matching features selection process.

4.6.1 Element-level Features

We perform features selection over each generated local training set $ts_{ij,q}$ in order to build its local classifier. The latter identifies the local alignments of its local matching task $lm_{ij,q}$. For example, for a given local training sets: $ts_{ij,1}$, $ts_{ij,2}$ and $ts_{ij,3}$, we separately perform the features selection over these three local training sets. As depicted in Table 4.2, we integrate 14 of the state-of-the-art similarity measures as the element level matchers. These similarity measures are widely employed by state-of-the-art matching systems. We classify the 14 similarity measures into four groups: edit-distance, character-based, term-based and subsequence-based. Each group employs a specific technique in order to measure the similarity between two entities. We are based on these four groups in order to cover all types of syntactic heterogeneity between the annotations of the two input ontologies.

Table 4.2: Element level features

Chracter based	Block Distance [15]	A similarity measure based on n-dimensional vector space defined through the characters of input strings.
	Cosine Similarity [18]	Instead of summing the edge distances, this variant calculates the cosine value of the angle between input strings
Subsequence based	QGrams [70]	Computes the similarity by splitting the strings into tokens, comparing the tokens by the help of an internal measure.

Group	Feature	Description
Term based	Dice similarity [99]	It is defined as twice the number of common terms divided by the total number of terms in both input strings
	Jaccard [48]	It is defined as the size of the intersection divided by the size of the union of terms within the input strings
	Euclidian Distance [9]	The similarity is measured through the length of the line segment between two vectors composed of input strings' terms
	Overlap Coefficient [106]	It is defined as the size of common terms divided by the size of the shortest input string
Edit distance	Levenshtein [68]	One of the most widely used string similarity measures based on edit distance, i.e. copy, substitute, insert and delete a character from one string to another
	Needleman Wunch [79]	A weighted variant of Levenshtein by adding a variable cost to the gap.
	ISUB [100]	Considers not only the similar but also the different parts of two strings.
	Smith Waterman [110]	A variant of Needleman Wunch which is originally developed to identify optimal matching between related DNA and protein sequences.
	Monge Elkan [75]	An extension of Smith Waterman by allowing a specific gap penalty function between sequences
	Jaro Winkler [112]	A variant of Jaro similarity measure which is better suited for short strings by the help of an internal measure.
Subsequence based	QGrams [70]	Computes the similarity by splitting the strings into tokens, comparing the tokens by the help of an internal measure.

4.6.2 Structural-level Features

We draw in Table 4.3 the set of structural level features as well as their description. We propose 7 structural level features. These features are based on the following intuition: the more similar the annotations of two entities are the more likely they correspond to each other [35]. A set of structural level features (NbSib, NbSup, NbSub, and Depth) computes the similarity between entities of a local matching $lm_{ij,q}$ task by analyzing their position in the ontology. Moreover, another set of structural level features (SibSim, SupSim, and Subsim) are based on the syntactic similarity of the neighborhood of entities in order to compute their similarity. The intuition behind this is that, if two entities from two local matching tasks are similar, their neighbors must also be somehow similar [35]. In the proposed 7 structural level features, we compute the similarity between two entities based on their siblings, superclasses, subclasses, and depth in each ontology. The generated numeric values of structural level features are normalized between 0 and 1. Some ontologies do not contain a root entity. Therefore, we search for high-level entities of an ontology.

Table 4.3: Structural level features

Feature	Description
NbSib	The modulus between the total number of siblings of two entities
NBSup	The modulus between the total number of superclasses of two entities
NBSub	The modulus between the total number of subclasses of two entities
Depth	The modulus of the distance to the root entity between two entities in a local training set.
SibSim	The modulus of the average similarity score between the siblings of two entities
SupSim	The modulus of the average similarity score between the super-classes of two entities
SibSub	The modulus of the average similarity score between the sub-classes of two entities

4.7 Local Classification

The final step of the local matching approach corresponds to the generation of a classifier for each local training set. Each classifier aligns the entities of its local matching task. Classifiers are based on machine learning algorithms. In chapter 5, we have compared the performance of the state-of-the-art machine learning algorithms for performing the local classification. All the generated candidate correspondences for a local matching task are processed. This process stands for a cleaning process in order to result in a set of correspondences of one-to-one (1:1) multiplicity. This process removes all the one-to-many (1:m) correspondences by only selecting the correspondence with the highest similarity score. The generated correspondences by each classifier are unified into a single alignment file. This alignment file represents the resulted correspondences for the whole ontology matching task. Correspondences can be compared to the reference alignment file in order to compute the accuracy of the local matching learning approach.

4.8 Conclusion

Ontology matching systems combine different matchers in order to align the input ontologies. Structural and terminological matchers result in different sets of alignments that may have in common a certain number of correspondences. Therefore, different matchers complement each other in order to improve the matching accuracy result. Terminological matchers are usually based on a similarity measure associated with its specific threshold. If the similarity score of two entities is above a certain threshold, the entities can be aligned. Therefore, the set of employed matchers should be well defined for each matching context. A good combination of the input matchers is essential to result in better matching efficiency. For each matching task, matchers combination should be automated and self tuned in order to derive the adequate matching setting, e.g. similarity measure, threshold, and matcher weight. State-of-the-art matching learning approaches do not perform full automation of matching process. Usually, these approaches do not take into consideration the matching context before generating the set of final alignments. Hence, matching learning classifier is usually generated from a single training set. This training set does not take into consideration each matching context. Therefore, the matching accuracy depends on the quality of the generated classifier by the training data, which is usually manually created or generated based on references alignments, which usually do not exist.

We have proposed a novel matching learning approach in order to automate matching process for each local matching task. We automatically generate a classifier for each local matching task. Classifiers are built based on their training sets, which are automatically extracted from external knowledge sources. These training sets are initially imbalanced due to the high number of negative samples compared to the number of positive samples. Imbalanced training data negatively affect the accuracy of the generated alignment by the local matching learning approach. Consequently, we applied resampling techniques over each training set of the sub-matching tasks in order to improve the alignment quality. Furthermore, we have applied features selection techniques over each training set of a sub-matching task. Each feature corresponds to a matcher. Therefore, the alignment process is fully automated for each sub-matching task in terms of the generation of training-set, the choice of its specific matchers and the construction of its adequate classifier. Every sub-matching task is aligned based on its adequate classifier independently from the other sub-matching tasks. Therefore, alignment quality is improved for the overall matching task.

Chapter 5

Evaluation

Contents

5.1 Introduction

In this chapter, we evaluate our approach for ontology partitioning and local matching learning. The evaluation asses the performance of our approach, which addresses two major issues: (i) integrating the large ontologies with a good matching accuracy, (ii) automating the ontology matching process. We address these two issues by proposing an ontology matching system called POMap++. POMap++ cope both with the large size of ontologies and the automation issues. POMap++ implements the local matching learning approach that combines ontology partitioning and matching learning. Therefore, we divide a large ontology matching task into a set of local matching tasks. This division is ensured based on our partitioning approach to cope with the following issues: the conceptual heterogeneity, the high search space and the quality of the resulted alignments. Hence, we reduce the huge search space into a set of smaller local matching tasks. The local matching approach automatically independently align each local matching task using machine learning techniques. The local matching process is self-tuned in order to derive the appropriate matching tuning for each local matching task matching. As a result, we can align effectively each local matching task to result in better matching accuracy. The evaluation of POMap++ aims at comparing our partitioning approach to the-state-of-the-art approaches. Moreover, based on POMap++, which implements the local matching learning approach, we asses the efficiency of the local matching approach to the existing ontology matching systems. Moreover, we compare our local matching learning approach to the global matching learning approach. The main purpose of this comparision is to know if there is a matching accuracy gain resulted by automatically and independently aligning the set local matching task compared to aligning large ontologies as a single matching task. We also study the performance of the different resampling techniques for balancing the automatically generated local training sets. This study aims to suggest the adequate resampling strategy. Furthermore, we study the impact of the feature selection methods for performing the local matching approach. This study aims to justify the choice of different employed features.

In order to evaluate the effectiveness of the proposed approach, we run experiments based on the benchmark of the OAEI campaign. We compare the results of the proposed matching system POMap++ with the systems of other participants in the campaign. POMap++ is the ontology matching system implementing all the contribution discussed in Chapter 3 and Chapter 4. Moreover, we have participated with PPOMap++ in the benchmark of OAEI 2017 [62] and OAEI 2018 [65].

In Section 5.2, we firstly introduce the experimental environment employed for the evaluation of the proposed contributions. Then, in Section 5.3, we evaluate the ontology partitioning approach. Later, in Section 5.4, we begin to evaluate the local matching learning approach based on different machine learning algorithms. This comparison aims to result in an adequate machine learning algorithm for the local matching learning approach. Once the algorithm is chosen, we study the impact of the imbalanced training sets for performing the local matching learning approach. Training sets are imbalanced since the number of negative samples is higher than the number of positive samples. The imbalanced training set affects the matching accuracy. In Section 5.5, we show the results of the resampling process of the training sets using state-of-the-art techniques. These techniques are the undersampling of the majority class, the oversampling of the minority class and the combination of the later techniques. After the resampling process, in Section 5.6, we study the impact of feature selection methods. Moreover, we compare the effectiveness of element level features to the structural level features. This study aims to find a way for future improvement of the employed features. In Section 5.7, we compare the local matching learning to the global matching learning as well as our results compared to the state-of-the-art matching systems. This comparison aims to study the efficiency of the local matching learning approach.

5.2 The Architecture for Experimental Assessment

In this section, we introduce the experimental environment employed throughout this chapter. We firstly present the architecture for experimental assessment. Then, we introduce the experimental configuration used to conduct our experiments. Later, we present the different datasets employed to asses the efficiency of the proposed contributions.

5.2.1 Experimental Archiecture

in Figure 5.1, we present the main processes of the proposed architecture for onology partitioning and local matching learning approach. The partitioning and local matching learning architecture has as input two ontologies and outputs the generated alignments. The ontology matching system POMap++, which participated in OAEI 2018 follows this architecture. The proposed architecture contains mainly four modules: (i) input ontologies indexing, (ii) input ontologies partitioning, (iii) local matching learning and (iv) output alignment generation.

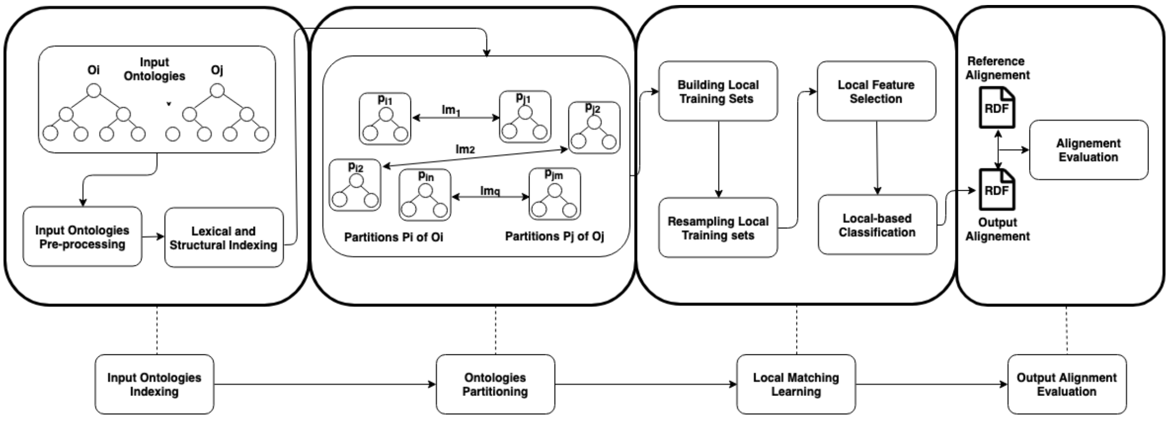


Figure 5.1: Partitioning and Local Matching Learning Architecture

5.2.2 Experimental Configuration

All experiments have been implemented in Java and Python on a MacOs operating system with 2.8 GHz Intel I7-7700HQ (4 cores) and 16 GB of internal memory. This platform is similar in terms of its configuration to the OAEI configuration. Therefore, we employ it in order to asses POMap++. We also conducted experiments on the servers of OSIRIM[1] (Observatoire des Systèmes d'Indexation et de Recherche d'Information Multimédia). This platform is open to researchers and students working in IRIT, but also for external researchers under conditions. OSIRIM is composed of a storage area of 1 Po with a cluster of 928 cores and 28 GPUs. This platform eases the running of multiple experiments at the same time. Therefore, we employed it to simultaneously align large biomedical ontologies matching tasks.

5.2.3 Datasets

Since we are proposing an ontology matching system that aligns large biomedical ontologies datasets, are based on the dataset provided by Ontology Alignment Evaluation Initiative (OAEI). OAEI aims at comparing ontology matching systems on precisely defined test cases. These datasets serve as an input for POMap++, which implements the partitioning and the local matching learning approaches. The output is a set of generated alignments by POMap++. These resulted alignments are compared to the reference alignments provided by OAEI benchmark in order to asses the effeciency of our proposed approach. These test cases can be based on ontologies of different levels of complexity (from simple thesauri to expressive OWL ontologies) [1]. We are based mainly on two matching tracks: Anatomy track and LargeBio track. These two tracks are widely employed by the existing work to asses matching systems aligning biomedical ontologies.

- The Anatomy track contains a single matching task.
- The Largebio track contains three matching tasks: FMA-NCI, FMA-SNOMED, and SNOMED-NCI.

We present in the following the two matching tracks :

- The anatomy real-world case is about matching the Adult Mouse Anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy. The task is placed in a domain where we find large, carefully designed ontologies that are described in technical terms. Besides their large size and conceptualization that is only to a limited degree based on the use of natural language, they also differ from other ontologies with respect to the use of specific annotations and roles, e.g. the extensive use of the ”partOf” relationship. The manual harmonization of the ontologies leads to a situation, where we have a high number of rather trivial mappings that can be found by simple string comparison techniques. At the same

[1]http://osirim.irit.fr/site/

time, we have a good share of non-trivial mappings that can require careful analysis and sometimes also medical background knowledge [1].

- The LargeBio track was elaborated to find alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI). These ontologies have the advantage of being semantically rich and contain tens of thousands of classes. UMLS Metathesaurus has been selected as the basis for the track reference alignments. UMLS is currently the most comprehensive effort for integrating independently-developed medical thesauri and ontologies, including the Foundational Model of Anatomy Ontology (FMA), SNOMED CT, and NCI. The integration of new UMLS sources combines automatic techniques, expert assessment, and auditing protocols. Therefore, we are based on this dataset, which is employed in the biomedical domain and provided by the OAEI benchmark with its reference alignments.

In the following we present each of the employed ontologies by the Anatomy track and the LargeBio Track:

- The Anatomy track:
  - The anatomy real-world case is about matching the Adult Mouse Anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy. The focus of the anatomy track is to confront existing matching technology with real-world ontologies. The ontologies of the anatomy track are the NCI Thesaurus describing the human anatomy, published by the National Cancer Institute (NCI) [38], and the Adult Mouse Anatomical Dictionary, which has been developed as part of the Mouse Gene Expression Database project. Both resources are part of the Open Biomedical Ontologies (OBO).
- The LargeBio track:
  - FMA is an evolving computer-based knowledge source for biomedical informatics [93]. It is concerned with the representation of classes or types and relationships necessary for the symbolic representation of the phenotypic structure of the human body in a form that is understandable to humans and is also navigable, parseable and interpretable by machine-based systems [93]. Specifically, the FMA is a domain ontology that represents a coherent body of explicit declarative knowledge about human anatomy. Its ontological framework can be applied and extended to all other species.
  - SNOMED CT or SNOMED Clinical Terms is a systematically organized computer processable collection of medical terms providing codes, terms, synonyms, and definitions used in clinical documentation and reporting [32]. SNOMED CT is considered to be the most comprehensive, multilingual clinical healthcare terminology in the world [32]. The primary purpose of SNOMED CT

is to encode the meanings that are used in health information and to support the effective clinical recording of data with the aim of improving patient care. SNOMED CT provides the core general terminology for electronic health records [32]. SNOMED CT comprehensive coverage includes clinical findings, symptoms, diagnoses, procedures, body structures, organisms and other etiologies, substances, pharmaceuticals, devices, and specimens.

– NCI provides reference terminology for many NCI and other systems. It covers vocabulary for clinical care, translational and basic research, and public information and administrative activities

5.3 Ontology Partitioning Approach Evaluation

Large ontologies alignment is a complex task due to their high conceptual heterogeneity. Different ontology developers construct the same domain ontology using different conceptual models. A high conceptual heterogeneity affects the efficiency of ontology matching systems. As a result, the discovery of adequate alignments becomes a challenging process. Moreover, the alignment of large ontologies results in a huge search space, where ontology matching systems fetch for the right correspondences. This process is time consuming especially where the matching process should combine different matchers. The main issues of the alignment of large biomedical ontologies are the conceptual heterogeneity, the high search space and the decreased quality of the resulted alignments.

We have proposed a novel ontology partitioning approach to cope with the later issues. In figure 5.2, we depict the employed experimental architecture for the partitioning process, which is proposed in Chapter 3. The partitioning approach has as input two large ontologies and results as an output a set of local matching tasks. This set of local matching task is later aligned using the matching learning process. We evaluate the accuracy of the resulted alignment based on the biomedical anatomy track of OAEI benchmark. Moreover, we evaluate the partitioning approach based on the coverage ratio criteria. The partitioning coverage ratio defines the set of alignments that can be discovered after performing the partitioning process. A low coverage ratio leads to missing a high number of alignment that should be discovered (reference alignments). The partitioning approaches split a large ontology matching task into a set of sub-matching tasks called local matching tasks. As a result, the search space is reduced from the alignment of the hole large matching into the matching of smaller matching tasks. Consequently, the alignment process can reduce the number of false positives to result in better matching accuracy. Our approach overcomes the issue of conceptual heterogeneity by performing a novel multi-cut strategy and a partitions merging method. Consequently, this approach generates do not generate large partitions or isolated ones. The proposed ontology partitioning approach follows mainly three stages: (i) the input ontologies partitioning pre-processing, (ii) the partitioning algorithm, and (iii) the identification of local matching tasks.

We evaluated the proposed partitioning approach according to the current partitioning

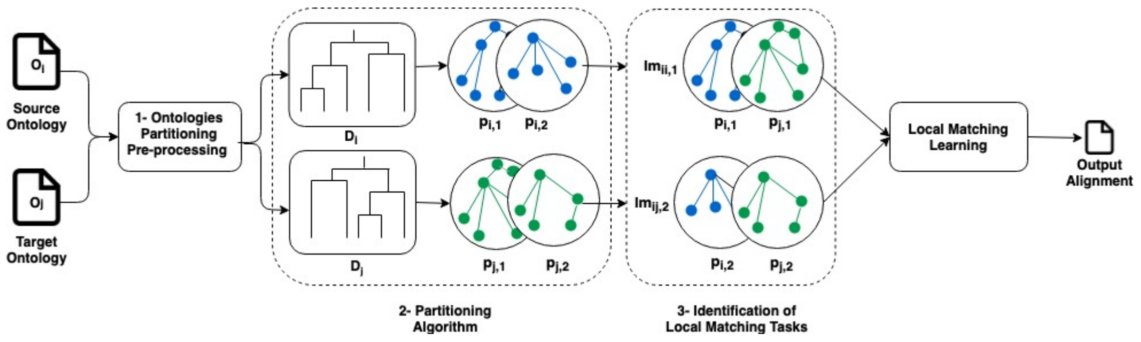

Oi
Source Ontology
Oj
Target Ontology
1- Ontologies Partitioning Pre-processing
Di
Dj
Pi,1
Pi,2
Pj,1
Pj,2
2- Partitioning Algorithm
Imij,1
Imij,2
3- Identification of Local Matching Tasks
Local Matching Learning
Output Alignment

Figure 5.2: Ontologies Partitioning Experimental Architecture

strategies SeeCOnt [4], Falcon [44], [5] and [51]. All these approaches are evaluated using the biomedical anatomy track of OAEI. In Table 5.1, 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. A good coverage ratio guarantee that performing the partitioning process does not result in separating the expected correspondence to be found. Moreover, the recall value is obtained due to the reduction of the search space from the entity-pairs of two ontologies to only the entities of the local matching tasks. As a result, the number of false-positive can be reduced during the matching process. The precision value is lower than Seecont [4], Falcon [44] and [5] due to the additional number of wrong alignments discovered compared to the later approaches. SeeCOnt [4] 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 7 of Chapter 3.

Table 5.1: Anatomy track partitioning results

Approach	F-Measure	Precision	Recall	Partitions	Coverage Ratio	Run time (mn)
POMap++	**0.896**	**0.915**	**0.877**	**57/57**	**98.3**	**8.13**
SeeCOnt [4]	0.863	0.951	0.789	n/d	n/d	n/d
Jimenez-Ruiz et al. [51]	0.850	0.880	0.820	5/10	n/d	42
Falcon [44]	0.730	0.964	0.591	139/119	n/d	10
Algergawy et al. [5]	0.753	0.975	0.613	84/80	n/d	0.98

We can deduce from Table 5.1 that the proposed partitioning approach outperforms the existing approaches in terms of F-measure and recall. We argue these results due to the achieved coverage ratio [64]. This coverage ratio is obtained because of the performed multi-cut strategy and partitions merging strategy of Algorithm 2 of Chapter 2. We achieved a high coverage ratio of 98.3%. Therefore, the partitioning process of two ontologies into a set of local matching tasks results in missing only 1.7% of the expected correspondences to find the reference alignments. As a result, we maximize the probability of adequate alignments discovery, which results in better matching accuracy. The partitions merging strategy ensures that almost the entities that should be aligned are covered. The number of the resulted partitions for the two ontologies of the anatomy track is 57 for both ontologies due to the proposed merging strategy. Therefore, there is no ontology that can have a higher number of partitions (after merging) than the other ontology. This is ensured due to the proposed partitions merging strategy.

5.4 Machine Learning Algorithms Evaluation

Combination of different matchers is essential to align effectively an ontology matching task. An automatic combination is highly required in order to provide the adequate matchers for each ontology matching task. Therefore, the ontology matching system should not align an ontology matching task based on only one matching setting. Each ontology matching task has its specific matching settings that should be automatically defined. In our case, we are dealing with large biomedical ontologies matching. We divide an ontology matching task into a set of local matching tasks. Therefore, we automatically provide the matching tuning for each local matching task. The main issue that we are dealing with is the automatic choice of the matching settings (eg. the weight of each matcher, the threshold) for every local matching task. Therefore, the matching process should be self-tuned for every local matching task in order to result in a good matching accuracy for a large matching task.

The purpose of local matching learning is to divide a large ontology matching task into a set of local matching tasks that are independently and automatically aligned. This process is ensured based on machine learning techniques. The local matching learning process selects adequate matchers for each local matching task. We are based on element level matcher and structural level matchers as a set of initial matchers before the automatic selection process. After identifying the set of local matching tasks $\mathcal{LM}_{ij}$, we perform the local matching learning of each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. The local matching learning automatically discovers the alignments between the entities of each local matching task.

The employed architecture for this experiment is the same as depiected in Figure 4.1. This architecture has as input the set of local matching tasks and results in a set of alignments for the whole ontology matching task.

In Figure 5.2, we compare the local matching accuracy while employing different machine learning algorithms for building the set of local machine learning classifiers. Since we are performing a supervised classification strategy, we draw the results of the top-performing machine learning algorithm for each classification type. We have employed the common supervised classification machine learning algorithms used by the state-of-the-art for matching learning. These algorithms are reported in the following:

- SVM: A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples

- Bayes: Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. The featured image is the equation with P(A—B) is posterior probability, P(B—A) is a likelihood, P(A) is class prior probability, and P(B) is predictor prior probability.

- Decision rules: A decision rule is a simple IF-THEN statement consisting of a condition (also called antecedent) and a prediction. For example: IF it rains today AND if it is April (condition), THEN it will rain tomorrow (prediction). A single decision rule or a combination of several rules can be used to make predictions.

- Decision trees: Decision tree methods construct a model of decisions. That is made based on actual values of attributes in the data. Decisions fork in tree structures until a prediction decision is made for a given record. Decision trees are trained on data for classification and regression problems. Decision trees are often fast and accurate and a big favorite in machine learning.

In Figure 5.3, we report the achieved results by SVM (function), Naive Bayes (bayes), JRip (decision rules) and Random Forest (decision tree). We also depict the resulted accuracy based on the OAEI 2018 biomedical ontologies matching tasks: the Anatomy dataset and the Largebio dataset fragments of FMA-NCI, FMA-SNOMED, and SNOMED-NCI. Random Forest achieved a better accuracy for the local matching learning than the other algorithms. Random forests overcome several problems with decision trees, including:

- Reduction in overfitting: by averaging several trees, there is a significantly lower risk of overfitting.

- Less variance: By using multiple trees, you reduce the chance of stumbling across a classifier that does not perform well because of the relationship between the train and test data.

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.

As depicted in Figure 5.3, the results for the Anatomy track are better than the other matching tasks. We argue this result because the anatomy track contains less conceptual heterogeneity compared to the large track. For instance, the maximum depth of the human anatomy ontology is nine entities. However, the maximum depth of FMA ontology is twenty-four entities. Therefore, the alignment of Largebio tasks is harder than the

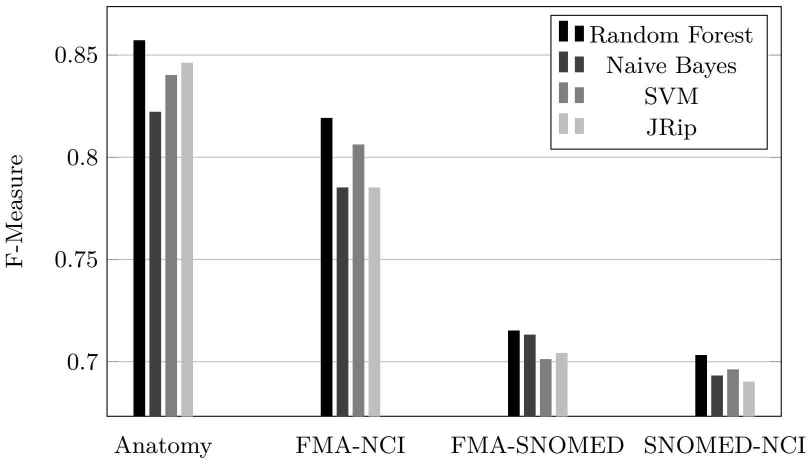


Figure 5.3: Comparing different machine learning algorithms for local matching

Anatomy track due to higher conceptual heterogeneity. We have deduced this behavior for the majority of ontology matching systems.

5.5 Resampling Local Training Sets Evaluation

We generated the set of local training sets and we chose the adequate machine learning algorithm for the local matching learning process. An additional step is required to enhance the accuracy of the matching process. This additional step corresponds to the resampling of the local training sets. We perform resampling in order to balance the training data, which contains a number of negative samples higher than the number of positive samples. Imbalanced training data is one of the main issues occurring while dealing with ontology matching learning. Imbalanced data typically refers to a classification problem where the number of observations per class is not equally distributed [76]. The purpose of the resampling process is to have an equal number of the majority class and the minority class. In our case, the majority class corresponds to the set of negative samples and the minority class corresponds to the set of positive samples. Therefore, local machine learning classifiers can learn wrong assumptions from imbalanced training sets. Therefore, for each local training set $ts_{ij,q}$ of $\mathcal{T}s_{ij,q}$ we apply resampling techniques. Different methods have been proposed by the state-of-the-art to resample imbalanced data [29]. As a result, we implement the architecture of Figure 4.1 but using different resampling techniques in order to study the obtained matching accuracy by each resampling technique. Resampling techniques are applied over the $\mathcal{T}s_{ij,q}$ local training sets. The goal of applying resampling techniques over each local training set $ts_{ij,q}$ is to result in better matching accuracy. We discuss the obtained results according to this criterion. We are based on the biomedical Anatomy track provided by OAEI benchmark for the evaluation process.

5.5.1 Impact of Undersampling on Local Training Data

We apply the widely employed undersampling method in order to balance the class distribution of the local training data. These methods are described as follows:

- **Random undersampling** is a non-heuristic method that aims to balance the class distribution through the random elimination of instances belonging to the majority class.

- **Tomek links** [105] removes the unwanted overlap between classes where majority class links are removed until all minimally distanced nearest-neighbor pairs are of the same class.

- **One-sided selection (OSS)** [59] aims at creating a training dataset composed only by ”safe instances”. Consequently, this technique removes instances that are noisy, redundant, or near the decision border. As the other undersampling techniques, OSS removes only instances from the majority class.

- **Edited Nearest Neighbors** [111] method removes the instances of the majority class with a prediction made by the K-means method. Therefore, if an instance has more neighbors of a different class, this instance will be removed.

In Table 5.2, we depict the results of each undersampling method in terms of the obtained accuracy. We deduce that Tomek links, One-sided selection, and Edited nearest neighbors obtained comparable results [63]. The Edited Nearest Neighbors resulted in the highest F-Measure of 85.3%. We deduce from Table 5.2 that the random undersampling resulted in the lowest F-measure (74.8%) as well as the lowest precision (65.9%). We argue this result due to the random feature of this undersampling method. This random feature negatively affect matching accuracy. Local training sets are balanced by randomly removing the instances of negative samples $\mathcal{NS}_{ijq}$ until being equal to the number of positive samples $\mathcal{PS}_{ijq}$. As a result the negative samples are not representative of the set of true negative and true positive that should be obtained by the learned classifiers. Therefore, the classifiers built from the balanced local training sets can result in a set of wrong correspondences (false negative and false positive). Consequently, the precision percentage is affected and became lower thant the other heuristic undersampling methods. The highest recall value is obtained by the random undersampling method. However, the recalled correspondences have lower precision than the other method. We can deduce that the number of returned correspondences by the random undersampling is higher than the other methods. Nevertheless, the returned correspondences contain a higher number of false-positive and false-negative. As a result, the precision value is lower than the other undersampling techniques but the recall is higher due to the high number of the returned correspondences. These correspondences contain a higher number of false negatives and false positives than the other undersampling techniques.

Table 5.2: Local Matching accuracy for each undersampling method

Undersampling method	Precision	Recall	F-Measure
Random Undersampling	65.9%	86.4%	74.8%
Tomek links	93.7%	77.4%	84.8%
One-sided selection	93.7%	77.1%	84.6%
Edited Nearest Neighbors	93.4%	78.4%	85.3%

5.5.2 Impact of Oversampling on Local Training Data

In this section, we perform the oversampling of the minority class instead of performing the undersampling of the majority class. There are several oversampling methods used in typical classification problems. The most common technique is known as SMOTE [20]: (Synthetic Minority Oversampling Technique). We perform the oversampling using the random oversampling method.

In the following, we briefly describe each of these methods:

- **SMOTE** oversamples the minority class by taking each positive instance and generating synthetic instances along a line segment joining their k-nearest neighbors of the minority class.

- **Random oversampling** is a non-heuristic method that aims to balance class distribution through the random elimination of instances belonging to the minority class.

In Table 5.3, we depict the results of the local matching $\mathcal{LM}_{ij}$ after performing the oversampling of the local training sets for each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. We show below the results of applying SMOTE with $k = 5$ as the number of nearest neighbors in order to achieve a ratio $r = 1$. This ratio corresponds to an equal number of positive samples $\mathcal{PS}_{ijq}$ and negative samples $\mathcal{NS}_{ijq}$. We also perform the random oversampling with a ratio r= 1

Table 5.3: Local Matching accuracy for each oversampling method

Oversampling method	Precision	Recall	F-Measure
Random oversampling	70.8%	82.8%	76.3%
SMOTE	86.7%	78.8%	82.6%

We deduce from Table 5.3 that SMOTE outperforms the random oversampling method in terms of precision and F-Measure. We argue this result due to the randomly generated instances by the random oversampling method. Random generated instances as negative samples are not generic enough to result in good classifiers. Therefore, a classifier can learn wrong assumptions from random negative samples. As a result, compared to the SMOTE oversampling, the random oversampling precision (70.8%) is negatively affected as we can see in Table 5.3. However, the recall percentage (82.8%) resulted from the random oversampling is better than SMOTE. We argue this result due to the high number of the returned correspondences by the random oversampling method. However, compared to SMOTE, random oversampling returns a higher number of false positives and false negatives. As a result, SMOTE has a better precision percentage (86.7%).

5.5.3 Impact of Oversampling combined to Undersampling on Local Training Data

It is possible to combine oversampling and undersampling techniques into a hybrid strategy. Performing a combination of oversampling and undersampling can often yield better results than either in isolation. [76] Common state-of-the-art methods [76, 19] include the combination of SMOTE and Tomek links, SMOTE and Edited Nearest Neighbors (ENN) or SMOTE and Random Undersampling. SMOTE is employed for the oversampling with a ratio r= 0.5 and combined with each of the latter undersampling techniques with a ratio r= 0.5. Therefore, we perform the oversampling of the minority class (positive samples) and the undersampling with the majority class (positive samples). We set for each sampling method a ratio of 0.5 in order result in a balanced training sets.

In the following we briefly describe each of the used undersampling methods:

- **SMOTE and Tomek links**, SMOTE is performed for the oversampling (r= 0.5) and Tomek links is employed for undersampling (r= 0.5).
- **SMOTE and Edited Nearest Neighbors (ENN)** SMOTE is used for the oversampling (r= 0.5) and Edited Nearest Neighbors is performed for undersampling (r= 0.5).
- **SMOTE and Random Undersampling** SMOTE is used to perform the oversampling with (r= 0.5) and Random undersampling with a ratio (r= 0.5) is employed for undersampling.

We evaluate these three methods by resampling the local training sets, then we perform the local matching process based on the generated local classifiers.

In the following Table 5.4, we depict the results of each combined method. We deduce that the combination of SMOTE and Tomek Link results in the best accuracy in terms of F-Measure and Precision. The combination of SMOTE and Random Under sampling results in the best recall value due to the random nature of this approach. Therefore, it returns the highest number of alignments with the lowest precision compared to the other combination methods.

According to the achieved results, we highlight the following points:

- The random undersampling and oversampling methods are unstable. A deeper study on the convergence of these methods can be investigated.
- We can observe in table 5.4 that combining SMOTE with undersampling methods decreases the matching accuracy.

Table 5.4: Impact of the combination Oversampling and Undersampling on Local Training Data

Hybrid method	Precision	Recall	F-Measure
SMOTE + Random Undersampling	87.8%	81.3%	84.4%
SMOTE + ENN	88.4%	80.5%	84.3%
SMOTE + Tomek Link	92.0%	79.0%	85.0%

We deduce that for the matching learning context, undersampling methods outperform the other resampling methods. We argue this result since the undersampling method removes redundant instances rather than creating new synthetic instances like the oversampling of the minority class. We conclude that the undersampling using ENN [111] yields to the best Precision and F-Measure. ENN removes instances of the majority class with a prediction made by the K-means method which is different from the majority class. For the generation of positive samples, we combined the use of cross-searching, cross-referencing and exact matching order to construct local training sets. The impact of each method on the resulted matching quality can be investigated. Some of the later methods can have a different impact in terms of the matching accuracy than the other positive samples generation methods. We tend to validate the hypothesis that the quality could depend on the use of methods used in the construction of the training data sets.

5.6 Local Feature Selection Evaluation

In this section, we apply the state-of-the-art feature selection methods over the generated training set for each local matching task of $\mathcal{LM}_{ij}$. Feature selection is one of the core concepts in machine learning which hugely impacts the performance of a machine learning model. Feature selection is the process where features are automatically or manually selected in the order to contribute to accurate predictions. Having irrelevant features in the training data can decrease the accuracy of the classifiers and make the classifiers learn based on irrelevant features. In our case, each feature corresponds to a matcher. We firstly study the impact of each of the employed feature selection methods. This comparison aims at choosing the adequate feature selection method in order to perform the local matching learning. Then, we compare the efficiency of the element level features and structure level features. This comparison aims at analyzing the obtained results in order to suggest new features for the local matching learning process.

5.6.1 Studying the Impact of Feature Selection Methods

In Figure 5.4, we compare the impact of three feature selection techniques on the local matching. These techniques are a filter-based method [41], a wrapper-based method [58] and an embedded-based method [87]. Each local classifier is built for each local training set based on its specific features, which are selected through feature selection methods.

According to the results of our experiments, for every matching task depicted in Figure 5.4, we can deduce that the wrapper feature selection method slightly outperforms the other employed local feature selection methods. The wrapper method considers the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A local classifier is used to evaluate a combination of features and assign a score based on the local classifier accuracy. However, filter-based feature selection method works in isolation of the local classifier. Therefore, the wrapper-based method outperforms the filter-based method.

In Figure 5.4, in the first bar chart of every experiment, we draw the accuracy of local matching $\mathcal{LM}_{ij}$ without applying any feature selection method (No FS). We deduct that applying feature selection improves the accuracy compared to no use of feature selection for local matching for different matching tasks. We argue these results due to the selection of the adequate features for every local-based classifier for a local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. Therefore, each local matching task is aligned through its specific local classifier, which captures the optimal settings (employed features, weight, and threshold) for the local matching context.

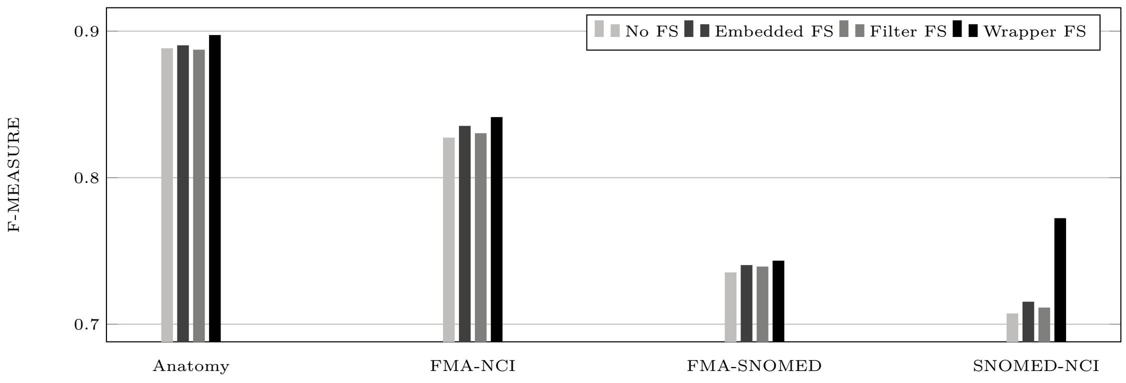


Figure 5.4: Evaluating the impact of feature selection methods

5.6.2 Impact of Element level and Structure level Features

In this section, we evaluate the impact of element level features compared to structure level features. This evaluation consists of studying the performance of each used feature level separately and combined. Thus, we discuss the obtained results and we deduce the limitations of the employed features.

We perform three experiments using the Anatomy track of OAEI 2018. For each experiment, we run the local matching process, without feature selection methods, while varying the employed features. These three experiments are summarized as follows.

1. In the first experiment, we perform the local matching using only element level features.
2. In the second experiment, we use only the structure level features for performing the local matching.

3. In the third experiment, we perform the local matching using both element level features and structure level features.

In Figure 5.5, we draw the obtained accuracy for each experiment based on the 1516 reference alignments of the Anatomy track provided by OAEI 2018. The local matching using the element level and structure level features achieved an F1-measure of 0.836, which is better than the accuracy of the local matching using only element level features (0.802 F1-Measure) and the local matching with only structure features (0.503 F1-Measure). We can deduce that the combination of structure level features and element level features is beneficial for the local matching learning process in terms of accuracy. We deduce from the result of the local matching using only element level features that the majority of the correspondences of the reference alignments are based on terminological heterogeneity more than the conceptual heterogeneity.

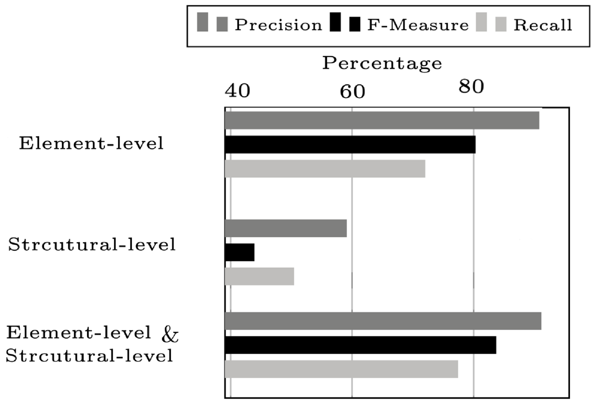


Figure 5.5: The accuracy of Element level based features compared to the structure level based features.

In order to analyze in-depth the obtained results, we compare the true positive alignments resulted from each experiment. The Venn diagram of Figure 5.6 depicts all the intersections between the obtained true positive alignments of each experiment and the reference alignments. Overlapping circles reveal common alignments obtained by two or more groups. We can observe in Figure 5.6 that no experiment is able to find 202 alignments, which represents 13% of the reference alignments. This can be argued due to the high heterogeneity of the 202 alignments. We can also observe that the local matching is not able to find 22% (13% + 9%) of the reference alignments. The local matching finds all the alignments discovered by the element level alignments. In future work, we plan to define new features to find the rest of the alignments (22%). For instance, we can define more features in order to discover more alignments. Finally, the 0% depicted in Figure 5.6 is resulted due to the absence of unique alignments discovered exclusively by the element level features and not the local matching. Thus, the local matching combining both feature levels discovers all the alignments discovered by the element level local matching. The local matching classifiers, which correspond to the combination of the element level features and structure level features result in discovering 29 unique alignments. These 29 alignments are neither discovered separately by the element feature level features nor the structure level features, which proves the importance of combining both types of features.

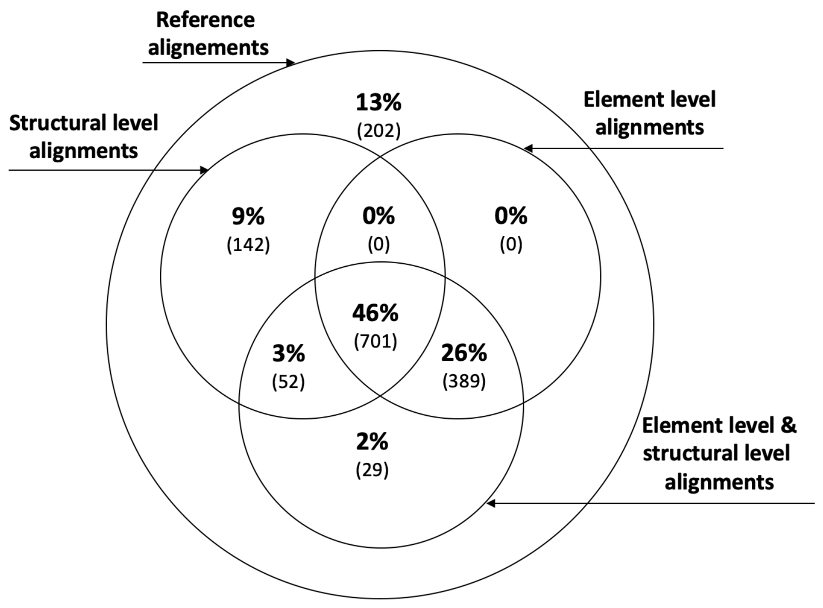


Figure 5.6: Venn diagram of alignments resulted from local matching based on the element level features, the structure level features and their combination.

We can deduce after performing the feature selection for local training sets that the wrapper feature selection methods are the most efficient method since it resulted in the best matching accuracy. We argue the obtained results due to the interaction process that this method makes with the classifier. Therefore each set of selected features are adequate to the employed machine learning algorithm. Compared to the filter method, wrapper feature selection conducts a search for a good subset of features using the learning algorithm itself as part of the evaluation function. In our case, the learning algorithm is Random Forest. Therefore, the wrapper feature selection makes the interaction between feature subset search and model selection and is able to take into account feature dependencies.

After studying the impact of each feature group in separation versus their combination, we can deduce that the combination of structural level features and element level features is beneficial. For instance, the combination of these features while performing local matching learning increases the number of the true positive alignments by 2%, which correspond to 29 alignments for the Anatomy matching the task. Moreover, we can deduce that 13% of the alignments are not discovered. Consequently, new features should be proposed in our feature work in order to discover these alignments.

5.7 Comparing Local Matching to Global Matching

As depicted in Figure 5.7, we conducted four experiments using the ontology matching tasks of OAEI 2018. The goal of these experiments is to compare the local matching $\mathcal{LM}_{ij}$ accuracy to the global matching $\mathcal{GM}_{ij}$ accuracy. The accuracy is computed using the F1-Measure according to the reference alignments, which are provided by OAEI 2018 for each ontology matching track.

The global matching $\mathcal{GM}_{ij}$ is the alignment of two input ontologies using a single global machine learning classifier. This classifier is generated using one global training set. We automatically build a global training set by the same method used to build local training sets. We perform the local matching $\mathcal{LM}_{ij}$ between the two input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ of each task. To accomplish that, we begin by performing the partitioning of each matching task to generate a set of local matching tasks $\mathcal{LM}_{ij}$. For each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$, we generate a local training set $ts_{ij,q}$ using the features of Table 4.2 and Table 4.3. Each local classifier $lcl_{ij,q}$ is built using its specific local training set $ts_{ij,q}$. Each local classifier classifies the entity pairs ($\mathcal{V}_{i,q}$ $\times$V$_{j,q}$) of its local matching task $lm_{ij,q}$ to be aligned or not. The final set of alignments for each local matching task $lm_{ij,q}$ are summed up in order to generate the final set of alignments for the ontology matching task. Therefore, we combine the correspondences of each local matching task into one alignment set to be compared to the reference alignments. This comparison results in the accuracy of the local matching learning approach.

We can observe that the local matching significantly outperforms the global matching for each matching task. For some tracks, we achieved comparable results to the top matching system AML [37] and LogMap [53]. AML manually defines the threshold for the employed similarity matchers. AML and LogMap have been the top-ranked OAEI matching systems for several years. AML defines a syntactic threshold of 0.6. This threshold is updated based on the size of the matching task. However, our local matching approach is completely automated with no user involvement for setting the matching parameters. Nonetheless, our approach fixes automatically the adequate threshold value for every employed feature along with the weight of each feature for every local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$.

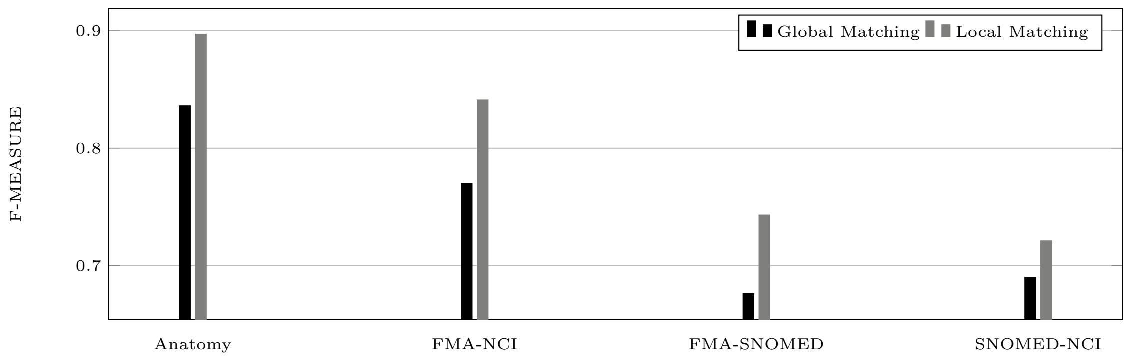


Figure 5.7: Comparing local matching to global matching

5.8 POMap++: Local Matching Learning Evaluation

We are based on the OAEI 2018 datasets in order to asses the efficiency of the POMap++. POMap++ implements the partitioning and the local matching learning approach in order to align large biomedical ontologies. The architecture of POMap++ is depicted in Figure 5.1. Its has as input two ontologies and results in a set of correspondences. These alignments are compared to the reference alignment provided by the OAEI 2018 benchmark. We are mainly based on the biomedical datasets for the evaluation of POMap++. Therefore, we evaluate POMap++ based on the two biomedical tracks: Anatomy and LargeBio. In the following tables, we report the obtained results for each matching task. We have achieved good results for the alignment of biomedical ontologies. This approach generates matching setting automatically for each matching task. Therefore, it can be adapted for every biomedical ontology matching domain.

Table 5.5: Local matching learning evaluation for the Anatomy track

Matching system	F-Measure	Precision	Recall
AML	0.943	0.95	0.936
LogMapBio	0.898	0.888	0.908
POMap++	**0.897**	**0.919**	**0.877**
XMap	0.896	0.929	0.865
LogMap	0.88	0.918	0.846
SANOM	0.865	0.888	0.844
FCAMapX	0.859	0.941	0.791
KEPLER	0.836	0.958	0.741
Lily	0.832	0.872	0.795
LogMapLite	0.828	0.962	0.728
ALOD2Vec	0.785	0.996	0.648
DOME	0.761	0.997	0.615
ALIN	0.758	0.998	0.611
Holontology	0.451	0.976	0.294

Table 5.6: Local matching learning evaluation for the FMA-NCI matching task

Matching system	F-Measure	Precision	Recall
AML	0.855	0.838	0.872
POMap++	**0.841**	**0.860**	**0.822**
LogMap	0.831	0.856	0.808
LogMapBio	0.830	0.830	0.831
XMap	0.804	0.878	0.742
FCAMapX	0.743	0.665	0.841
LogMapLite	0.741	0.676	0.819
DOME	0.729	0.803	0.668

Table 5.7: Local matching learning evaluation the FMA-SNOMED matching task

Matching system	F-Measure	Precision	Recall
FCAMapX	0.789	0.819	0.762
AML	0.772	0.882	0.687
POMap++	**0.743**	**0.829**	**0.673**
LogMapBio	0.731	0.834	0.650
LogMap	0.730	0.840	0.645
XMap	0.661	0.723	0.608
LogMapLite	0.334	0.851	0.208
DOME	0.326	0.941	0.197

Table 5.8: Local matching learning evaluation for the NCI-SNOMED matching task

Matching system	F-Measure	Precision	Recall
AML	0.768	0.904	0.668
FCAMapX	0.733	0.796	0.680
LogMapBio	0.723	0.854	0.627
POMap++	**0.721**	**0.851**	**0.625**
LogMap	0.706	0.867	0.596
LogMapLite	0.662	0.798	0.566
DOME	0.632	0.907	0.485
XMap	0.610	0.640	0.582

We can deduce from the later table that POMap++ ranked among the top ontology matching systems for Anatomy and LargeBio matching tracks. We mention that our approach automatically defines the matching setting for both element level matchers and structural level matchers. 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 thresholds values for every employed matcher 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. We can deduce from the obtained results that the local matching approach obtained a better accuracy than the global matching approach due to the reduced search space by dividing each ontology matching task into a set of local matching tasks. The search space for an ontology matching task is reduced from all the entities-pairs $\mathcal{V}_i \times \mathcal{V}_j$ of two input ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ to the set of entity-pairs $\mathcal{V}_{i,q}$ $\times \mathrm{V}_{j,q}$ of each local matching task $lm_{ij,q}$ of $\mathcal{LM}_{ij}$. Therefore, the percentage of wrong alignments (false positive) is reduced by the local matching approach. Moreover, each local matching classifier captures the characteristics of a local matching task than a single global matching classifier for a large ontology matching task. Thus, the local matching $\mathcal{LM}_{ij}$ has a better recall than the global matching process $\mathcal{GM}_{ij}$.

5.9 Conclusion

In this chapter, we have presented the evaluation of the ontology partitioning approach and the local matching learning approach. Our partitioning approach outperformed the existing systems. In order to asses the local matching learning approach, we have compared this approach to the state-of-the-art matching systems as well as the global matching systems. POMap++ is ranked among the top ontology matching systems for the different biomedical matching tracks. Moreover, the comparison between the local matching and the global matching has shown that our proposed approach outperforms the global matching. Global matching corresponds to the use of machine learning techniques without employing any partitioning strategy. We can deduce that dividing a large ontology matching task into a set of sub-matching tasks, and aligning each sub-matching task based on its specific classifier yields to promising results. Therefore, each local classifier takes into consideration the heterogeneity of each sub-matching context in the generation of local alignments. Consequently, the overall alignment results are improved. Moreover, we have compared the state-of-the-art resampling techniques in order to resample the unbalanced training sets. Results show that random undersampling and random oversampling methods are unstable compared to the other methods. Furthermore, combining SMOTE with undersampling methods decreases the matching accuracy. We also studied the efficiency of feature selection methods. Therefore, we compared the state-of-the-art feature selection methods for selecting the adequate feature for the training sets. The comparison state that the wrapper feature selection outperforms the other methods for performing local matching learning. Besides the comparison of feature selection methods, we have compared structural level features to element level features. This comparison aims to study the behavior of each type of features as well as finding a way to improve the obtained results by thinking about adding more types of features. As a result, a study of the alignment not found by both type of feature may yield to propose new features. We consider this task as future work.

Chapter 6

Conclusion and Future Work

Ontologies are the backbone of the semantic web. They enable sharing, reusing and accessing the knowledge resources. Hundreds of large 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 [101]. Large biomedical ontologies usually describe the same domain of interest but using different modeling standpoints and vocabularies. Hence, aligning these complex and heterogeneous ontologies is a cumbersome task. Matching systems should provide high-quality results while dealing with the large size of these resources. Therefore, an ontology matching system should be based on different matchers in order to cope with the high heterogeneity of the matching tasks. Different matchers should be combined to result in better matching accuracy. These matchers should not depend on a single configuration for all the matching problems. Hence, the matching tuning process should be automatically configured for different matching tasks. State-of-the-art ontology matching systems need to cope with two major issues for the alignment of large ontologies: (i) integrating the large size not yet feasible with a good matching accuracy, (ii) automating the ontology matching process.

The integration of large ontologies requires efficient matching systems supporting the complex ontologies [1, 35, 89]. For instance, the cartesian product between the entities of NCI and SNOMED ontologies results in more than 45 billion comparisons. As a result, an agile solution is required. Ontology mapping becomes a challenging task due to the large size of ontologies. Large ontologies are characterized by high conceptual heterogeneity. Consequently, the discovery of mappings between large ontologies became more challenging. As a result, ontology matching systems combine different matchers to cope with these issues. These matchers are usually based on the predefined matching configuration. However, each domain ontology should be aligned based on its characteristics, which should be taken into account by the matching settings. To sum up, the major issues of the alignment of large biomedical ontologies are the conceptual heterogeneity, the high search space and the decreased quality of the resulted alignments.

Usually, ontology matching systems combine different matcher in order to cope with the high heterogeneity of large matching tasks. This combination should define at least the choice of matchers to combine and the weight of each matcher. Different matchers treat different types of heterogeneity. There is no effective combination of all ontology matching tasks. Each matching task has its appropriate specificity. Therefore, the matching tuning should be well defined by ontology matching systems in order to result in good matching accuracy. The manual configuration of the matching setting is complex, especially for large ontology matching tasks. Consequently, matching tuning should be automatically defined for the matching process. The ontology matching process should be self-tuned for an automatic selection of the matching settings.

We have proposed an approach to cope both with the large size of ontologies and the automation issue. This approach is called local matching learning. We divide a large matching problem into a set of smaller local matching problems. Each local matching problem is independently aligned based on a matching learning approach. Therefore, we reduce the huge search space into a set of smaller local matching tasks. As a result, we can align effectively each local matching task to result in better matching accuracy.

Our proposed partitioning approach is based on a novel multi-cut strategy generating non-large partitions or non-isolated ones. As a result, we can overcome the issue of conceptual heterogeneity. The novel partitioning algorithm is based on Hierarchical Agglomerative Clustering. This approach generates a set of local matching tasks with a sufficient coverage ratio and no isolated partitions. The partitioning process follows three steps: (i) pre-processing and partitioning input ontologies, (ii) applying the partitioning algorithm and (iii) identification of local matching tasks. We aligned each local matching task using its specific local-based classifier.

Each local matching task is automatically aligned based on machine learning techniques. A local classifier aligns a single ontology matching task. Local classifiers are based on element and structure level features and built using a specific local training set. The class attribute for each local training set is automatically labeled using an external knowledge base to generate an adequate classifier for each local matching context. We applied a wrapper-feature selection technique across each local classifier. This local feature selection technique ensures that adequate matchers are selected for each local matching context. This approach decreases the complexity of an ontology matching task and increases the overall accuracy compared to traditional matching learning approaches.

We have proved that the partitioning approach outperforms the existing state-of-the-art approaches in terms of accuracy, coverage ratio and the absence of isolated partitions. We have evaluated the proposed local matching learning approach using various experiments. Firstly, we compared the local matching approach and the global matching approach using six datasets from the OAEI 2018 benchmark. We deduced that local matching outperforms global matching in terms of accuracy. Therefore, dividing a large ontology matching task into a set of local matching tasks is beneficial. We can affirm this result on account of the reduced search space, which results in fewer false negatives and false positives. Secondly, we tested the impact of resampling and feature selection techniques. Resampling of local training sets yields better results. Applying feature selection techniques across each local classifier using its local training set increases the recall value for each local matching context. Finally, we studied the impact of element level features compared to structure level features. We show that additional alignments are captured after combining two feature levels in one local classifier.

As an immediate future work, we plan to investigate additional features, such as semantic features, in particular word embedding approaches. Word embedding approaches are effective at capturing language semantics. They have been proposed for semantic matching in some of the existing state-of-the-art ontology alignment systems. Semantic matching approaches do not always outperform string-based similarity and effectively combining both strategies in a matching learning alignment system remain a challenge. We plan also to integrate features based on profiling metrics [27]. Profiling metrics can be employed in order to automatically optimize the system's configuration depending on the characteristics of the ontologies to be matched. Profiling metrics can investigate several features, such as the average number of attributes per class, the average number of subclasses per class and the average distribution of instances across all the classes.

Moreover, we plan to extend our approach for matching learning to other domain ontologies besides the biomedical domain. Therefore, we intend to enlarge the automatic generation of the training data for different domain ontologies. Generating context-aware training data for different domains without employing the Gold Standard to perform the labeling is a cumbersome problem that should be resolved. To resolve this issue, we can extract the labeled data from external knowledge sources. Ontology repositories in order to extract the labeled data using ontology repositories. These repositories offer facilities to access the stored domain ontologies. Therefore, related domain ontologies can be explored to extract labeled data

Most existing approaches are still limited to pairwise matching [94]. However, in complex domains where several ontologies describing different but related aspects of the domain have to be linked together, matching multiple ontologies simultaneously, known as holistic matching, is required [94]. Therefore, as a long term future work, we plan to extend the matching learning approach to perform the holistic matching. The training data should be adapted to this type of alignment. For instance, we plan to extend the set of feature for the training data with some characteristics related to holistic matching tasks.

Bibliography

[1] Manel Achichi, Michelle Cheatham, Zlatan Dragisic, Jérôme Euzenat, Daniel Faria, Alfio Ferrara, Giorgos Flouris, Irini Fundulaki, Ian Harrow, Valentina Ivanova, et al. Results of the ontology alignment evaluation initiative 2017? In *12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference*, volume 2032, pages 61–113. CEUR-WS, 2017.

[2] Sareh Aghaei, Mohammad Ali Nematbakhsh, and Hadi Khosravi Farsani. Evolution of the world wide web: From web 1.0 to web 4.0. *International Journal of Web & Semantic Technology*, 3(1):1, 2012.

[3] Asan Agibetov, Giuseppe Patanè, and Michela Spagnuolo. Grontocrawler: Graph-based ontology exploration. 2015.

[4] Alsayed Algergawy, Samira Babalou, Mohammad J. Kargar, and S. Hashem Davarpanah. Seecont: A new seeding-based clustering approach for ontology matching. In *Advances in Databases and Information Systems - 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings*, pages 245–258, 2015.

[5] Alsayed Algergawy, Sabine Massmann, and Erhard Rahm. A clustering-based approach for large-scale ontology matching. In *Advances in Databases and Information Systems - 15th International Conference, ADBIS 2011, Vienna, Austria, September 20-23, 2011. Proceedings*, pages 415–428, 2011.

[6] Emna Ammar El Hadj Amor and Sonia Ayachi Ghannouchi. Toward an ontology-based model of key performance indicators for business process improvement. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 148–153. IEEE, 2017.

[7] Amina Annane. *Using Background Knowledge to Enhance Biomedical Ontology Matching.* PhD thesis, Université Montpellier; Ecole Nationale Supérieure d'Informatique (ESI)-Alger, 2018.

[8] Amina Annane, Zohra Bellahsene, Faiçal Azouaou, and Clement Jonquet. Yam-bio–results for oaei 2017. 2017.

[9] Howard Anton. Elementary linear algebras. page 170–171, 1994.

[10] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[11] David Aumueller, Hong Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, pages 906–908, 2005.

[12] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, pages 315–329. 2013.

[13] Alain Berro, Imen Megdiche, and Olivier Teste. Holistic statistical open data integration based on integer linear programming. In *9th IEEE International Conference on Research Challenges in Information Science, RCIS 2015, Athens, Greece, May 13-15, 2015*, pages 468–479, 2015.

[14] Alain Berro, Imen Megdiche, and Olivier Teste. A linear program for holistic matching: Assessment on schema matching benchmark. In *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part II*, pages 383–398, 2015.

[15] Paul E Black. Manhattan distance"" dictionary of algorithms and data structures. *http://xlinux. nist. gov/dads//*, 2006.

[16] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[17] Vrdoljak Boris. Cromatcher-results for oaei 2016. In *The 11th International Workshop on Ontology Matching, OM-2016*, 2016.

[18] Said Broumi and Florentin Smarandache. *Cosine similarity measure of interval valued neutrosophic sets*. Infinite Study, 2014.

[19] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 875–886. 2010.

[20] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

[21] Agnese Chiatti, Zlatan Dragisic, Tania Cerquitelli, and Patrick Lambrix. Reducing the search space in ontology alignment using clustering techniques and topic identification. In *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, Palisades, NY, USA, October 7-10, 2015*, pages 21:1–21:4, 2015.

[22] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *ACM Sigmod Record*, 35(3):34–41, 2006.

[23] Watson Wei Khong Chua and Jung-Jae Kim. Eff2match results for oaei 2010. *Ontology Matching*, 150:105–106, 2010.

[24] Gene Ontology Consortium. The gene ontology (go) database and informatics resource. *Nucleic acids research*, 32(suppl_1):D258–D261, 2004.

[25] Isabel F Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, 2(2):1586–1589, 2009.

[26] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Agreementmaker: Efficient matching for large real-world schemas and ontologies. *PVLDB*, 2(2):1586–1589, 2009.

[27] Isabel F. Cruz, Alessio Fabiani, Federico Caimi, Cosmin Stroe, and Matteo Palmonari. Automatic configuration selection using ontology matching task profiling. In *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, pages 179–194, 2012.

[28] Jérôme David. Aroma results for oaei 2009. In *Proc. 4th ISWC workshop on ontology matching (OM)*, pages 147–152. No commercial editor., 2009.

[29] Marcílio Carlos Pereira de Souto, Valnaide G. Bittencourt, and José Alfredo Ferreira Costa. An empirical analysis of under-sampling techniques to balance a protein structural class dataset. In *Neural Information Processing, 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006, Proceedings, Part III*, pages 21–29, 2006.

[30] Warith Eddine Djeddi, Sadok Ben Yahia, and Mohamed Tarek Khadir. Xmap: results for OAEI 2018. In *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.*, pages 210–215, 2018.

[31] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In *Handbook on ontologies*, pages 385–403. Springer, 2004.

[32] Kevin Donnelly. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006.

[33] Kai Eckert, Christian Meilicke, and Heiner Stuckenschmidt. Improving ontology matching using meta-level learning. In *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, pages 158–172, 2009.

[34] Marc Ehrig and Steffen Staab. Qom–quick ontology mapping. In *International Semantic Web Conference*, pages 683–697. Springer, 2004.

[35] Jérôme Euzenat and Pavel Shvaiko. Ontology matching, vol. 1, 2007.

[36] Daniel Faria, Catia Pesquita, Isabela Mott, Catarina Martins, Francisco M. Couto, and Isabel F. Cruz. Tackling the challenges of matching biomedical ontologies. *J. Biomedical Semantics*, 9(1):4:1–4:19, 2018.

[37] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, pages 527–541, 2013.

[38] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thesaurus and ontology. *Journal of Web Semantics First Look 1_1_4*, 2003.

[39] Anika Groß, Michael Hartung, Toralf Kirsten, and Erhard Rahm. GOMMA results for OAEI 2012. In *Proceedings of the 7th International Workshop on Ontology Matching, Boston, MA, USA, November 11, 2012*, 2012.

[40] Tom Gruber. What is an ontology. *WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)*, 1993.

[41] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 359–366, 2000.

[42] Fayçal Hamdi, Brigitte Safar, Nobal B Niraula, and Chantal Reynaud. Taxomap alignment and refinement modules: Results for oaei 2010. *Ontology Matching*, page 212, 2010.

[43] Sven Hertling and Heiko Paulheim. Dome results for oaei 2018. In *OM@ ISWC*, pages 144–151, 2018.

[44] Hu. Matching large ontologies: A divide-and-conquer approach. *Data Knowl. Eng.*, 67(1):140–160, 2008.

[45] Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke. Codi: Combinatorial optimization for data integration–results for oaei 2011. *Ontology Matching*, 134, 2011.

[46] Ryutaro Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *7th IEEE/ACIS International Conference on Computer and Information Science, IEEE/ACIS ICIS 2008, 14-16 May 2008, Portland, Oregon, USA*, pages 340–346, 2008.

[47] Valentina Ivanova and Patrick Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *Extended Semantic Web Conference*, pages 1–15. Springer, 2013.

[48] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[49] Yves R Jean-Mary, E Patrick Shironoshita, and Mansur R Kabuka. Asmov: Results for oaei 2010. *Ontology Matching*, 126:2010, 2010.

[50] Qiu Ji, Peter Haase, and Guilin Qi. Combination of similarity measures in ontology matching using the owa operator. In *Recent Developments in the Ordered Weighted Averaging Operators: Theory and Practice*, pages 281–295. Springer, 2011.

[51] Ernesto Jiménez-Ruiz, Asan Agibetov, Matthias Samwald, and Valerie Cross. We divide, you conquer: from large-scale ontology alignment to manageable subtasks with a lexical index and neural embeddings. In *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.*, pages 13–24, 2018.

[52] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[53] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[54] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *European Semantic Web Conference*, pages 185–199. Springer, 2008.

[55] Marouen Kachroudi, Gayo Diallo, and Sadok Ben Yahia. Kepler at oaei 2018. In *OM@ ISWC*, pages 173–178, 2018.

[56] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The knowledge engineering review*, 18(1):1–31, 2003.

[57] Jean François Djoufak Kengue, Jérôme Euzenat, and Petko Valtchev. Ola in the oaei 2007 evaluation contest. In *Proceedings of the 2nd International Conference on Ontology Matching-Volume 304*, pages 188–195. Citeseer, 2007.

[58] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[59] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Nashville, USA, 1997.

[60] Amir Laadhar, Faïza Ghozzi, Imen Megdiche, Franck Ravat, and Olivier Teste. Partitioning and matching tuning of large biomedical ontologies. In *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.*, pages 220–221, 2018.

[61] Amir Laadhar, Faïza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faïez Gargouri. Pomap: An effective pairwise ontology matching system. In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - (Volume 2), Funchal, Madeira, Portugal, November 1-3, 2017.*, pages 161–168, 2017.

[62] Amir Laadhar, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faiez Gargouri. Pomap results for oaei 2017. 2017.

[63] Amir Laadhar, Faïza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faïez Gargouri. The impact of imbalanced training data on local matching learning of ontologies. In *Business Information Systems - 22nd International Conference, BIS 2019, Seville, Spain, June 26-28, 2019, Proceedings, Part I*, pages 162–175, 2019.

[64] Amir Laadhar, Faïza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faïez Gargouri. Partitioning and local matching learning of large biomedical ontologies. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 2285–2292, 2019.

[65] Amir Laadhar, Faiza Ghozzi, Imen Megdiche Bousarsar, Franck Ravat, Olivier Teste, and Faiez Gargouri. Oaei 2018 results of pomap++. CEUR-WS: Workshop proceedings, 2018.

[66] Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification. 1998.

[67] Bach Thanh Le, Rose Dieng-Kuntz, and Fabien Gandon. On ontology matching problems. *ICEIS (4)*, pages 236–243, 2004.

[68] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[69] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[70] Paul Mcnamee and James Mayfield. Character n-gram tokenization for european language text retrieval. *Information retrieval*, 7(1-2):73–97, 2004.

[71] Imen Megdiche. *Intégration holistique et entreposage automatique des données ouvertes. (Holistic integration and automatic warehousing of open data)*. PhD thesis, Paul Sabatier University, Toulouse, France, 2015.

[72] Imen Megdiche, Olivier Teste, and Cássia Trojahn dos Santos. LPHOM results for OAEI 2016. In *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.*, pages 190–195, 2016.

[73] Imen Megdiche, Olivier Teste, and Cassia Trojahn. An extensible linear approach for holistic ontology matching. In *International Semantic Web Conference*, pages 393–410. Springer, 2016.

[74] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th International Conference on Data Engineering*, pages 117–128. IEEE, 2002.

[75] Alvaro E. Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 267–270, 1996.

[76] Ajinkya More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *CoRR*, abs/1608.06048, 2016.

[77] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *CoRR*, abs/1109.2378, 2011.

[78] Christopher J Mungall, Carlo Torniai, Georgios V Gkoutos, Suzanna E Lewis, and Melissa A Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome biology*, 13(1):R5, 2012.

[79] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[80] Azadeh Haratian Nezhadi, Bita Shadgar, and Alireza Osareh. Ontology alignment using machine learning techniques. *International Journal of Computer Science & Information Technology*, 3(2):139, 2011.

[81] Duy Hoa Ngo. *Enhancing ontology matching by using machine learning, graph matching and information retrieval techniques.* PhD thesis, 2012.

[82] DuyHoa Ngo and Zohra Bellahsene. Overview of YAM++ - (not) yet another matcher for ontology alignment task. *J. Web Semant.*, 41:30–49, 2016.

[83] Ikechukwu Nkisi-Orji, Nirmalie Wiratunga, Stewart Massie, Kit-Ying Hui, and Rachel Heaven. Ontology alignment based on word embedding and random forest classification. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, pages 557–572, 2018.

[84] Natalya F Noy and Mark A Musen. The prompt suite: interactive tools for ontology merging and mapping. *International journal of human-computer studies*, 59(6):983–1024, 2003.

[85] Peter Ochieng and Swaib Kyanda. Large-scale ontology matching: State-of-the-art analysis. *ACM Computing Surveys (CSUR)*, 51(4):75, 2018.

[86] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.

[87] Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

[88] Sunny Pereira, Valerie Cross, and Ernesto Jiménez-Ruiz. On partitioning for ontology alignment. In *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.*, pages 219–220, 2017.

[89] Erhard Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer, 2011.

[90] Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[91] Quentin Riché-Piotaix, Patrick Girard, Frédéric Bilan, and Ladjel Bellatreche. Example based programming and ontology building: A bioinformatic application. In *HCI International 2018 - Posters' Extended Abstracts, 20th International Conference, HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part I.*, pages 101–108, 2018.

[92] Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. More: Modular combination of owl reasoners for ontology classification. In *International Semantic Web Conference*, pages 1–16. Springer, 2012.

[93] Cornelius Rosse and José LV Mejino Jr. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of biomedical informatics*, 36(6):478–500, 2003.

[94] Philippe Roussille, Imen Megdiche, Olivier Teste, and Cássia Trojahn. Boosting holistic ontology matching: Generating graph clique-based relaxed reference alignments for holistic evaluation. In *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings*, pages 355–369, 2018.

[95] Philippe Roussille, Imen Megdiche, Olivier Teste, and Cássia Trojahn. Holontology: results of the 2018 OAEI evaluation campaign. In *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.*, pages 167–172, 2018.

[96] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[97] Md Hanif Seddiqui and Masaki Aono. An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):344–356, 2009.

[98] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251, 2007.

[99] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34, 1948.

[100] Giorgos Stoilos, Giorgos B. Stamou, and Stefanos D. Kollias. A string metric for ontology alignment. In *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, pages 624–637, 2005.

[101] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.

[102] Heiner Stuckenschmidt and Anne Schlicht. Structure-based partitioning of large ontologies. In *Modular ontologies*, pages 187–210. Springer, 2009.

[103] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[104] Boontawee Suntisrivaraporn, Guilin Qi, Qiu Ji, and Peter Haase. A modularization-based approach to finding all justifications for owl dl entailments. In *Asian Semantic Web Conference*, pages 1–15. Springer, 2008.

[105] Ivan Tomek. Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, 6:769–772, 1976.

[106] MK Vijaymeena and K Kavitha. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28, 2016.

[107] Lucy Lu Wang, Chandra Bhagavatula, Mark Neumann, Kyle Lo, Chris Wilhelm, and Waleed Ammar. Ontology alignment in the biomedical domain using entity definitions and context. In *Proceedings of the BioNLP 2018 workshop, Melbourne, Australia, July 19, 2018*, pages 47–55, 2018.

[108] Peng Wang and Baowen Xu. Lily: Ontology alignment results for oaei 2008. In *Proceedings of the 3rd International Conference on Ontology Matching-Volume 431*, pages 167–175. CEUR-WS. org, 2008.

[109] Zhichun Wang, Xiao Zhang, Lei Hou, Yue Zhao, Juanzi Li, Yu Qi, and Jie Tang. Rimom results for oaei 2010. *Ontology Matching*, 195, 2010.

[110] MS Waterman. Identification of common molecular subsequence. *Mol. Biol*, 147:195–197, 1981.

[111] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

[112] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[113] Zhibiao Wu and Martha Stone Palmer. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics, 27-30 June 1994, New Mexico State University, Las Cruces, New Mexico, USA, Proceedings.*, pages 133–138, 1994.

[114] Xingsi Xue and Jeng-Shyang Pan. A segment-based approach for large-scale ontology matching. *Knowl. Inf. Syst.*, 52(2):467–484, 2017.

[115] Chee Een Yap and Myung Ho Kim. Instance-based ontology matching with rough set features selection. In *2013 International Conference on IT Convergence and Security, ICITCS 2013, Macau, China, December 16-18, 2013*, pages 1–4, 2013.