DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND RESEARCH

SAAD DAHLEB BLIDA-1 UNIVERSITY
FACULTY OF SCIENCES
Computer Science Department

DOCTORAL THESIS

Option: Data Sciences and Computing

Semantic Matching of Big Ontologies

Author:

Meriem Ali Khoudja

Jury:

Nadjia Benblidia	Professor	University of Blida 1	President
Narhimene Boustia	Professor	University of Blida 1	Examiner
Leila Zemmouchi	Associate professor	ENST of Algiers	Examiner
Hafida Bouarfa	Professor	University of Blida 1	Supervisor
Messaouda Fareh	Associate professor	University of Blida 1	Co-supervisor

June, 2023

إلى أمي.. جنتي...

وأبي.. عطرها...

أهدي هذا العمل

Acknowledgments

Alhamdulillah. All praises to ALLAH, the Almighty for without His graces and blessings, this study would not have been possible. All thanks to ALLAH for giving me the wisdom, strength, support and knowledge in exploring things, for all the opportunities and the guidance in helping surpass all the trials that I encountered, and for giving me determination to complete this assignment. My humblest gratitude to the holy Prophet Mohamed (Peace be upon him) whose way of life has been a continuous guidance for us from wickedness to the truth of Islam.

I am forever indebted to my family who, understanding the importance of this work, suffered my hectic working hours. Special thanks are conveyed to them for the sincere love and support they have shown to me throughout my entire life. I eagerly await the day when I can share the joy of this achievement with them.

I would like to earnestly thank my loving parents, Mohamed and Nadjet, for their moral encouragement, active participation and non-stop assistance, as well as for every effort that they have done to support and encourage me in every path I take. May ALLAH gives chance for me to make them happy more than what they did.

My deepest appreciation is presented to my brothers, Abderraouf, Amine and Abdesselam, and to my sister, Hadjer, for their support, helping and encouragements that they have spent for me during the process of realizing this project. May ALLAH shower them with success and honour in their lives.

My sincere gratitude goes to my husband, Mounir, for his time, patience and endless help to finish this research. I express my sincerest appreciation for the strength and motivation from him, and for his assistance in any way that I may have asked.

I would like to deeply thank my unborn baby for bearing the potential impact of my rigorous work schedule and sleepless nights. Its presence has been a constant reminder of the importance of perseverance and dedication to pursue my academic goals.

I acknowledge my supervisor Pr. Hafida Bouarfa and my co-supervisor Dr. Messaouda Fareh for the continuous support and guidance throughout the preparation of this research, with their immense knowledge ant their directions.

I would sincerely like to thank the jury members for accepting to review this thesis, and for their constructive and valuable comments.

I warmly thank my bountiful friend, Aicha, who have ever right by my side until the completion of this study. I really appreciate her encouragements, never-ending support and meaningful advices during this work.

I am also deeply indebted to my former supervisor, Dr. A.C Mazari, for his invaluable advice and his effort in giving me clear insights, suggestions and recommendations to improve this study.

Last but not least, I would like to extend my deepest sincerest gratitude to all the people who helped me in any manner in order to make this research a reality. Nothing can reward their generosity and helping, but ALLAH can give them both the greatest rewards in the world and the Day After.

Meriem ALI KHOUDJA

Abstract

Ontology matching is an efficient method to establish interoperability among heterogeneous ontologies. Large-scale ontology matching still remains a big challenge for its long time and large memory space consumption. The actual solution to this problem is ontology partitioning which is also challenging.

Artificial neural networks are powerful computational models biologically inspired from the human brain, and the way how they learn and process information. Deep learning is a promising avenue of research and an important step toward artificial intelligence, emulating the human brain's mechanisms especially for extremely complex problems. Deep learning techniques have been particularly successful when dealing with high-dimensional and massive amounts of data. However, they have limited use in ontology matching, particularly in large-scale ontology matching.

In this research, we propose three different semantic solutions to deal with the large-scale ontology matching challenges without partitioning. (1) The first solution is NeuralOM, a supervised reuse approach based on artificial neural networks. It consists of combining the mappings of the top ranked matching systems by means of a single layer perceptron, to define a matching function that leads to generate a better alignment between ontologies. (2) The second solution is DeepOM, an ontology matching system that we propose to deal with the large-scale heterogeneity problem using deep learning techniques. It consists on creating semantic embeddings for concepts of input ontologies using a reference ontology, and use them to train an auto-encoder in order to learn more accurate and less dimensional representations for concepts on which similarities are computed. (3) The third solution is SemBigOM, the global methodology of this research that combines NeuralOM and DeepOM in order to perfectly and independently achieve the large-scale ontology matching process. It consists on exploiting DeepOM to generate initial mappings that NeuralOM requires as input to be reused so as to output the final matching results.

The experimental results of evaluating the proposed solutions on different test cases from the Ontology Alignment Evaluation Initiative, and comparing them with all participant systems of these tracks are very encouraging. They demonstrate the high efficiency of the

proposed work to increase the performance of the ontology matching task, and to tackle the large-scale ontology matching issue.

Keywords

Large-Scale Ontology Matching, Ontology Alignment, Semantic Web, Artificial Neural Networks, Deep Learning, Auto-Encoder, Semantics, Embeddings, OAEI.

Résumé

L’appariement d'ontologies est une méthode efficace pour établir l'interopérabilité entre les ontologies hétérogènes. L'alignement d'ontologies à grande échelle reste toujours un grand défi pour sa longue durée et sa grande consommation d'espace mémoire. La solution actuelle à ce problème est le partitionnement d'ontologies, qui est également un défi et présente aussi des difficultés.

Les réseaux de neurones artificiels sont de puissants modèles de calcul biologiquement inspirés du cerveau humain et de la manière dont il apprend et traite les informations. L'apprentissage profond est une voie de recherche prometteuse et une étape importante vers l'intelligence artificielle, imitant les mécanismes du cerveau humain, en particulier pour les problèmes extrêmement complexes. Les techniques d'apprentissage profond sont particulièrement efficaces lorsqu'il s'agit de traiter des quantités de données massives. Cependant, ils ont une utilisation limitée dans l'appariement d'ontologies, en particulier dans l'appariement d'ontologies à grande échelle.

Dans cette recherche, nous proposons trois solutions sémantiques différentes pour faire face aux défis d'appariement d'ontologies à grande échelle sans partitionnement. (1) La première solution est NeuralOM, une approche de réutilisation supervisée basée sur des réseaux de neurones artificiels. Elle consiste à combiner les résultats des systèmes d'appariement les mieux classés au moyen d'un perceptron à une seule couche, pour définir une fonction d'appariement qui conduit à générer un meilleur alignement entre les ontologies. (2) La deuxième solution est DeepOM, un système d'alignement d'ontologies que nous proposons pour traiter le problème d'hétérogénéité à grande échelle en utilisant des techniques d'apprentissage profond. Il consiste à créer des représentations sémantiques pour les concepts d'ontologies d'entrée à l'aide d'une ontologie de référence, et à les utiliser pour entraîner un auto-encodeur afin d'apprendre des représentations plus précises et moins dimensionnelles pour les concepts sur lesquels les similarités sont calculées. (3) La troisième solution est SemBigOM, la méthodologie globale de cette recherche qui combine NeuralOM et DeepOM afin de réaliser parfaitement et indépendamment le processus d'appariement d'ontologies à grande échelle. Il consiste à exploiter DeepOM pour générer des alignements initiaux dont NeuralOM a besoin en entrée pour être réutilisés afin de produire les résultats finaux d’appariement.

Les résultats expérimentaux de l'évaluation des solutions proposées sur différents cas de test de l'initiative d'évaluation de l'appariement d'ontologies (OAEI), et de leur comparaison avec tous les systèmes participants de ces défis d'appariement sont très encourageants. Ils démontrent la grande efficacité du travail proposé pour augmenter les performances de la tâche d'appariement d'ontologies et pour résoudre le problème d'appariement d'ontologies à grande échelle.

Mots-clés

Alignement d'ontologies à grande échelle, web sémantique, réseaux de neurones artificiels, apprentissage profond, Auto-Encoder, OAEI.

ملخص

مطابقة الأنطولوجيا هي طريقة فعالة لتأسيس قابلية التشغيل البيني بين الأنطولوجيا غير المتجانسة .لا تزال مطابقة الأنطولوجيا على نطاق واسع تمثل تحديًا كبيرًا نظرا لطول الوقت واستهلاك مساحة كبيرة من الذاكرة .الحل الفعلي لهذه المشكلة هو تقسيم الأنطولوجيا الذي يمثل تحديًا أيضًا.

الشبكات العصبية الاصطناعية هي نماذج حسابية قوية مستوحاة بيولوجيًا من الدماغ البشري، والطريقة التي يتعلم بها المعلومات ويعالجها .التعلم العميق هو وسيلة واعدة للبحث وخطوة مهمة نحو الذكاء الاصطناعي خاصة بالنسبة للمشاكل المعقدة للغاية .إنه ناجح بشكل خاص عند التعامل مع كميات هائلة وعالية الأبعاد من البيانات .ومع ذلك، فإن استخدامه محدود في مطابقة الأنطولوجيا، لا سيما في مطابقة الأنطولوجيا على نطاق واسع.

في هذا البحث، نقترح ثلاثة حلول دلالية مختلفة للتعامل مع تحديات مطابقة الأنطولوجيا واسعة النطاق دون تقسيمها. (1) الحل الأول هو NeuralOM ، وهو نهج إعادة استخدام خاضع للإشراف يعتمد على الشبكات العصبية الاصطناعية .وهو ينص على الجمع بين نتائج أنظمة المطابقة الأعلى مرتبة عن طريق شبكة اصطناعية، لتحديد معادلة مطابقة تؤدي إلى إنشاء محاذاة أفضل بين الأنطولوجيات. (2) الحل الثاني هو DeepOM، وهو نظام مطابقة الأنطولوجيا الذي نقترحه للتعامل مع مشكلة عدم التجانس واسعة النطاق باستخدام تقنيات التعلم العميق .وهو ينص على إنشاء تمثيلات دلالية لمفاهيم الأنطولوجيا باستخدام أنطولوجيا مرجعية من أجل الحصول على تمثيلات أكثر دقة وأقل أبعادًا للمفاهيم التي تُحسب عليها أوجه التشابه. (3) الحل الثالث هو SemBigOM ، وهي المنهجية العامة لهذا البحث التي تجمع بين NeuralOM و DeepOM من أجل تحقيق عملية مطابقة الأنطولوجيا على نطاق واسع بشكل مثالي ومستقل .وهو ينص على استعمال DeepOM للحصول على نتائج أولية يتطلبها NeuralOM كمدخل لإعادة استخدامها لإخراج نتائج المطابقة النهائية.

النتائج التجريبية لتقييم الحلول المقترحة في حالات الاختبار المختلفة من OAEI ومقارنتها مع جميع الأنظمة المشارك في هذه المسارات مشجعة للغاية .إنها تثبت الكفاءة العالية للعمل المقترح لتحسين أداء مطابقة الأنطولوجيا، ومعالجة مسألة مطابقة الأنطولوجيا على نطاق واسع.

الكلمات المفتاحية

مطابقة علم الوجود على نطاق واسع، محاذاة علم الوجود، الويب الدلالي، الشبكات العصبية الاصطناعية، التعلم العميق، الدلالات، OAEI.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

OM	Ontology Matching
OAEI	Ontology Alignment Evaluation Initiative
NeuralOM	Neural Ontology Matching
DeepOM	Deep Ontology Matching
SemBigOM	Semantic Big Ontology Matching
ANN	Artificial Neural Networks
DL	Deep Learning
AE	Auto Encoder
DIK	Data Information Knowledge
KIF	Knowledge Interchange Format
ISI	Information Science Institute
OCML	Operational Conceptual Modeling Language
F-Logic	Frame-Logic
SHOE	Simple Html Ontology Extensions
XML	EXtended Markup Language
SGML	Standard General Markup Language
W3C	World Wide Web Consortium
XOL	Xml Ontology Language
RDF	Resource Description Framework
DAML	DARPA Agent Markup Language
OIL	Ontology Inference Layer
OWL	Web Ontology Language
P2P	Peer-to-Peer
AGRO	AGRonomy Ontology
DRON	Drug Ontology
ECTO	Environmental conditions, treatments and exposure ontology
CIDO	Coronavirus Infectious Disease Ontology
CLO	Cell Line Ontology
ONS	Ontology for Nutritional Studies
OMIT	Ontology for MicroRNA Target
MOM	Modularisation-based Ontology Matching

OPM	Ontology Parsing graph-based Mapping
SGD	Stochastic gradient descent
MSE	Mean Squared Error
NLTK	Natural Language ToolKit
AEO	Anatomical Entity Ontology
CARO	Common Anatomy Reference Ontology
MeSH	Medical Subject Headings
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
HP	Human Phenotype
MP	Mammalian Phenotype
DOID	Human Disease Ontology
ORDO	Orphanet and Rare Diseases Ontology
FMA	Foundational Model of Anatomy
NCI	National Cancer Institute Thesaurus
SNOMED-CT	Systematized Nomenclature Of MEDicine-Clinical Terms
FLOPO	Flora Phenotype Ontology
PTO	Plant Trait Ontology
ENVO	Environment Ontology
SWEET	Semantic Web for Earth and Environment Technology Ontology
PM	Process Model
UA	University Admission
BR	Birth Registration
OA4QA	Ontology Alignment for Query Answering

List of Algorithms

Introduction

Research Context

In computer science, knowledge engineering is a field dedicated to collect information about the world, model this information and represent it in a way that a computer system can use to solve complex problems. Knowledge representation is one of the three main aspects of knowledge engineering. It is a field dedicated to represent and organize human knowledge in order to be used and shared, between humans, between computer systems, as well as between humans and computer systems.

Ontologies are representation methods. They allow representing a given domain so that its knowledge can be used and unified for all applications developed in different ways. Ontologies, first appeared in the 90s in many research axes, are the cornerstone of the semantic web. They helped to solve several problems, and improve the knowledge engineering process. An ontology [1] is a specification of a conceptualization, that is, a description of the concepts and relationships that may exist for a particular domain. An ontology is an explicit description of the concepts, properties, relationships and individuals that may exist for a particular domain. It reflects knowledge from a certain domain of discourse [2].

However, most applications require access to multiple ontologies and need to use information from different data sources. They often use multiple ontologies from different fields, and sometimes for the same field. Also, ontology construction is a very complex and critical task, because the main goal is to represent the real world. So, it is reasonable to think that two persons can have different points of view about the world, and how to represent it. Thus, due to the rapid development of the semantic web, the construction of ontologies by various experts leads to heterogeneity at different levels: terminological level, syntactic level, semantic level and semiotic level. Moreover, the necessary information for web users is often located in independent heterogeneous and distributed data sources [3].

Therefore, it is very interesting to identify correspondences between semantically related entities of heterogeneous ontologies. That allows agents using different ontologies to inter-

operate. These correspondences, called alignment or mapping, are the backbone of the ontology matching task which is the promising solution to this ontology semantic heterogeneity problem. It is generally based on computing similarity between the heterogeneous ontologies to be matched, in order to find semantic equivalences between them. Ontology matching can be applied to numerous fields, such as information integration, web service composition, ontology engineering, autonomous communication systems, peer-to-peer information sharing, navigation and query answering on the web [4].

Problem Statement and Challenges

Nowadays, ontologies of most applications are of big size, like in medicine and astronomy. And, large ontologies include a high conceptual heterogeneity. That could decrease the efficiency of ontology matching systems facing other challenges as shortage of memory and long-time processing. Such issue makes scaling up the ontology matching process a very interesting problem.

Although a wide number of systems have been developed in order to address ontology matching issue at the small-scale [3]; [4]; [5], large-scale ontology matching still presents several challenges. Much more work is required at this scale. Current ontology matching systems have to deal with big ontologies containing thousands to millions of entities each. Thus, they suffer from some difficulties related to memory consumption and processing time at the large-scale scene.

Partitioning large ontologies is the wide commonly used solution to deal with identifying semantic correspondences between different ontologies at the large scene [5] [6] [7] [8] [9]. It consists on dividing the input ontologies into several sub-ontologies. The overall result of matching is obtained after combining the individual results of matching sub-ontologies. However, partitioning ontologies also suffers from interesting challenges. It may decrease the matching quality, owing to the fact that, several semantic links inside ontologies are expected to be lost in the matching process, while they actually exist. Also, the partitioning parameters (number of partitions, size of each partition, number of elements per partition, how to divide ontologies, how to align these divisions, …etc) are also interesting and affect the matching performance. Moreover, ontology partitioning also suffers from the high time and space complexity while creating partitions. This has a direct impact on the efficiency of the ontology matching process.

Parallel large-scale ontology matching emerges as a complementary solution to the gaps of partitioning-based techniques. It is an effective solution at the large scale, but still has demands for ontology partitioning methods.

Reusing the previous matching results for large-scale ontology matching is another solution which is not very used in ontology matching. This is due to the fact that, they undergo some difficulties, even if they provide both high quality and low complexity of matching.

Artificial neural networks, often called neural networks, are one of the main tools used in machine learning. They are biologically inspired from human brain in a way to replicate the sort how human brains learn. In the last decades, they have become a major part of artificial intelligence, and have been used for various tasks such as image processing, speech recognition, natural language processing, and many others. This is due to their excellent ability to solve non-linear problems by learning, which is such a complicated and difficult task.

Deep learning techniques have been recently used to address important problems in many research axes, such as image processing, natural language processing, information retrieval, signal processing and many others. These techniques are very appropriate for dealing with large datasets. They have the ability to analyse and interpret massive amounts of data, that require efficient and effective computational tools. However, they have limited use in ontology matching. Moreover, the few approaches that employ these computational models aim at enhancing the performance of the ontology matching task, and not at handling the large-scale heterogeneity problem [10] [11] [12] [13] [14]. Besides, they tested their methods on ontologies of small sizes.

Objectives and Contributions

This research aims at addressing the large-scale ontology matching challenges, and developing a solution that exploits semantics inside ontologies and which is adapted to the requirements of big ontologies.

- The main objective of this study is to achieve both ***high quality*** and ***low complexity*** in large-scale ontology matching, and keeping these factors at the small scale as well. In other words, we aim at achieving high quality in small-scale matching where the complexity is obviously low, then passing these two factors to the large scale.

- The particular objective of this work is to overcome the limits of the existing solutions for large-scale ontology matching, and find a solution for this issue ***without partitioning***.

We attempted to achieve these research objectives by proposing three different solutions for the large-scale ontology matching issue.

Solution 1- Reuse-based semantic approach for large-scale ontology matching

We propose ***NeuralOM***, a new automatic ontology matching approach based on artificial neural networks, with focus on large-scale matching. It consists of reusing the alignments of the most effective ontology matching systems basing on artificial neural networks in order to define a matching function which leads to generate the ideal alignment between input ontologies. The aim behind training the involved network is to adjust a weight for each matcher according to its efficiency. This refining strategy using artificial neural networks serves to have optimal matching results and to increase the performance of the ontology matching task.

The main contribution of the approach proposed in this work is that, it combines, according to a very detailed state-of-the-art on the existing ontology matching techniques, the produced mappings of the most efficient matching systems aiming to refine their results that have been validated through various test cases. We aim by refining these results to achieve a perfect ontology matching. NeuralOM considers the input mappings generated by the selected candidate matchers as initial alignment that takes the matching process as parameter. Reusing and refining ontology matching results denotes working on a higher and more precise level. Moreover, unlike the actual ontology matching techniques that combine several similarity measures, NeuralOM combines several mappings which that have been generated through complicated processes and validated by various tests. This matching reuse is performed using artificial neural networks which are very appropriate for such combination tasks. Furthermore, as the selected matching tools work differently, NeuralOM acquires a great chance to provide a maximal number of correct correspondences for the reason that it benefits from different matching strategies.

We evaluate this matching approach through a very detailed experimental study according to twelve different test cases from different campaigns of the Ontology Alignment Evaluation Initiative (OAEI). The results of the performed experiments show that NeuralOM has proven its efficiency in front of all OAEI matching systems in terms of the different

measures adopted for evaluation. It has achieved very excellent scores for all matching tasks and with negligible matching time even for dealing with large ontologies. That shows a very high accuracy of ontology matching especially at the large-scale.

Solution 2- Deep embedding learning with auto-encoder for large-scale ontology matching

We propose a new ontology matching system, called ***DeepOM***, which employs deep learning techniques in order to efficiently match huge ontologies without partitioning them, and at the lowest time process and memory space cost.

The idea behind DeepOM is to automatically treat the large-scale ontology matching issue in two main stages. At each stage, it aims to provide more representative and less dimensional real-valued vectors for concepts of input ontologies. DeepOM first transform the ontological concepts of input ontologies into numerical vectors that deep learning models can use as input. This embedding process is based on the semantic similarity between them and the concepts of a smaller and well selected reference ontology. Second, DeepOM trains an auto-encoder on the generated vectors, so as to learn high-level and more compact representations for concepts of input ontologies. Similarities of the generated correspondences is then computed using the cosine similarity between the compressed representations of concepts. At each stage, DeepOM improves the performance and reduces the complexity of large-scale ontology matching.

The proposed system provides several contributions. The core contribution is that it employs deep learning techniques which are very appropriate for dealing with huge amounts of data to represent the concepts of input ontologies in a multi-dimensional embedding space. The aim behind this procedure is to transform the concepts into richer and more precise representations which serves to reduce the matching complexity, and that deep learning models can use. Moreover, the embedding process is based on using a smaller and well selected reference ontology which has a great impact on the matching results. Furthermore, an auto-encoder is trained on the produced embeddings, in order to learn more accurate and more compact representations for input concepts. Exploiting such models which are great at representation learning leads to better performance and less complexity as well. This is due to the fact that this dimensionality reduction serves for keeping the most important attributes of the input vectors in the lower dimensional compressed representations.

The results of evaluating DeepOM on large OAEI ontologies, and its comparison with ontology matching systems which have participated to the same test case, demonstrate its high ability to tackle the large-scale ontology matching problems. All the matching factors of DeepOM are positive towards perfecting the matching performance and reducing its complexity.

Solution 3- Matching large-scale ontologies semantically by combining NeuralOM and DeepOM

The two solutions previously described, NeuralOM and DeepOM address the challenges and respond to the main as well as the particular objectives of this study. NeuralOM is characterized by its excellent matching results, but it is related to initial mappings of other matching systems. DeepOM is totally independent, but its results are poorer than the results of NeuralOM.

As we aim for an ideal ontology matching, we propose ***SemBigOM***, the global methodology of this research that combines the two proposed solutions in order to achieve excellent matching results independently. SemBigOM seeks for tackling the challenges of large-scale ontology matching and overcoming the limits of NeuralOM and DeepOM. The basic idea of SemBigOM is that, it makes use of DeepOM to output initial mappings that NeuralOM requires as input to be reused so as to generate the final matching results. In other words, SemBigOM exploits NeuralOM to refine the matching results of DeepOM so as to perfectly and independently achieve the large-scale ontology matching process. Figure 1 illustrates an overview of the research methodology.

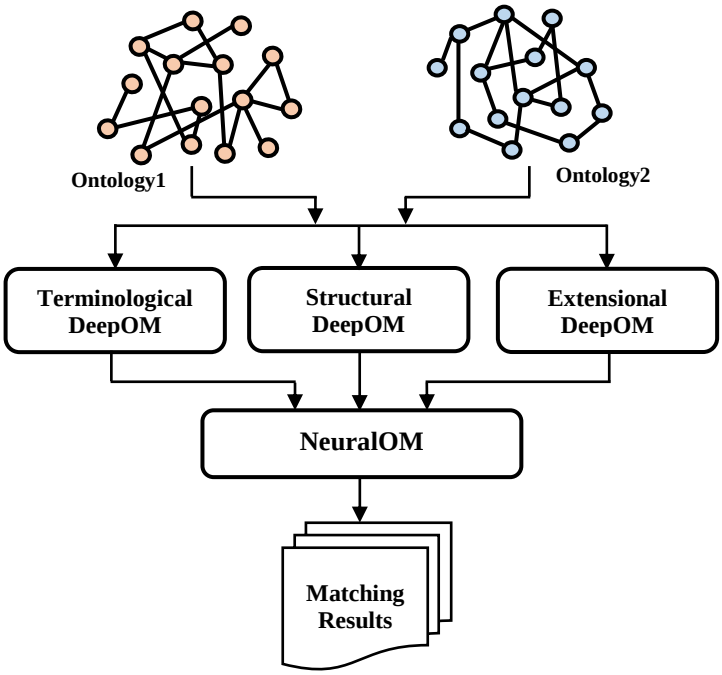

Figure 1. Research Methodology Overview.

As we aim for exploiting semantics inside ontologies, SemBigOM first generates three different mappings between input ontologies basing on the different aspects of the ontological concept, and using three different versions of DeepOM: Terminological-based DeepOM, Structural-based DeepOM and Extensional-based DeepOM. Each matcher represents concepts of ontologies in a numerical multi-dimensional vector space using a reference ontology. An auto-encoder is then trained in order to transform the produced numerical vectors into finer and smaller representations for concepts. Similarities of the initial mappings are computed using the cosine similarity. After that, the three generated mappings are reused by NeuralOM which defines the matching function that leads to generate, after a filtering procedure, an ideal mapping between input ontologies.

The proposed methodology provides several contributions. The basic contribution is that, it is advantages of providing a correct alignment, since it exploits semantics of input ontologies and benefits from covering all aspects of their concepts. Also, it exploits background knowledge resources for measuring the required similarities. In addition, SemBigOM treats the large-scale ontology matching problem at two main stages aiming for perfecting and simplifying the matching process at each stage. First, SemBigOM involves deep learning methods, in order to create semantic embeddings for concepts of input ontologies, and compress them to a lower-dimensional vector space using an auto-encoder. That provides better matching performance and decreases the matching complexity. Second, SemBigOM

performs a refining procedure on its initial mappings in order to generate an ideal mapping. Likewise, that allows perfecting the matching process with negligible matching complexity. Moreover, SemBigOM adapts for parallelization any multiple tasks that can run in parallel, aiming for reducing the complexity of large-scale ontology matching.

The results of evaluating SemBigOM on large OAEI ontologies against the different ontology matching systems participants to the same case as well as NeuralOM and DeepOM, demonstrate a high accuracy of matching. It has proven its efficiency to match large-scale ontologies, to address the large-scale ontology matching problems, and to achieve all objectives which have motivated this work.

Manuscript Outline

This manuscript is structured in six chapters divided into two main parts. The first part is composed of three chapters of the state of the art. The second part comprises three chapters which presents our contributions. Figure 2 shows the organization of the remaining chapters of this manuscript.

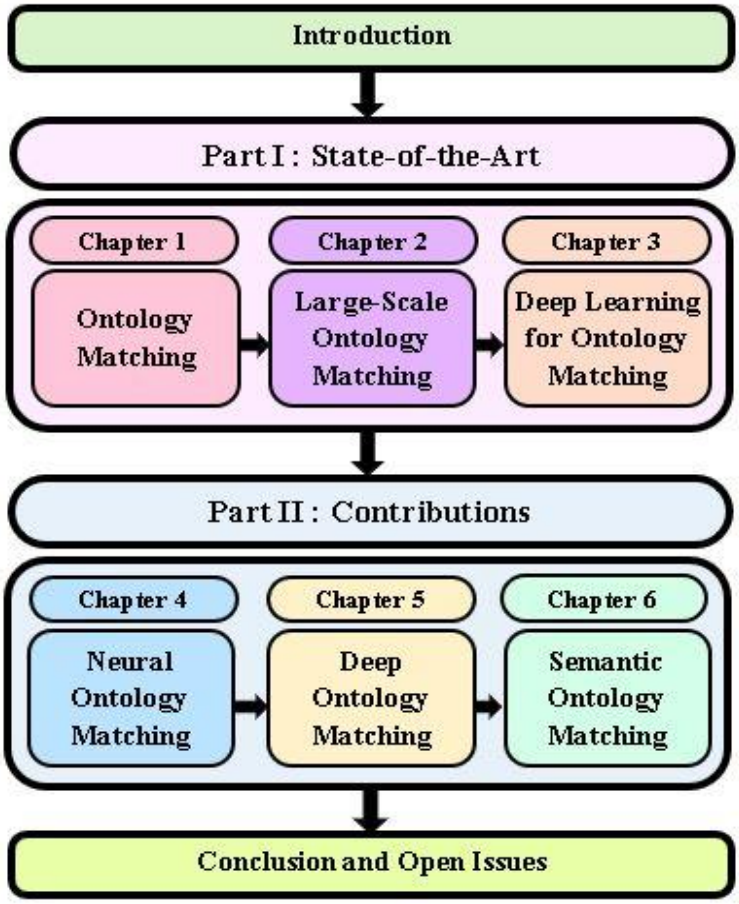

Figure 2. Organization of the manuscript.

Part I. State-of-the-Art

- Chapter 1 marks the beginning of the theoretical background of this thesis. It is divided in two main parts. In the first part we introduce the central component of this work which is "*ontology*", where we present the knowledge engineering field, knowledge representation models, the background of ontologies, their components and their expressing languages. The second part is dedicated to ontology matching, where we present the need behind ontology matching, ontology matching evaluation and its different applications. We also review and classify the different ontology matching techniques with a part of prominent ontology matching systems.
- Chapter 2 reviews a state-of-the-art on large-scale ontology matching. It includes three main sections: the first one focuses on studying the motivations behind this issue and the need for matching large ontologies; the second section describes the different possible strategies to deal with large-scale matching, and proposes a classification of these techniques; the third section reviews the existing ontology matching systems which have been developed to handle the large-scale matching problems, and provides an analytical summary of these tools.
- Chapter 3 reviews the existing work related to employing deep learning techniques for ontology matching. It is organized into two major parts. In the first part, we describe the basic concepts of deep learning, artificial neural networks, their architectures, auto-encoders, embedding models, activation functions and learning methods. The second part is dedicated to the related literature on the existing ontology matching tools based on artificial neural networks, and particularly on deep neural networks. We also provide an analytical summary and discuss the presented techniques.

Part II. Contributions

- Chapter 4 presents NeuralOM, the first solution that we propose to address the large-scale ontology matching challenges basing on artificial neural networks. After describing the proposed matching method, we present the evaluation framework where we conduct our experiments on twelve different test cases from the open Ontology Alignment Evaluation Initiative (OAEI) in both small and large scales. We discuss the results of these experiments and study the performance of our approach.

- Chapter 5 presents DeepOM, another different solution that we propose to deal with the large-scale heterogeneity problem using deep learning techniques. After introducing the proposed ontology matching system and its detailed workflow, we describe the evaluation of our system, conducted on the Anatomy track from the 2020 campaign of the OAEI Initiative. We also analysis the results of these experiments and discuss the performance of DeepOM.
- Chapter 6 presents SemBigOM, a solution that we propose for addressing the current challenges of large-scale ontology matching, as well as the challenges of NeuralOM and DeepOM. It combines the previous two proposed solutions and represents the global methodology of this research. After reviewing the methodology and presenting its contributions, we describe its workflow. We also present the strategy followed and the adaptations made for evaluating this research. And, we present and analysis the results of evaluating SemBigOM according to the Anatomy track from the most recent campaign of the OAEI Initiative, and we discuss its efficiency.

This manuscript ends with a conclusion which summarizes the thesis with a discussion about the different results, and concludes with some perspectives for future research directions.

Published Work

The following published papers are partial outputs of this thesis.

Ali Khoudja, M., M. Fareh, and H. Bouarfa, *Deep Embedding Learning With Auto-Encoder for Large-Scale Ontology Matching.* International Journal on Semantic Web and Information Systems (IJSWIS), 2022. **18**(1): p. 1-18.

Ali Khoudja, M., M. Fareh, and H. Bouarfa. *A new supervised learning based ontology matching approach using neural networks*. in *International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*. 2018. Springer.

Ali Khoudja, M., M. Fareh, and H. Bouarfa. *Ontology matching using neural networks: Evaluation for OAEI tracks*. in *International Symposium on Modelling and Implementation of Complex Systems*. 2020. Springer.

Ali Khoudja, M., M. Fareh, and H. Bouarfa. *Ontology matching using neural networks: survey and analysis*. in *2018 International Conference on Applied Smart Systems (ICASS)*. 2018. IEEE.

Chapter 1

Ontology Matching

Contents

1.1. Introduction

Ontologies are the cornerstone of the semantic web. They are highly heterogeneous. Ontology matching is the most adapted solution to ontology heterogeneity problems. Therefore, it has become a crucial task in semantic web applications. Several ontology matching systems have been developed in the scientific literature in order to define semantic correspondences between entities of heterogeneous ontologies.

Since our work lay in the context of ontology matching, this chapter marks the beginning of the theoretical background of this thesis. It is divided in two main parts; in the first part we introduce the central component of this work which is "*ontology*", and the second one is dedicated to ontology matching. We first present the background of ontologies; the knowledge engineering field, and we outline the different models of knowledge representation. Then, we review ontology engineering, starting with what ontologies are and their origins rooted in philosophy, defining their components and describing the various languages in which ontologies can be expressed. After that, we present the need behind ontology matching and heterogeneity problems. The matching process and some used terminologies are also defined. Then, we describe and classify the different ontology matching techniques, and give a brief overview of some prominent ontology matching systems. We also discuss ontology matching evaluation; evaluation principles, types of evaluation and evaluation measures. Finally, we present several applications which can take advantage of ontology matching.

1.2. Ontology Background

1.2.1. Knowledge Engineering

Knowledge engineering has evolved from the late 1970s onward, and consists of constructing different aspect models of human knowledge [15]. In computer science, knowledge engineering focuses on the identification, creation, storage, provision and representation of knowledge in order to be used and shared. Like any engineering, knowledge engineering goes through several stages:

- **Collection** of information about the world;
- **Modelling** this information;
- **Representation** of knowledge in a way that a computer system can use to solve complex problems.

This study fits into the third stage which we explain by the following. Before presenting the main concepts related to knowledge representation, it is necessary to define the meaning of the word “knowledge”. However, this meaning is difficult to define because knowledge very much depends on context. We thus clarify this notion in confront to the terms “data” and “information”.

1.2.2. Data, Information and Knowledge

In computer science, *data*, *information* and *knowledge* belong closely together and have slightly different meanings. They are distinguished by the level of interpretation associated with them. We briefly define them as follows:

Data (the lowest point of the pyramid)**.** Data are unstructured and uninterpreted facts, figures and signals that reach our senses by the zillions every minute, but have the least impact on our decisions.

Information (the next level)**.** Information is data equipped with meaning. It is considered as structured data in a given context;

Knowledge (the highest level)**.** Knowledge is closely linked to practice. It is the collection of data and information that we bring to practical use in order to carry out actions and create new information.

Data, information, and knowledge can be represented by a pyramid. Figure 1.1 presents the DIK pyramid with an illustrating example. As shown by this example:

- “Red” is a *data*;
- “The color of the traffic light is red” is an *information*;
- For a human car driver, a red traffic light is not just a signal of some colored object, but it is interpreted as an indication to stop, this is a *knowledge*.

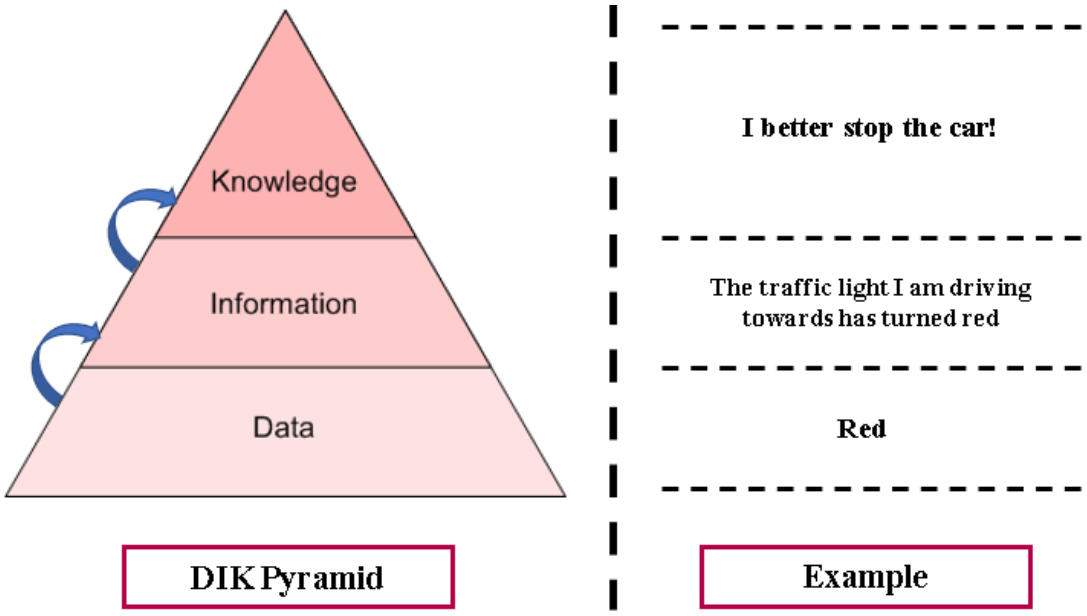

Figure 1.1. Data, Information and Knowledge Hierarchy.

1.2.3. Knowledge Representation Models

The vast data irreversible movement that is leading us towards a world of knowledge highlights the importance of representing this knowledge in a structured, visual and transmissible way. Knowledge representation is defined by the set of tools of which the objective is to identify, structure, and organize human knowledge in a schematic representation in order to make it usable and shareable between machines and between humans and machines. In the following, we present the main models of knowledge representation. Their definitions are adapted from [16].

1. Tags and Folksonomies

Tags and folksonomies are used in popular web sites. They are used as very simple ways to describe the content by tagging it, i.e., describe a corpus of knowledge by just giving names, called tags, to them.

2. Directories

A *taxonomy* is a partially ordered set of taxons (classes) where one taxon is greater than another one only if what is denoted by the former includes what the latter denotes. A *directory* or a classification is a taxonomy which is used by companies to present goods on sale, by libraries to store books, or by individuals to sort files on a personal computer.

3. Relational Database Schemas

Relational databases organize data in a predefined way in terms of tables and relations. A relational schema specifies the names and the types of columns of each table. The relational model includes the notion of a key for each table. i.e., the subset of columns which uniquely identifies each row. And, a column in a table may point to a column in another table via a foreign key. This is intended to support referential constraints among various entities.

4. XML Schemas

XML schemas have been introduced for the purpose of specifying the structure of XML documents. Their main components are elements, attributes, and types. XML schemas are complementary to directories.

5. Conceptual Models

Conceptual models allow specifying entities in a domain with a high level of expressivity. They provide constructors for both organising classes in a hierarchy and describing the internal structure of objects. Thus, they offer the best of both directories and databases.

6. Ontologies

Ontologies, first appeared in 90s, are the most used models for representing knowledge. The representation models presented previously can be considered as variations of ontologies with various differences and degrees of formality and expressivity. The primary objective of an ontology is to represent a given domain, so that its knowledge can be used and unified for all its applications developed in different and independent ways. For this purpose, ontologies are used in artificial intelligence, semantic web, software engineering, biomedical informatics, information retrieval and many other domains. In the next section, we provide a more detailed explanation of ontologies.

1.2.4. Ontology Engineering

1.2.4.1. What is an Ontology?

In philosophy, ontology denotes the theory of *"the nature of being or the kinds of existents"* [17]. It is a term that appeared in Metaphysics with the Greek philosophers Socrates and Aristotle, who were the first creators of ontology foundations. Socrates introduced the notion of a hierarchy of abstract ideas and class-instance relations. Later, Aristotle formed the logical basis of ontologies by adding logical associations. The resulted model is well-structured and capable to describe and represent knowledge about the real world. In this context, an ontology is the philosophical study of the nature of being and existence, i.e., the

study of the general properties of what exists, by defining all the knowledge about the world. The first papers introducing the philosophical ontology area were published around 1960 [18].

Afterward, the term *"Ontology"* has become widely used in the field of Computer Science. It appeared for the first time, at the beginning of the 90s, in the context of research on artificial intelligence in order to solve the problems of knowledge engineering, more precisely, in knowledge representation. Thus, computer scientists have adopted the term *"Ontology"* for their own needs.

In order to clarify this notion, many researchers have proposed definitions of what an ontology is. In this section, we cite several definitions about ontology, and observe how these definitions have evolved over the years, and the relationships between them.

- One of the first definitions was given by Neches et al. [19]: "*An ontology defines the basic terms and relations comprising the vocabulary of a domain, as well as the rules for combining these terms and relations in order to define extensions of the vocabulary".* According to this definition, an ontology includes the terms which are defined in an explicit manner, and knowledge that can be conveyed by terms.
- The most prominent and cited definition is that given by Thomas R. Gruber where he describes an ontology as an explicit specification of a conceptualization modelling concepts and relationships between concepts. He defined ontology as: "*An ontology is a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general"* [1].
- Based on the Gruber definition, Borst [20] has proposed in 1997 to define ontology as: "*an ontology is a formal specification of a shared conceptualization*". This definition specifies the fact that the ontology must be formal and shared.
- In 1998, the last two definitions (those of Gruber and Borst) were combined by Studer et al. [21] to have the following definition: "*An ontology is a formal, explicit specification of a shared conceptualization".* They explain it as follows:
    - ***Conceptualization*** refers to an abstract model of some phenomenon in the world by having identified the relevant and more appropriate concepts of that phenomenon.

- ***Explicit*** means that the type of used concepts, and the constraints on their use are explicitly defined.
- ***Formal*** refers to the fact that the ontology should be expressed into a machine-readable language.
- ***Shared*** reflects the notion that an ontology captures consensual knowledge, i.e., it is not private to some individual. Shared does not necessarily mean shared by all individuals, but accepted by a group.

- In [22], authors made semantic interpretations to introduce the notion of formal ontology, and proposed the following definition: "*An ontology is an agreement on a shared and possibly partial conceptualization*".
- There is another type of definition based on the process of constructing ontologies, such as the definition proposed by [23]: "*an ontology provides the means to describe in an explicit way the conceptualization of knowledge represented in knowledge bases*". This definition proposes the extraction of ontology from a knowledge base.
- Another definition of the same type was given in [24] by Swartout et al.: "*an ontology is a set of hierarchically structured terms, designed to describe a domain that can be used as a basic skeleton for knowledge bases*".
- Later, John F. Sowa specified this notion more precisely. He defined ontology as a catalog of types, resulting from the study of categories of abstract and concrete entities that exist or can exist in a domain: "*The subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study, called an ontology, is a catalogue of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D. The types in the ontology represent the predicates, word senses, or concept and relation types of the language L when used to discuss topics in the domain D*" [25].
- A formal definition of ontology is given in [26]. According to Udrea et al., ontologies model data structure (i.e., sets of classes and properties), data semantics (in the form of axioms such as inheritance relations or property constraints), and data instances (individuals). Thus, the entities of an ontology are composed of a "**structure**" part, and a "**data**" part.

- Also, according to Cheatham and Pesquita [27], the information of classes, properties, and axioms that restrict their interpretation, are called the "structure", "scheme", or "**T-box**" (as Terminology) of the ontology, and information of instances and their axioms are called "data", "instances data" or "**A-box**" (like Assertions) and contain assertions about instances using T-box data.
- Finally, Zhang et al. [28], have formally defined an ontology as a tree model, because of the principle of hyponymy (*is-a* subsumption) which means that each entity is inherited from a single direct super-entity, thus forming a rooted acyclic graph structure. But in the case of multiple inheritance, the ontology becomes a network model that can contain cycles and in which several paths can lead to an entity.

1.2.4.2. Ontology Components

As discussed in the section of ontology definitions, ontologies formally represent the vocabulary of a given domain and provide a specification of its meaning. According to Gruber [1], knowledge in ontologies is mainly formalized using the five types of components: concepts, relations, functions, axioms and instances. In this section, we introduce these main components that build up an ontology.

1. Concepts

Ontology is a specification of a conceptualization. Concepts, also known as Classes, are the basic notion in an ontology. They are used in a wide sense. They can represent abstract concepts (intentions, feelings, beliefs, …etc.) or concrete concepts (people, tables, computers, …etc.). In short, a concept can be an abstract notion about anything; the description of a task, action, strategy, reasoning process, …etc. More formally, a concept is an abstraction that brings together a number of real-world objects that are its instances.

A concept is defined by a semantic triangle (see Figure 5.3), i.e., it is defined by 3 dimensions: *Term*, *Intention* and *Extension*. These basic elements preserve the semantics carried by the concept. More details are provided in Chapter 5.

2. Relations

Relations represent interactions between concepts allowing the construction of complex representations of domain knowledge [29]. These relations are links that reflect the relevant associations existing between the domain concepts. Ontology relations are usually binary semantic links which have two arguments. The first argument is known as the domain of the relation, and the second argument is the range.

There are several types of relations:

- Relations used by the hierarchy ***is-a*** *(*also known as ***Subclass-Of)***. Ontology classes are generally organized in taxonomies which are widely used to organize the domain knowledge using this generalization/specialization relationship.
- Relations which are used to define the hierarchy according to ***part-of***, ***composed-of***, …etc.
- Semantic relations which are defined by four elements:
  - ***Term***: represents the relation in language;
  - ***Extension****:* is the set of the effective realizations of a relation between concepts;
  - ***Intension***: is the set of properties and attributes of the relation;
  - ***Signature***: is the set of concepts participating to this relation.

3. Functions

Ontology functions are complex structures made from some relations that can be used as an individual term in a statement. Functions are particular cases of ontology relations, in which the n-th element of the relation is unique for the n-1 preceding elements [30].

4. Axioms

Axioms are logical assertions and constraints that comprise the theory described by the ontology in its domain of application. They are ordinarily used to represent knowledge which cannot be formally defined by the other components of the ontology [30]. They are employed to better define and give more meaning to the other components (concepts, attributes, relations, …etc.). Thus, ontology axioms can serve to verify the ontology consistency, constrain the information represented by the ontology, deduce new information, …etc.

5. Instances

Instances, also known as individuals, are used to represent elements of the domain described by the ontology at the ground level. The instances in an ontology may include concrete objects (people, tables, molecules, animals, … etc.), as well as abstract individuals (numbers, words, … etc.).

1.2.4.3. Ontology Example

In order to clarify the notion of ontologies and their components described previously, we present, in this section, a very simple ontology example from the domain of organizing

conferences. This ontology example is graphically illustrated in Figure 1.2. It is visualized using the *Web-based Visualization of Ontologies* **(WebVOWL)**[1] tool. The example contains *article*, *document*, *person*, *conference*, and the other objects depicted as circular forms as concepts. *conference*, *session*, *pc_meeting* are sub-classes of the concept *event*. Another subsumption relation is between *chair* and *person*; a chair is a person. The concept *chair* is related to the concept *reviewer* by the relation *assigns_reviewers*. Each event has two attributes: *has_startdate* and *has_enddate* of type *dateTime*. An example of axioms is that, a *person* should have at least one connection with *review* by the relation *writes_review* to be a *reviewer*. Instances are not depicted in the graph, but abundant in the real world.

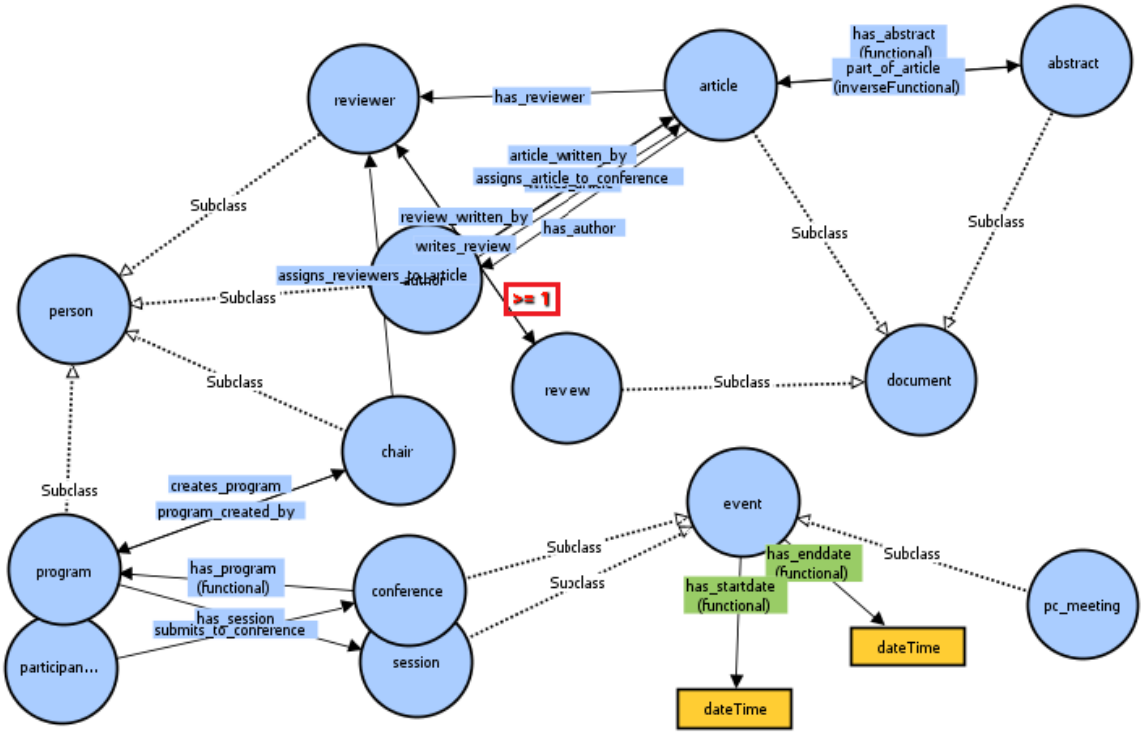

Figure 1.2. Ontology example from the domain of organizing conferences.

1.2.4.4. Ontology Languages

Ontologies are normally expressed in an ontology language. An ontology language is a formal language to describe the different elements that compose an ontology. There is a wide variety of languages for representing ontologies [31]. Most of them share equivalent kinds of entities with comparable interpretations. They vary in their levels of expressivity. We can distinguish between two main families of ontology languages:

[1] http://vowl.visualdataweb.org/webvowl.html

1.2.4.4.1. Traditional Ontology Languages

At the beginning of the 90s, a set of Artificial Intelligence based ontology languages was created. Their global layout is shown in Figure 1.3.

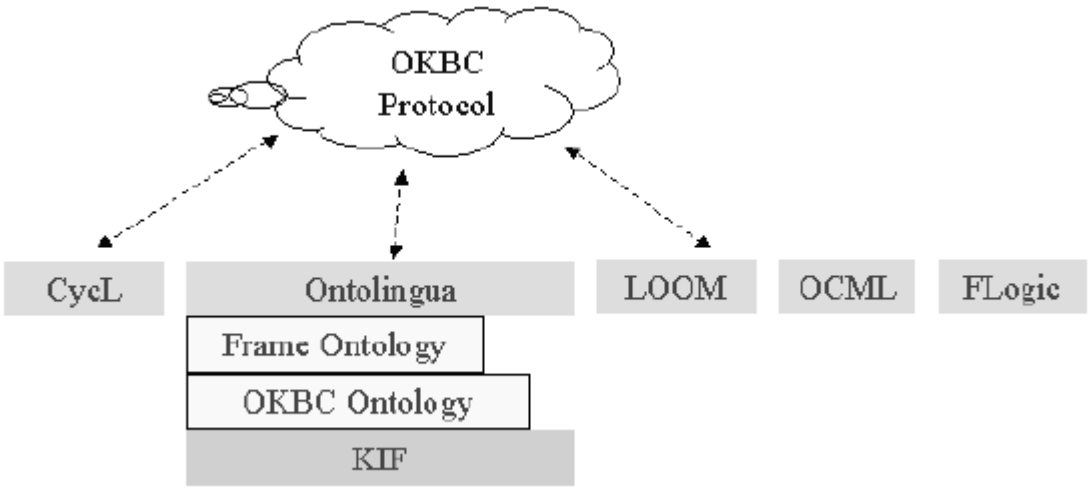

Figure 1.3. Traditional ontology languages [32].

We briefly outline these traditional languages:

- **Cycl**

  This is the first language that was created (developed in 1990 within the Cyc project [33]). It is a formal language based on *Frames* and *First-order logic*.

- **KIF** (Knowledge Interchange Format)

  KIF [34] was developed in 1992 to solve the problem of language heterogeneity in knowledge representation, and to allow the interchange of knowledge between diverse information systems. It is based on *First-order logic* with some extensions. It also permits the representation of meta-knowledge, reifying functions and relations, and non-monotonic reasoning rules.

- **Ontolingua**

  Ontolingua [35] was released by the Knowledge Systems Laboratory of Stanford University in 1992. It is based on KIF language and on the *Frame Ontology*. Moreover, it is the ontology-building language used by the Ontolingua Server [36].

- **LOOM**

  LOOM [37] is a high-level programming language, which is not designed for the development of ontologies but for knowledge bases, it is based on description logic and production rules, it provides an automatic classification of concepts. It was being

developed by the Information Science Institute (ISI) of Southern California University from 1986 to 1995.

- **OCML** (Operational Conceptual Modeling Language)

  OCML [38] was developed in 1998 at the Knowledge Media Institute (UK). It was designed as an extension of the Ontolingua language in order to fill its gaps by supporting *production rules*, which allows improve the reasoning mechanisms of Ontolingua.

- **F-Logic** (Frame-Logic)

  FLogic [39] combines both Frames and First-order logic. It was created in 1995 at the Department of Computer Science of the State University of New York. It was specially used for deductive and object-oriented databases, and was later adapted and used for implementing ontologies. It has an inference engine, *Ontobroker*, which can be used to derive new knowledge.

1.2.4.4.2. Ontology Markup Languages

The explosion of Internet technologies has led to the creation of languages for exploiting the features of the Web. These languages are generally called *Web-based languages* or *ontology annotation languages*. Their goal is to represent and exchange data over the web. They are represented in Figure 1.4 and described as follows:

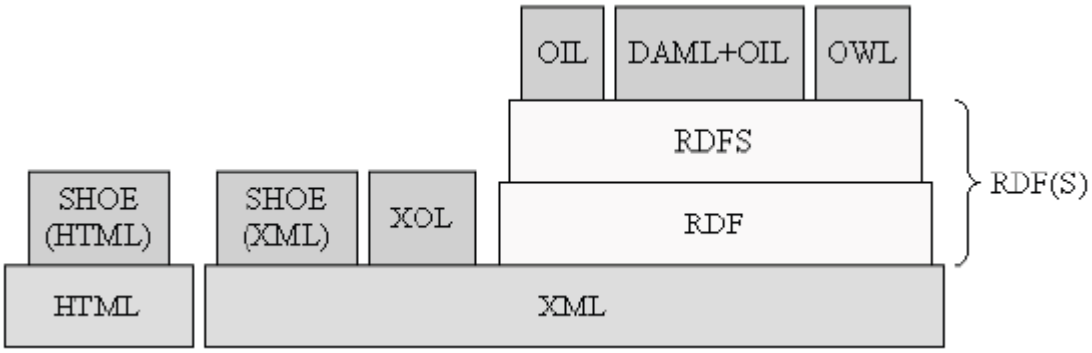

Figure 1.4. Ontology markup languages [32].

- **SHOE** (Simple Html Ontology Extensions)

  SHOE [40] is the first ontology annotation language. It was developed, in 1996 in the University of Maryland (USA), as an extension of HTML. It uses special tags that allow inserting semantic data into web pages. These tags are of two categories: tags for constructing ontologies and tags for annotating web documents. This language combines *Frames* and *Production rules*, so as to represent concepts, taxonomies, relationships, and also rules that allow to infer new knowledge.

- **XML** (EXtended Markup Language)

  XML [41] is a language for describing and exchanging structural documents, derived from the ISO standard SGML (Standard General Markup Language). In 1998, XML had been very quickly adopted as a standard for the exchange of information on the Web by the W3C[2] (World Wide Web Consortium), for ease of implementation and better interoperability between information systems with both SGML and HTML. It is used to exchange a wide variety of data on the Web, allowing users to define their own tags and attributes, define data structures, extract data from documents and develop applications [30]. Some languages were subsequently created based on the syntax of XML, whereas other existing languages were modified so that they could support structured documents described in XML.

- **XOL** (Xml Ontology Language)

  XOL [42] was developed in 1999 as an XML-based language that allows the specification of concepts, taxonomies and binary relationships. Thus, it is not used for developing ontologies, but as an intermediate language for exchanging ontologies and transferring them among several database systems, application programs or ontology-development tools.

- **RDF(S): RDF** (Resource Description Framework) and **RDF Schema**

  RDF [43] was developed by W3C as a language based on semantic networks to describe web resources. It is an infrastructure that enables the encoding, exchange and reuse of structured metadata, so that intelligent agents, browsers, search engines and human users can make use of semantic information.

  In order to reinforce this language, RDF Schema was built by W3C as an extension of RDF combining semantic networks with frames to provide primitives of knowledge representation systems.

  The combination of RDF and RDF Schema is known as RDF(S). RDF(S) is widely used as a representation format in many tools and projects. Many resources for handling RDF(S) exists, such as editing, browsing, querying, validating, storing, …etc. However, RDF(S) languages are not very expressive. concepts, instances and relations of an ontology can be easily defined with RDF(S), but it will lack from functions and axioms.

- **DAML** (DARPA Agent Markup Language) **+ OIL** (Ontology Inference Layer)

[2] https://www.w3.org/

The DAML Language is designed to allow expressing ontologies in an extension of RDF. It provides the usual primitives of frames-based representation using the RDF syntax.

The OIL language is based on RDF(S). It can express ontologies on the Web, by combining the modeling primitives used in frame languages and the formal reasoning of description logics.

The DAML language has been merged with the OIL language to form the DAML+OIL language [44], which inherits the advantages of both of them. As a result, DAML+OIL is an expressive, machine-readable and human-readable language with an RDF-based syntax.

- **OWL** (Web Ontology Language)

  The OWL language [45] has been created by the W3C Web Ontology (WebOnt) Working Group. It was built upon RDF(S) and derived from the DAML+OIL language. OWL aims for publishing and sharing ontologies in the Web. More specifically, aims to make web resources readily accessible to software applications and automated processes, as opposed to situations where the content is only presented to humans.

  OWL allows to explicitly describe ontologies, i.e., define terminologies and relationships between them to describe concrete domains. A terminology is basically made up of concepts and properties, while, a domain is basically made up of instances of concepts. For this, OWL is more powerful in expressing the meaning and semantics than XML, RDF and RDF(S). In addition, it allows linking ontologies and sharing information between different knowledge sources.

  The OWL language consists of three sub-languages designed for specific developer communities and users: OWL Lite, OWL DL and OWL Full. They provide increasing expressiveness; each sub-language is an expansion over its simpler predecessor.

  - **OWL-Lite:** supports users who mainly need a classification hierarchy and simple constraints (of cardinality 0 or 1 for example). This cardinality corresponds to functional relationships, for example a person has an address. However, this person may have one or more given names. OWL Lite is therefore not sufficient for this situation.
  - **OWL-DL:** is based on Description Logic theoretical properties, and supports users who require maximum computational completeness (all conclusions are guaranteed to be computable), and decidability (computations will finish in

finite time). It includes all constructs of the OWL language, but places certain constraints to use them.

- **OWL-Full:** is the complete OWL language. It provides the maximum of expressivity and more flexibility to represent ontologies than the precedent languages. It allows mixing OWL with RDF Schema and does not enforce a strict separation of ontology entities. For example, a class can be treated simultaneously as a collection of individuals and as an individual in its own right. However, using OWL Full features can lead to lose some guarantees provided by OWL-Lite and OWL-DL for reasoning systems [30].

The ontology languages presented previously differ in their abilities of expressivity. The ontologies that we use in this study are expressed in OWL language which is supported by numerous initiatives and tools, and provides the higher level of expressivity.

In the real world, ontologies describe particular domains of knowledge. As presented above, they are expressed in different ontology languages with different degrees of expressiveness. That can lead to various problems of heterogeneity. In the following section, we introduce the different forms of heterogeneity that may occur, and the most adapted solution to this problem which is ontology matching.

1.3. Ontology Matching

1.3.1. Why Ontology Matching

In the semantic web and its distributed applications where ontologies are used, heterogeneity poses a real problem. Different users have different habits and interests, use different tools and knowledge, and usually at different levels of detail. This leads to diverse forms of heterogeneity, which should be taken into consideration.

Ontology matching aims to reduce heterogeneity between ontologies. Heterogeneity lies in the differences between goals of the applications using ontologies, in the expressing languages in which ontologies have been encoded, how to use them, …etc. Therefore, we distinguish several types of heterogeneity that usually occur together [16]:

- **Terminological heterogeneity** happens due to variations in names when referring to the same entities in different ontologies. This is due to using different natural languages, for example, "*Paper"* vs. "*Articulo"*, using synonyms, e.g., "*Paper"* vs. "*Article"* or using different technical sublanguages, e.g., "*Paper"* vs. "*Memo"*.

- **Syntactic heterogeneity** occurs when two ontologies are not expressed in the same knowledge representation language. This happens when comparing two ontologies of different expressive formalisms, for example, *F-logic* and *OWL*. This kind of mismatch is generally tackled at the theoretical level, establishing equivalences between constructs of different languages [16].
- **Conceptual heterogeneity** (also called **semantic heterogeneity**) stands for the differences in modelling the same domain of interest. This may happen due to using different axioms for defining concepts, or using totally different concepts. Three important reasons of conceptual differences are identified by [46] basing on the represented domain, the level of detail and the intended perspective. We discuss them below with examples about geographic map:
    - *Difference in coverage* occurs when two ontologies represent different domains at the same level of detail and from a unique perspective. For instance, two partially overlapping geographic maps.
    - *Difference in granularity* occurs when two ontologies describe the same domain from the same perspective but at different levels of detail. For instance, geographic maps with different scales: one displays buildings, while another depicts whole cities as points.
    - *Difference in perspective*, occurs when two ontologies describe the same domain, at the same level of detail, but from a different perspective. For instance, maps with different purposes, a geological map and a political map do not display the same objects.
- **Semiotic heterogeneity** (also called **pragmatic heterogeneity**), is concerned with how entities are interpreted by people. Entities that have exactly the same semantic interpretation are often interpreted by humans with regard to the context and how they are ultimately used. This type of heterogeneity is difficult for the computer to detect and solve.

1.3.2. Terminology

It is observed that, in the area of ontology matching, the usage of some terms related to "***Ontology Matching***" differs frequently. Some authors use different terms to refer to the same concept, whereas, others use the same term to refer to different concepts. This is undoubtedly confusing. In this sub-section, we provide a glossary with the definitions of such terms as they are used in this project. They are adapted from [16].

Ontology Matching is the process of finding correspondences or relationships between entities of different ontologies.

Ontology Alignment is the output of the matching process. It is a set of correspondences between two ontologies.

Ontology Correspondence is the relation holding between entities (classes) of different ontologies according to a particular alignment.

Ontology Mapping is the oriented version of an alignment, i.e., it maps an entity from the first ontology to at most one entity from the second one.

Ontology Merging is the creation of a new ontology from two source ontologies. The initial ontologies are unaltered, and the merged ontology contains the knowledge of both of them.

1.3.3. The Matching Process

The process of matching ontologies determines an alignment *A'* for a pair of input ontologies *o* and *o'*. Other parameters extend the definition of the matching process:

(i) A: an initial alignment which is intended to be completed by the matching process;

(ii) P: matching parameters, e.g., weights, thresholds;

(iii) R: external resources used by this process, e.g., common knowledge and domain-specific thesauri.

Following this definition, the matching process can be schematically represented as illustrated in Figure 1.5.

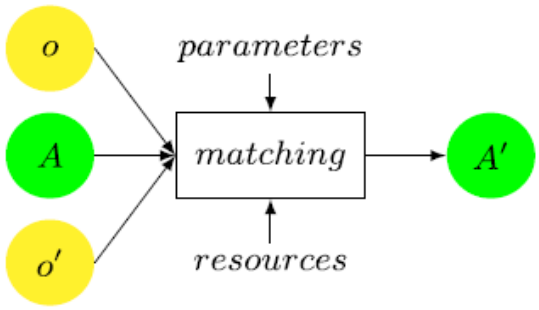

Figure 1.5. The matching process.

The matching process can be technically defined as follows [16]:

Definition 1.1 (*Matching process*). The matching process is a function *f* which, from a pair of input ontologies *o* and *o'*, an input alignment *A*, a set of parameters *p* and a set of resources *r*, returns an alignment *A'* between these ontologies:

$$A' = f(o, o', A, p, r) \quad (1.1)$$

1.3.4. Ontology Matching Techniques

Ontology matching consists of defining the semantically related entities between different ontologies, in order to solve semantic heterogeneity problems. For that object, several matching approaches have been developed in the scientific literature. Numerous classifications of the various ontology matching approaches are given as in [47]; [48]; [49]; [50]; [51].

Ontology matching process is generally based on measuring similarity between concepts of the concerned ontologies. The proposed matching algorithms and systems differ in their strategy for generating correspondences between ontologies. On this basis that we classify, in this section, the different ontology matching techniques and discuss the recent systems proposed by the state-of-the-art.

1.3.4.1. Terminological Techniques

Terminological techniques are based on comparing the strings or texts of the entities of input ontologies, i.e., names, labels, and comments of concepts, in order to compute similarities between them and match the similar ones. Thus, these methods are based on text similarity measures, and can be further classified into two sub-categories:

- ***String-based Techniques:*** consider strings as sequences of letters in an alphabet. They are based on the intuition that, the more similar the strings, the more likely they denote the same concepts [16]. There are several string-based methods used in matching systems as: *Hamming Distance, N-gram Similarity, Edit Distance, Levenshtein Distance, Jaro-Winkler Measure, TF-IDF, …etc*.
- ***Language-based Techniques:*** rely on *Natural Language Processing* techniques to extract the meaningful terms from the texts of concepts. This helps to assess the similarity between the to-be-matched entities of ontologies. Examples of these methods are, for example, *the use of linguistic knowledge as lexicons and thesauri, basing on linguistic relations like synonyms and hyponyms, Resnik Similarity, Jiang-Contrath Method, Cross-Translation, …etc*.

Terminological techniques are used by several ontology matching systems, such as, POMAP++ [52] [53], Lily [54], LogMap [55], RiMOM [56], XMap [57], Falcon-AO [58], AML [59], Eff2Match [60], FCAMapX [61], OLA [62] and AROMA [63].

1.3.4.2. Structural Techniques

Structural methods exploit the structural information (e.g., properties, subsumption relations, sibling concepts, …etc.) between ontology entities to derive correspondences. The typical intuition behind these methods is that, if the structural context or characteristics of the two compared concepts are similar, then they may be also similar. For example, if two classes are similar, their subclasses should also be similar. These techniques can be divided into two sub-categories:

- ***Internal structure techniques:*** exploits the internal structure of the entities, such as, their properties, types, keys, cardinalities of their attributes, …etc, to compute the similarity between them.
- ***External structure techniques***, also called ***relational techniques***: consider the relationships between concepts within the ontology structure, i.e., the other entities to witch the compared entities are related.

Structural ontology matching techniques are adopted by various systems, such as, COMA++ [64], AML [59], Falcon-AO [58], XMap [57], Anchor-flood [65], Lily [54], CroMatcher [66], LogMap [55], TaxoMap [67] and POMAP++ [52] [53].

1.3.4.3. Extensional Techniques

These techniques compute the similarity between concepts of input ontologies by comparing their extensional information, that is their instances (individuals). They are typically based on the intuition that, the more significant the overlap of common instances is, the more related the concepts they belong to are. In other words, if the instances are similar, then the concepts that they belong to should also be similar.

The effectiveness of these techniques depends on the availability of instances in ontologies, they are more efficient if more individuals are available. Some of the frequently used similarity measures for extensional techniques are: *Jaccard Similarity, Hamming Distance, K-statistic, Hausdorff Distance,* and many others.

Extensional techniques are used by several ontology matching approaches and systems, such as, ASMOV [68], AROMA [63], InsMT+ [69], SAMBO [70] and RiMOM [56].

1.3.4.4. Semantic Techniques

Semantic techniques are the most challenging type of ontology matching. They explore the semantic information encoded in the entities of the input ontologies. Their key feature is the use of model theoretic semantics to express the meaning of the compared entities without ambiguity. They are based on the intuition that, if two entities share the same interpretations, then they are semantically similar. They are based on logical models and deductive methods, using different strategies, such as, *Propositional Satisfiability, Description Logics Reasoning, Detecting Inconsistency and Repairing Alignment*. Although the meaning of "*semantic*" is not simple to define, and its exploitation in ontology matching has different forms, we present, in the following, some of these scenarios which intend to express the meaning encoded in the entities of input ontologies to achieve the matching task.

- Description logics can be used to take advantage of the semantic information of ontologies in order to discover inconsistent mappings so as to be removed from the final alignment set.
- Reasoning may be applied to expand relations between entities, and generate new relationships between them in order to discover new correspondences.
- Description logics can be further employed to transform the resulted alignment to the optimization problem on constraint programming [71].
- Entities of ontologies can be annotated by new semantic information extracted from background knowledge sources in order to discover new correspondences and reduce the heterogeneity between ontologies. For instance, names of concepts can be extended by new definitions.
- Several machine learning methods can be used to define correspondences between ontologies, such as, *neural networks, naïve bayesian learning, support vector machines, decision trees, …etc*.
- It is frequently applied to use the previous matching results and extend them in order to provide more efficient ontology matching results.

Examples of ontology matching systems which take advantage of semantic techniques are: CODI [72], RiMOM [56], ASMOV [68], AML [59], LogMap [55], XMap [57], FCAMapX [61], Yam-bio [73], CroMatcher [66] and Gomma [74].

1.3.4.5. Discussion

Very much work has been developed in the area of ontology matching. We have classified the ontology matching techniques above, and cited only examples for each type. Thus,

analysing more techniques should certainly lead to many conclusions and directions that ontology matching has opened. These challenges have been highlighted by numerous surveys as cited above. However, we focus on the most important conclusion that we can point out and discuss from the presented sample of the ontology matching algorithms and systems. That is, the majority of the ontology matching approaches do not rely just on one type of ontology matching techniques, but combine multiple matching strategies. Therefore, we conclude that there is a need for developing performant aggregation strategies. Furthermore, it is highly recommended to take advantage of each of the ontology matching methods with focus on semantic techniques.

Once an ontology matching approach has been developed basing on the techniques presented in this section, it must be evaluated in order to be improved by its designers, and then, to be put to application by interested users. In the following section, we discuss the procedure of evaluating ontology matching approaches and systems.

1.3.5. Ontology Matching Evaluation

Ontology matching methods and systems must be tested to evaluate their performance. This helps system designers to assess the strengths and weaknesses of their systems. In addition, it helps developers to choose the most appropriate techniques and algorithms for ontology matching. For this reason, different benchmarks, data sets and measures have been proposed for evaluating matching systems. We present below the evaluation principles, the different types of evaluation as well as the most commonly used evaluation measures.

1.3.5.1. Evaluation Principles

Euzenat & Shvaiko [16] have defined a set of principles that must guide the evaluation process, so as to be clear in the context of an evaluation framework. They are briefly outlined as follows:

- **Systematic procedure.** The procedure of evaluation should be reproducible. The evaluation results must be non-ambiguous. And, applying the evaluation procedure to different systems or to the same system at different times should be comparable.
- **Continuity**. Evaluation must be a continuous effort and not a one-shot exercise, in order to assess the evolution of evaluated systems, and to identify the progress made by the field.

- **Quality and equity**. The evaluation rules should be precise and well defined beforehand. In addition, the evaluation material should be of the best possible quality and not be biased towards a particular kind of algorithms.
- **Dissemination**. Evaluation should be available without excessive barriers. For this, the data sets and results of evaluation must be published and made as freely available as possible.
- **Intelligibility**. The evaluation results must be explained and able to be analysed and understood by everyone.

According to [75], evaluations are often based on three main steps:

1. **Planning.** It defines the task to be performed and its constraints.
2. **Processing.** It consists of executing the defined plan.
3. **Analysing.** It involves evaluation the results achieved in accordance with the planned measurements.

1.3.5.2. Types of Evaluations

Figure 1.6 illustrates the basic evaluation design. The evaluating process receives an alignment *A* and computes a (set of) quality measure (s) *m* by comparing it to the reference alignment *R*.

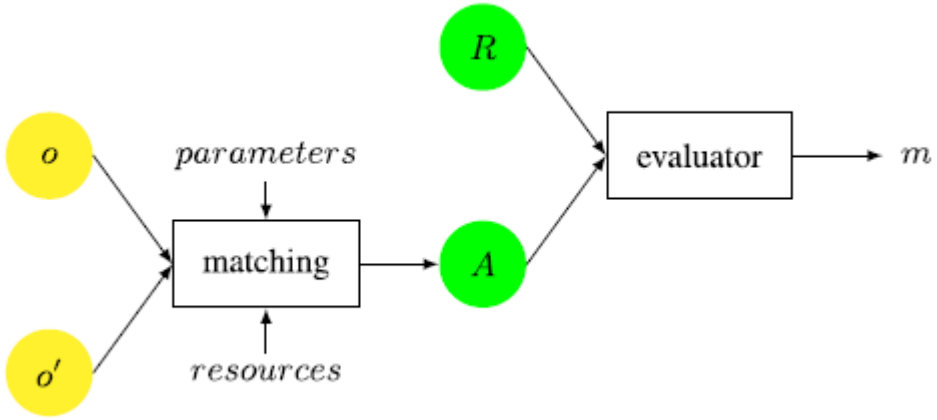

Figure 1.6. Basic evaluation design.

Several classifications of ontology matching evaluation may exist depending on the used criteria. Euzenat & Shvaiko [16] proposed following classification of evaluations basing on what they are supposed to evaluate.

1) Competence benchmarks

One particular type of evaluation is benchmarking. A benchmark is a well-defined set of tests on which the results of a system can be measured [75]. It is used for testing the improvement or degradation of a system, as well as for situating a system among other systems.

Competence benchmarks allow the characterisation of the level of performance (competence) of a particular system according to a set of evaluation tasks. They aim at finding the weak points and strong points of a system, i.e., characterising the kind of task this system is good for, or the type of input it can handle well, …etc.

2) Comparative evaluation

Comparative evaluation consists on comparing the results of several systems (or various versions of the same system) on a common task. Thus, such an evaluation requires a well-defined processing mode, and a clearly specified rules and evaluation criteria. It is also preferable to run blind (or nearly blind) tests. This kind of evaluation aims at improving the field as a whole in addition to individual systems.

3) Application-specific evaluation

Application-specific evaluation does not consider the matching in isolation, but compares the results of various systems evaluation on the output of a particular application. Such kind of evaluation is useful for a competitive evaluation, or for a company which wants to find the more adequate system to use in a real and particular application.

These three types of evaluation differ in their goals. In this study, we evaluate our approaches basing on both the first and the second type. This allows us to measure, our work, improve it and situate it with regard to a common stable matching framework.

1.3.5.3. Evaluation Measures

In order to evaluate the results of ontology matching algorithms, it is required to confront their produced alignments with a reference alignment based on some criteria (see Figure 1.7). For this reason, the evaluation measures represent another very important key to confidently evaluate the different matching systems. The reference alignment is designed either in a standard alignment format for the automatic evaluation, or entered manually by a domain expert. In the following, we review different possible measures for the evaluation of matching systems and algorithms. They are divided into compliance measures and performance measures [16].

1) Compliance Measures

Compliance measures intend to evaluate the degree of performance of returned alignments with regard to what is expected. We define below the most commonly used evaluation measures: ***precision***, ***recall*** and ***F-measure***. They are based on the comparison of the resulted alignment *A* against a reference alignment *R* (see Figure 1.7). *A* and *R* are considered to be sets of correspondences, being pairs of entities.

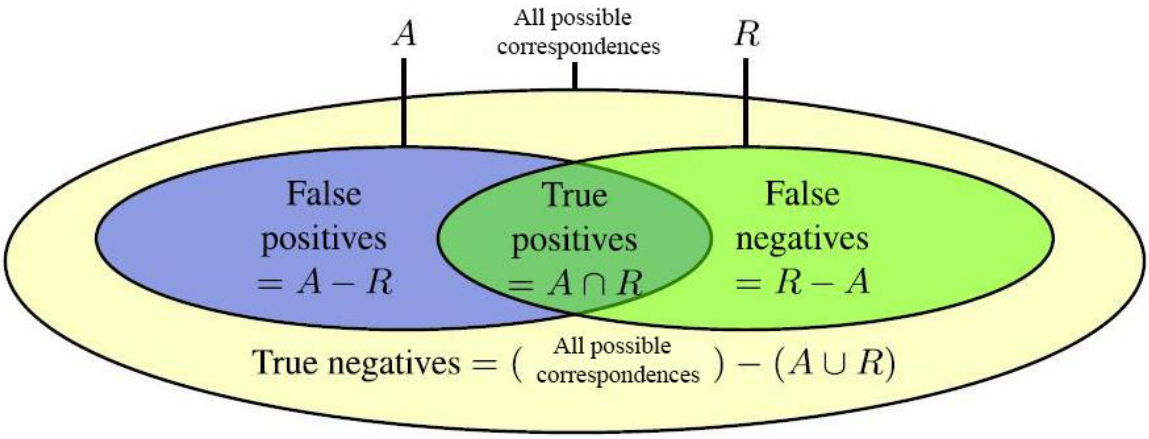

Figure 1.7. The resulted alignment (A) as compared-with the reference alignment (R).

- ***Precision:*** evaluates the degree of correctness of the algorithm. It measures the ratio of relevantly selected correspondences over the total number of selected correspondences (the *true positives* over the *true positives and false positives*; see Figure 1.7) as in formula (1).

$$Precision = \frac{|A \cap R|}{|A|} \tag{1.2}$$

- ***Recall:*** evaluates the degree of completeness of the alignment. It measures the ratio of relevantly selected correspondences over the total number of relevant correspondences (the *true positives* over the *true positives and false negatives*; see Figure 1.7) as in formula (2).

$$Recall = \frac{|A \cap R|}{|R|} \tag{1.3}$$

- Precision and recall are inversely proportional. A system which has higher precision may have a lower recall and vice versa. Thus, it is often preferable to use a unique measure for evaluation. For this purpose, another measure is introduced and also strongly used: F_b-measure, which aggregates precision and recall.

***F_b-measure*:** is a balanced score of precision and recall as given by formula (3).

$$F_b - measure = (1 + b^2) \times \frac{precision \times recall}{b^2 \times precision + recall} \quad (1.4)$$

If $b < 1$, then the F_b -measure biased to precision. If $b > 1$, then the F_b -measure is biased to recall. In between, if $b = 1$, then the F_b -measure combines precision and recall evenly, and does not compensate one for the other. In this case it is their harmonic mean as in formula (4).

$$F - measure = 2 \times \frac{precision \times recall}{precision + recall} \quad (1.5)$$

2) Performance Measures

Performance measures intend to check other features of matching which are related to the processing environment and the consumed resources. In this case, it is really important to run the compared algorithms under the same conditions (same memory consumption, same processor, …etc). We review some of these measures in what follows.

- **Speed.** It consists on measuring the amount of time required by the algorithms for achieve their matching tasks.
- **Network.** Some algorithms need to use network connectivity to perform the matching task. In this case, the network consumption can be measured in terms of bandwidth.
- **Memory.** It is also interesting to measure the amount of memory required and used by systems to achieve their matching processes.
- **Scalability.** It can be measured by a theoretical study, or by evaluation campaigns basing on quantified tests of increasing complexity [16].

1.3.5.4. Ontology Alignment Evaluation Initiative

The most well-known and used reference for evaluating ontology matching systems over several years is the *Ontology Alignment Evaluation Initiative*[3] (OAEI). It is an international initiative that aims at improving the quality of ontology matching systems by continuous comparison, using various tracks with different experimental designs. Its purpose is to compare different systems on the same basis, to identify their advantages and limits. OAEI has been evolving over the years. Since 2004, OAEI started and has been run yearly offering

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

various tracks and introducing new challenges for ontology matching evaluation. In this study, we evaluate our work according to the OAEI Campaigns.

Once an ontology matching approach or system has been evaluated and validated, it has to be put to work. This is described in the following section, presenting the wide need for ontology matching in several domains, and how their applications can take advantage of ontology alignments.

1.3.6. Applications of Ontology Matching

Ontology matching is a necessary operation in traditional applications, which are characterized by their heterogeneous models such as, ontology evolution, data integration and data warehouses. Ontology matching is also important in emerging applications, which are characterized by their dynamics, such as, linked data, peer-to-peer information sharing and query answering [16]. Therefore, ontology matching is a requisite operation tracing all domains of interest which run in heterogeneous environments. In this section, we overview different application domains that pose requirements for ontology matching.

1) Ontology Engineering

Ontology engineering is a context where users are confronted with heterogeneous ontologies. It is defined by the task of designing, implementing and maintaining ontology-based applications [16]. These activities require ontology matching support for the fact that ontology engineering deals with multiple, distributed and evolving ontologies. For example, in the process of designing ontologies, instead of creating or constructing a new ontology for a given application, it is much better to reuse ontologies (or parts of ontologies) that already exist. This allows to save the time required for ontology construction, which is a complex and time-consuming process, especially in the case of large-scale ontologies. In order to carry out this task of constructing ontologies basing on other already built ontologies, ontology matching is applied to identify the corresponding entities.

2) Semantic Web

The Semantic Web was created to ensure semantic interoperability between different sources of information expressed by ontologies [76]. However, these ontologies are heterogeneous and distributed. The matching of ontologies is the key to achieve this semantic interoperability, it allows to establish a set of semantic correspondences between heterogeneous ontologies. These mappings can be used for various tasks, such as navigating on the web, merging ontologies, translating data, browsing the web of data, ...etc. The

semantic web is considered a complex area where numerous matching applications can be encountered.

3) Information Integration

Information integration is a classic matching scenario that is already covered previously in databases. It is one of the oldest applications where matching is viewed as a solution. More specific problems, which require ontology matching, are found under this field, such as, schema integration, data integration, data warehousing and catalogue integration [16]. For instance (in schema integration), two enterprises want to perform a merger among them and integrate their databases into a single one. The first technical step is to identify semantic relationships between their related entities. This step is performed by applying ontology matching in order to merge the schemas databases.

4) Linked Data

Linked data is a part of the semantic web where data is described by instances and expressed in RDF [77]. These data are heterogeneous, and must be linked together to ensure semantic interoperability. Data interlinking can take advantage of instance-based ontology matching, which consists on performing the matching task basing on the instances (individuals or real objects) of the input ontologies.

5) Peer-to-Peer Information Sharing

Peer-to-Peer (P2P) systems are distributed communication models in which parties (peers) equivalently provide each other with data and services. They aim to the direct exchange of resources (text, pictures, videos, books, …etc.) between machines connected in a network. Examples of P2P file sharing systems are: messaging, telephone, Skype, ... etc. Peers are totally autonomous. Thus, their data are described using different terminologies and metadata models, even if they refer to the same domain of interest [16]. Ontology matching is used in order to allow a reasonable exchange of information between them.

6) Query Answering on the Web

Another application of ontology matching is the task of answering queries. Semantic search engines that use ontologies as support have better performance in query answering operations than traditional search engines. Furthermore, users employ their own terminology to query the web. Then, a system for semantic query answering on the web have to rewrite the query according to available ontologies so as to use reasoning for providing answers

[16]. Thus, ontology matching is used in order to match these entities to the concepts of the underlying ontology.

1.4. Conclusion

In this chapter, we have presented a detailed overview of ontologies and ontology matching. We have first reviewed the context of ontologies, and have defined them. Then, we have studied their components and the different languages used to express them. Afterwards, we have presented the motivations behind ontology matching and heterogeneity problems. We have also defined the matching process, and classify its different techniques and some of well-known systems. A detailed study of ontology matching evaluation has been also presented. And finally, we have overviewed the different fields where ontology matching can be applied.

If ontologies are of large size, then ontology matching systems should be more efficient due to the high heterogeneity of these ontologies. Besides matching quality, memory space and processing time are other challenges in this case. Thus, scaling up ontology matching systems to handle big ontologies remains a serious issue. This will be detailly studied in the next chapter.

Chapter 2

Large-Scale Ontology Matching

Contents

2.1. Introduction

As the heart of the semantic web, ontologies are used by a wide range of applications. For a same domain, various ontologies have been created by various people in different ways. This leads to heterogeneities at several levels. Ontology matching have frequently emerged in order to eradicate such heterogeneities, and numerous ontology matching systems have been developed for this purpose. Yet, most applications nowadays require using large ontologies like in the medical field. Therefore, when dealing with this large size, current ontology matching systems encounter many challenges like shortage of memory consumption and long processing time.

In this chapter, we present a state-of-the-art on large-scale ontology matching. First, we study the motivations behind this issue and the need for matching large ontologies. Then, we describe the different possible strategies to deal with large-scale matching. A classification of these techniques is also proposed. It includes partitioning-based techniques, parallel techniques and reuse of previous matching results. After that, we review the existing ontology matching approaches and systems which have been developed to handle the large-scale matching problems. And finally, we provide an analytical summary of these tools in order to identify their advantages and limits, and then, to outline the contributions of this work.

2.2. Need for Scaling Ontology Matching

Ontologies are the most commonly used model for knowledge sharing and reuse. Due to their importance, different ontologies describing different domains from different views have been developed. Therefore, multiple ontologies involve to represent the same or different domains with some overlapping information among them.

Ontology matching is the promising solution to this heterogeneity problem. It aims at generating correspondences between semantically related entities of the mismatched ontologies. Consequently, several approaches and systems have been developed in order to perform ontology matching.

However, with the increased evolution and pervasiveness of ontologies, ontology matching tools have to address additional matching challenges to establish high-quality mappings among ontologies within restricted computing resources.

Nowadays, ontologies can involve millions of concepts in many fields, like in medicine, astronomy, biology, …etc. In such areas, applications require several ontologies of huge sizes. Some examples of real-life voluminous ontologies with their descriptions and sizes are provided in Table 2.1.

Ontology	**Description**	**Size (# concepts)**
AGRonomy Ontology (AGRO)[4]	**AGRO** describes practices, techniques, and variables used in agronomic experiments. It is being built using traits identified by agronomists and other existing ontologies.	3,738
Drug Ontology (DRON)[5]	**DRON** supports comparative effectiveness researchers studying claims data. They need to be able to query U.S. National Drug Codes by ingredient, mechanism of action, physiological effect, and therapeutic intent.	617,960
Environmental conditions, treatments and exposure ontology (ECTO)[6]	**ECTO** describes exposures of humans, plants or any other organism to stressors, experimental treatments (e.g., temperature, lighting levels), stimuli and any kind of environmental condition or change in condition that can be experienced by an organism on earth. This is for purposes of public health and environmental monitoring.	13,289
Coronavirus Infectious Disease Ontology (CIDO)[7]	**CIDO** is an open-source community-driven biomedical ontology of coronavirus infectious disease. It is constructed for providing standardized and interpretable human-computer annotations and representations of various infectious diseases of coronavirus, including their transmission, aetiology, pathogenesis, prevention, diagnosis and treatment.	8,796
Cell Line Ontology (CLO)[8]	**CLO** is a community-based ontology in the field of biological cell lines with focus on permanent cell lines from culture collections. These cell lines are associated with terms	43,325

[4] https://raw.githubusercontent.com/AgriculturalSemantics/agro/master/agro.owl
[5] purl.obolibrary.org/obo/dron.owl
[6] https://raw.githubusercontent.com/EnvironmentOntology/environmental-exposure-ontology/master/ecto.owl
[7] https://raw.githubusercontent.com/CIDO-ontology/cido/master/src/ontology/cido.owl
[8] purl.obolibrary.org/obo/clo.owl

	from other ontologies, such as NCBI Taxonomy, Cell Type Ontology and Ontology for Biomedical Investigation.	
Ontology for Nutritional Studies (ONS)[9]	**ONS** is the first systematic effort aiming to provide a formal ontology framework for expressing nutritional studies.	6,056
Ontology for MicroRNA Target (OMIT)[10]	**OMIT** aims to provide common data elements and data exchange standards in the microRNA (miR) domain. Biologists and bioinformaticians can use OMIT to emerging semantic technologies in knowledge discovery and acquisition for more effective identification of essential roles performed by miRs in various diseases and biological processes of humans.	90,916

Table 2.1. Examples of large real-life ontologies.

In the previous chapter (Sect.1.3.5), we have presented two types of quality measures of ontology matching systems: *compliance measures* which intend to evaluate the degree of performance of returned alignments, and *performance measures* which intend to check other matching features related to the processing environment and the consumed resources. The key quality factor for small-scale ontology matching is the matching performance or quality. However, when ontologies are of a big size, additional features involve, and more techniques are required to handle the task of matching such large-scale ontologies by ontology matching tools. Thus, the main quality factors of ontology matching systems which deal with voluminous ontologies can be summarized in two major points:

- **Matching Quality.** Ontology matching quality seeks for evaluating the relevance of the matching results, by means of *precision*, *recall* and f_b*-measures*. Ontology matching systems should generate alignments that maximize these evaluation measures regardless of the size of input ontologies. However, increasing the sizes of ontologies penalizes the quality of the alignment generated by a matching tool [78]. The big sizes of input ontologies increase the need for more reasoning power. For instance, an ontology matching system has to be able to reason a high number of

[9] https://raw.githubusercontent.com/enpadasi/Ontology-for-Nutritional-Studies/master/ons.owl
[10] https://raw.githubusercontent.com/OmniSearch/omit/master/src/ontology/omit.owl

axioms in order to generate complete and accurate mappings. Otherwise, this large scale may decrease the matching quality with respect to the compliance measures.

- **Matching Complexity.** The large-scale matching problem is an extreme case in terms of complexity. This is due to the large sizes of ontologies which generate a high number of concepts pairs to be matched. Consequently, the matching algorithms for large ontologies could be inefficient. At this stage, large ontologies pose two key challenges face to ontology matching tools:
    - ***Demand for More Memory Space.*** Matching entities of two input ontologies intends to compare each entity of the first ontology against all entities of the second ontology. Assuming that each input ontology has n entities, this Cartesian product of matching these entities results in a memory space complexity of $O(n^2)$. In addition, most ontology matching systems integrate multiple matchers in order to improve the quality of their produced alignments. Therefore, this complexity would be further multiplied by the number of executed matchers. An ontology matching process with a space complexity of $O(n^2)$ can easily lead to an out-of-memory error in case of a large n [79].
    - ***Demand for More Processing Time.*** For ontology matching algorithms, how long they take to be run to completion is a very important issue. Efficient matching algorithms must complete the matching process and keep the processing time to a minimum. Similar to memory space, matching two input ontologies has a time complexity of $O(n^2)$ (with n entities for each ontology), obviously multiplied by the number of matchers that compose an ontology matching tool. Unfortunately, a process of such a complexity would require users to significantly wait for obtaining the matching results.

Therefore, in order to accurately match large ontologies within the limited computing resources, ontology matching systems must use strategies to reduce the high space and time complexities associated with the ontology matching process [79]. Thus, achieving both *high quality* and *low complexity* is the key challenge in large-scale ontology matching. We address this challenge in this work. Firstly, we review and study the techniques that the existing ontology matching tools employ to establish high quality alignments when matching large-scale ontologies. This is presented in the next section.

2.3. Large-Scale Ontology Matching Techniques

Although a wide number of systems have been proposed and developed in order to address the small ontology matching issue [47]; [50]; [51], large-scale ontology matching still presents several challenges. Current ontology matching systems have to deal with big ontologies containing thousands to millions of entities each. Thus, they suffer from some difficulties related to memory consumption and processing time at the large-scale scene.

In order to handle this issue and develop a performant ontology matching system adapted to the requirements of big ontologies, we must study the different techniques and approaches that deal with large-scale ontology matching. In the following, we provide a classification of these techniques. Recent works on large-scale ontology matching can be categorized into three main directions: *partitioning-based matching, parallel matching* and *reuse of previous matching results*.

2.3.1. Partitioning-based Large-Scale Ontology Matching

2.3.1.1. Ontology Partitioning for Ontology Matching

When dealing with large ontologies, it is beneficial to split the matching problem into a set of smaller sub-problems in order to reduce the matching space. Partition-based ontology matching approaches aim at partitioning the input ontologies into smaller sub-ontologies in such a way that, the matching process is independently performed by partitions, and the independent partial results are then combined to obtain the overall matching result.

This process is called blocking in other contexts [16]. The idea is to avoid the Cartesian product of comparisons among the large ontologies, and match entities only from corresponding partitions or blocks. Matching small ontological partitions requires less memory and time resources compared to matching full ontologies. Thus, it results in a significant reduction of the matching complexity to $O(n^2/k)$ if the large ontologies are partitioned into k blocks. Ontology partitioning improves the performance of applications for the reason that it reduces the irrelevant data to be accessed and shared among different nodes in a distributed system [80]. To further improve the matching efficiency, it is possible to perform the sub-matching tasks in parallel.

2.3.1.2. Partitioning Methodology

The partitioning framework for matching large ontologies is depicted in Figure 2.1. In general, three major stages involve in this procedure. They are described as follows:

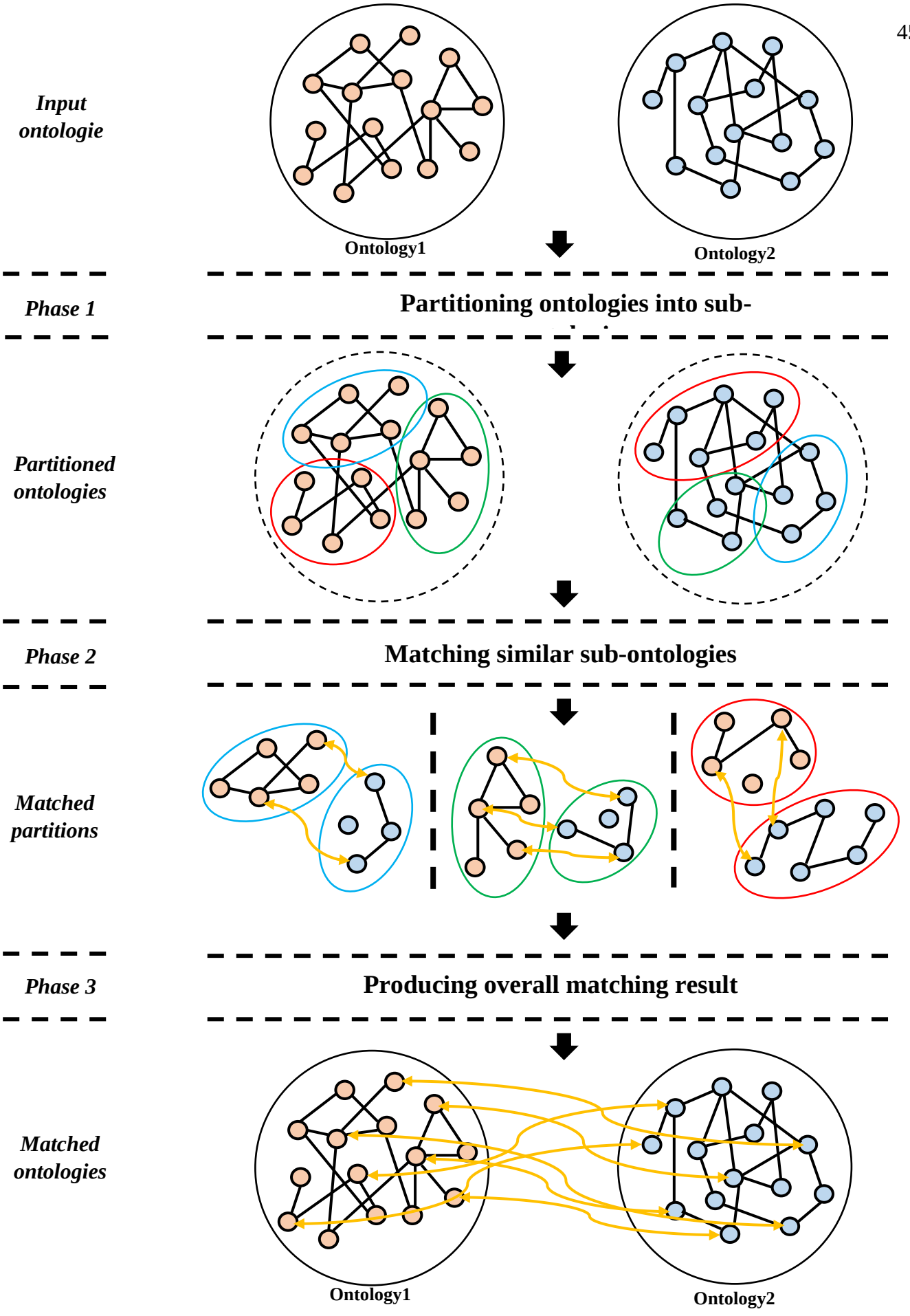

Figure 2.1. Partitioning framework for matching large ontologies.

Phase 1. Partitioning ontologies

Two voluminous ontologies are entered as input. This step consists on dividing each ontology into a set of smaller and disjoint sub-ontologies. Ontology concepts which are similar according to specific aspects (linguistic, structural, … etc.) are grouped together in a partition.

Phase 2. Matching partitions

This phase consists on doing the matching operation on similar partitions. It is performed in two main steps:

Step1. Identifying similar sub-ontologies

In this step, the most similar partitions which worth to be fully matched later are identified. After having divided the large ontologies into smaller sub-ontologies, it is simply possible to compare each sub-ontology of *Ontology1* with all sub-ontologies of *Ontology2*. But such calculations are of high complexity. The possible solution to avoid comparing all partitions and reduce this complexity is to match only the similar sub-ontologies. This step is also known as the filtering step [81], since it involves removing the pairs of dissimilar partitions.

Step2. Matching sub-ontologies

In this step, the matching techniques are applied in order to establish semantic correspondences between partitions. It performs comparisons among concepts of sub-ontology pairs that have been identified as similar in the previous step. For each pair of two similar sub-ontologies, each concept of the first one is compared with all concepts of the second one.

Phase 3. Discovering alignment

This step consists on combining and aggregating the partial results obtained from matching sub-ontologies in order to produce the overall matching result (e.g., by the union on partial correspondences). At this stage, it is also possible to refine the partial alignments and remove all inconsistencies and replications of correspondences.

2.3.1.3. Ontology Partitioning Algorithms

There are considerable possible ways to perform partitioning of large ontologies into several sub-ontologies. In the following, we describe the methods: *Modularisation, Summarization, Clustering* and *"Divide and conquer"*.

1. Modularisation Techniques

Modularising an ontology consists on identifying a set of components (modules) of that ontology which are considered as discrete parts but can be linked to each other. A module is composed of a minimum set of axioms (sub-class, equivalence, instantiation, etc.), which maintain its own entities and relations with encapsulating characteristics. For instance, the relations of a given concept are within that module, not belonging to another module [82].

There are diverse methods that deal with extracting modules from ontologies.

- Grau et al. [83] used E-connections [84] as a basis for the modularisation of large ontologies. E-connection is a set of partitioned knowledge bases which has been made up of ontology using the description logic [82].
- Garcia et al. [85] conducted the modularisation of large ontologies basing on the partitioning techniques of the graphs from the iGraph library[11].
- Similarly, authors in [86] [87] used logic-based approaches to extract the modules of ontologies.

2. Summarization Techniques

Summarization of an ontology provides a summary of that ontology as a smaller and more compacted ontology. In other words, the summarized ontology is a new version of the large ontology in such a way that, it covers all its main concepts and provides all its important information. Summarizing an ontology helps for rapid understanding and facilitating the engineering works on ontologies [88].

Several methods for ontology summarization exist.

- Peroni et al. [89] employed criteria such as the name, coverage surface and density, in order to extract key concepts of ontologies such as ontologies summarization.
- Li et al. [90] explored the most important characteristics of ontologies that are required to be included and covered in the summarized ontology.

[11] http://igraph.sourceforge.net/

- Zhang et al. [91] proposed a summarization method based on RDF sentences. It consists on building a graph where RDF sentences are the nodes and the links between them are the edges. Then, the centrality measure is computed as a proportional importance for each node.

3. Clustering Techniques

Clustering is the simplest and the most commonly used method for matching large ontologies [92]. It consists on dividing the huge ontology into several clusters using different techniques. This leads to significantly reduce the search space, to minimal memory requirements and thus to improve the matching efficiency [92].

Numerous approaches deal with ontology clustering.

- Algerygawy et al. [93] proposed a graph clustering method based on the structural similarity of graph nodes and their connections. The nodes in a cluster are similar to each other, whereas, nodes in different clusters are dissimilar. This idea of structural similarity has been derived from the AHSCAN algorithm [94], assuring that nodes with similar connections in a network would have very high structural similarity.
- Ahmed et.al. [95] introduced a new semantic similarity measure and proposed an enhancement and a revision of K-means clustering algorithm.
- Also, Tran et al. [5] semantically partitioned the large ontologies into clusters. The information content [96] of each entity is used for assessing the semantic similarity between the concepts.

4. "Divide and Conquer" Techniques

The "Divide and Conquer" technique consists on solving the large ontology task by breaking in into smaller sub-tasks, solving the sub-tasks and combining them to get the desired output.

The idea of this technique has been applied by several processes in the large scene.

- The PBM method [97] used a "divide and conquer" strategy to divide the large ontologies. It first partitions each ontology into small and independent blocks basing on linguistic and structural similarity and using the ROCK algorithm [98]. After that, the weighty links are defined by a structural proximity. Then, ontologies are divided into blocks according to these weighty links, and using two criteria which are the *cohesiveness within blocks* and the *pairing between blocks.*

2.3.2. Parallel Large-Scale Ontology Matching

When dealing with large ontologies, ontology matching requirements for different resources are even more increased. Ontology partitioning does not guarantee the solution of problems related to memory consumption and execution time [99]. As a result, parallel ontology matching techniques emerge as a complementary solution. A straight-forward solution to reduce the processing time of large-scale ontology matching is to run the matching process in parallel on several processors.

Parallel matching techniques aim at minimizing the execution time of large-scale ontology matching by distributing the concept comparisons among available resources of a distributed system [100]. According to [100], parallel matching techniques can be categorized into two major classes: *inter-matcher parallelization* and *intra-matcher parallelization*. Figure 2.2 illustrates these techniques. In order to achieve an effective ontology matching, it is highly required to determine several similarities between ontologies and combine multiple matchers.

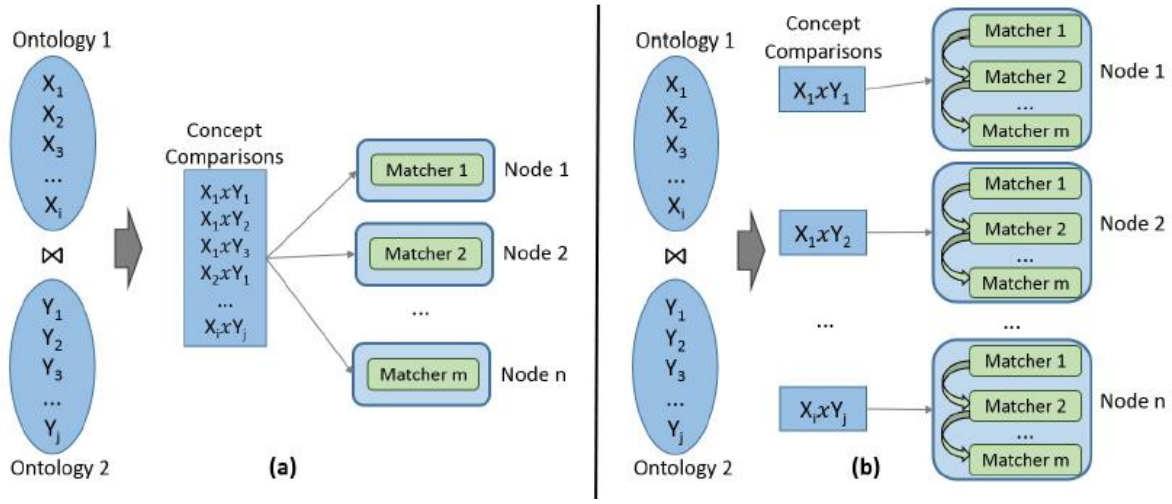

Figure 2.2. Inter-Matcher Parallelization (a) and Intra-Matcher Parallelization (b) [101].

2.3.2.1. Inter-Matcher Parallelization

The workflow of this matching type allows the parallel execution of independent matchers on a parallel platform. In inter-matcher parallelization (Figure 2.2 (a)), each matcher is executed on a different node (computer, virtual machine, …etc.) of a distributed computing infrastructure (e.g., cloud machines). The comparisons of concepts to be compared are performed by each matcher in parallel.

This kind of parallelization techniques is easy to support, and able to reduce the execution time by a factor of *n* if matchers are of similar complexity. However, it is limited

by the number of independent matchers. Also, the matching requirements for memory space are not reduced since the different matchers consider the complete ontologies [92].

2.3.2.2. Intra-Matcher Parallelization

Intra-matcher parallelization deals with the internal parallelization of individual matchers. It is typically based on decomposing the input ontologies in order to provide a set of smaller matching tasks which can be executed in parallel. Moreover, intra-matcher parallelization can be combined with inter-matcher parallelization, i.e., it can be applied for both sequential and independently executable matchers [100].

In this kind of parallel matching (Figure 2.2 (b)), each pair of concepts is compared at an available node by multiple matchers sequentially, whereas, comparisons between pairs of concepts are performed on different nodes in parallel. Therefore, the communication between matchers is possible as they are located at the same resource [101].

2.3.3. Reuse of Previous Matching Results

A lot of effort has been put in the development of ontology matching approaches and systems. A promising approach to enhance both the effectiveness and efficiency of ontology matching is the reuse of previous matching results for accomplishing a new matching task. This idea has been firstly introduced by [102] in order to improve schema matching systems, and then used for ontology matching.

The reuse of previously identified correspondences and matching results leads to a significant reduce of the matching effort. It is necessary to determine the fragments of identified correspondences on which the matching reuse is applicable. Also, exploiting this reuse requires a comprehensive infrastructure or a repository to maintain the already established matching results.

2.4. Related Literature

In the previous section, we have studied the possible solutions and techniques to handle the task of matching voluminous ontologies. In this section, we review and analyse the existing tools and methods currently available for dealing with large-scale ontology matching.

2.4.1. Review of Large-Scale Matching Tools

Some approaches and systems aim for addressing the problems of matching big ontologies. We review the related work in the following.

- Tran et al. [5] proposed a partitioning approach to break up the large ontology matching problem into smaller matching sub-problems. They first semantically split anatomy ontology into groups called clusters. Basing on the information content of their concepts, and a scalable agglomerative hierarchical clustering algorithm. They use then a filtering method to select the possible similar partitions in order to reduce the computation time.

- The study of Laadhar et al. [6] presented a local matching learning strategy to align large and complex biomedical ontologies, combining ontology partitioning with machine learning techniques. It defines a new partitioning approach, based on the hierarchical agglomerative clustering [103], which breakups the large ontology alignment task into a set of local sub-matching tasks. Instead of defining a global machine learning model for the entire ontology matching task, it performs a machine learning model for each local sub-matching task and provides its corresponding training set, which is automatically generated by exploring the external biomedical knowledge bases without any gold standard or user involvement. Therefore, each proposed local matching learning model automatically provides adequate matching parameters for every local sub-matching task.

- The study of Balachandran et al. [9] is based on graph partitioning to improve the execution time of ontology mapping process. The proposed ontology mapping process works in three consecutive phases. First, the cluster-walktrap methodology is used to partition the ontologies into sub-ontologies and identify the correspondence between the concepts in parallel. Then, the factored ontologies are represented in vector space model and similarities are computed between concepts. Finally, a collaborative decision on the mappings is generated, taking into account the similarities of the previous phase.

- The MOM (Modularisation-based Ontology Matching) method [104] is an approach which decomposes a large matching problem into several small problems using E-connections. Then, finding the similar module pairs is treated as a problem of finding the maximum bipartite match. Finally, MOM makes use of the OPM (Ontology Parsing graph-based Mapping) method in order to obtain correspondences between two similar modules.

- The approach proposed by Jiménez-Ruiz et al. [8] consists on splitting the ontology matching task into smaller and more tractable matching subtasks, basing on a lexical index and locality modules. Two clustering strategies are presented for the lexical index. Naive strategy relies on a simple splitting method, to randomly divide entries into a given number of clusters of the same size. And, neural embedding strategy relies on a log-linear neural embedding model. It aims to reduce the global size of the computed division of the matching task.

- In [105] [106], Lambrix et al. proposed to reuse a partial reference alignment in different ontology matching steps. First, it is used in the pre-processing step. Then, in the computation step, they compute similarity values basing on similar pattern of entity pairs in the reference alignment. And finally, the reference alignment is also used in the filtering step to filter mapping suggestions.
- The study in [107] proposed an innovative method of matching large ontologies based on filter and verification. It includes two phases: filter phase and verification phase. In the filter phase, it reduces the degree of heterogeneity and scale of ontology and then, in the verification phase, it matches the reduced ontologies. Specifically, the input ontologies are partitioned into several sub-ontologies to get a proper scale before matching. Similarities of irrelevant entities pairs are recognized beforehand and not calculated in subsequent steps. Then, the extracted sub-ontologies are matched. And finally, the alignments resulted from matching sub-ontologies are integrated to provide the final output.

- Laadhar et al. [7] proposed an approach that applies the hierarchical agglomerative clustering technique to divide an ontology into a set of partitions. Then, it uses an automated tuning process, which generates the adequate thresholds of the available similarity measure for any biomedical matching task.

- Kachroudi et al. [108] introduced an ontology partitioning method towards the final goal of large ontology matching. It consists on splitting both ontologies to be aligned in two reduced size coherent block sets, to deal only with blocks of manageable size, containing the elements capable to be matched. The proposed method is mainly based on the RDF transformation techniques, creation and processing of ontologies graphs, and on semantic-structural similarities.

- COMA [109] and its successor COMA++ [110] support the reuse of complete matching results. COMA++ is a system for combining matching algorithms in a flexible way [110] [111]. It deals with different kinds of metadata models, such as relational, RDF, XML and OWL ontologies. COMA++ proposed *fragment matching*. A fragment is a rooted subgraph which can be determined either by the user in an interactive way of matching, or by using a schema, sub-schema, or a shared strategy in an automatic way. Using these simple rules is efficient for matching XML schemas more than ontologies, for the reason that, they produce a great number of fragments when dealing with ontologies. COMA++ uses a lightweight similarity metric in order to identify similar fragments [111].
- Authors in [112] proposed V-Doc+, a parallel approach based on the MapReduce framework and the virtual document technique [113], for large-scale ontology matching. In the first stage, it performs two MapReduce processes in order to extract the textual descriptions of ontology entities (classes, instances and properties) and blank nodes. The extracted descriptions are exchanged in the second stage with neighbors in RDF graphs so as to construct virtual documents. In the third stage, a word-weight-based partitioning technique is proposed for parallel similarity computation using the TF-IDF model.
- The work in [114] proposed a modularisation technique for ontology matching. It extracts fragments from the input ontologies that contain only the essential classes and relations in order to resolve the detectable incoherence. The approach introduces also a global alignment repair algorithm which minimizes the degree of incoherence as well as the number of mappings removed from the alignment. Thus, it aims for overcoming the matching scalability problem by applying the proposed modularisation technique [114].
- Hu et al. proposed Falcon-AO [115], a "divide and conquer" system for solving the scalability problem of ontologies so as to match large ontologies. It first computes the structural similarity between classes and properties based on three types of hierarchies: *subClassOf*, *rdfs:subPropertyOf* and *rdfs:subDomains*. Based on the computed structural similarities, Falcon-AO develops an agglomerative algorithm for partitioning the input ontologies into two sets of partitions. Finally, it captures the whole sub-ontologies descriptions in order to identify similar partitions and compute the similarity between them.

- The corpus-based matching approach of Madhavan et al. [116] reuse the previous matching results. It consists on augmenting ontology elements with matching elements from a domain specific corpus of schemas. The idea is that, two ontology elements match if they match with the same corpus elements. A machine learning model based on several matchers is employed to find matches between ontology and corpus elements. Thus, a substantial effort is required for learning the models and defining the matches, especially for large ontologies.
- The work in [101] investigate the parallelization for the complex problem of matching large ontologies, and proposed a MapReduce-based ontology matching approach which distributes the computation of similarities between concepts on different nodes of a cloud computing infrastructure. This approach is based on intra-matcher parallelization. Assuming that input ontologies are partitioned and the pairs of subontologies to be compared are identified, this approach is used to parallel the correspondences computation aiming to reduce the execution time of the matching process. The workflow of this approach is composed of four phases: reading, map, shuffle, and reduce. For each pair of sub-ontologies, the concepts of the smaller sub-ontology are replicated in the map phase. In the reduce phase, the concepts of the larger sub-ontology are sent to be compared with these replicated concepts to be compared. Similarities between each pair of concepts is computed by matchers. And, the pairs of concepts which have a similarity value higher than a defined threshold are selected.
- SeeCOnt is a seeding-based clustering technique which aims at reducing the comparison complexity basing on cluster seeds. The seeds of clusters are first identified basing on the highest ranked concepts using a distribution condition. Then, the remaining concepts are assigned to their proper clusters using a membership function [117].

2.4.2. Analytical Summary

Large ontologies introduce several challenges to ontology matching. Previously, we presented a state-of-the-art on large-scale ontology matching approaches and systems. In the following, we discuss and study these tools in order to identify their advantages and limits, and outline contributions of this study.

A summary of the presented systems for large-scale ontology matching is offered in Table 2.2 and translated in Figure 2.3. They are classified basing on their strategies for dealing with the problems of large-scale matching. We did not include the results of their evaluation because they had not been evaluated under the same conditions.

Ontology Matching Tool	**Partitioning-based Matching**	**Parallel Matching**	**Reuse of Matching Results**
Tran et al. [5]	×		
Laadhar et al. [6]	×		
Balachandran et al. [9]	×	×	
The MOM method [104]	×		
Jiménez-Ruiz et al. [8]	×		
Lambrix et al. [105] [106]			×
Li et al. [107]	×	×	
Laadhar et al. [7]	×		
Kachroudi et al. [108]	×		
COMA [109]; COMA++ [110]			×
V-Doc+ [112]		×	
Santos et al. [114]	×		
Falcon-AO [115]	×		
Madhavan et al. [116]			×
Ara et al. [101]	×	×	

SeeCOnt [117]	×		

Table 2.2. Summary of large-scale ontology matching tools.

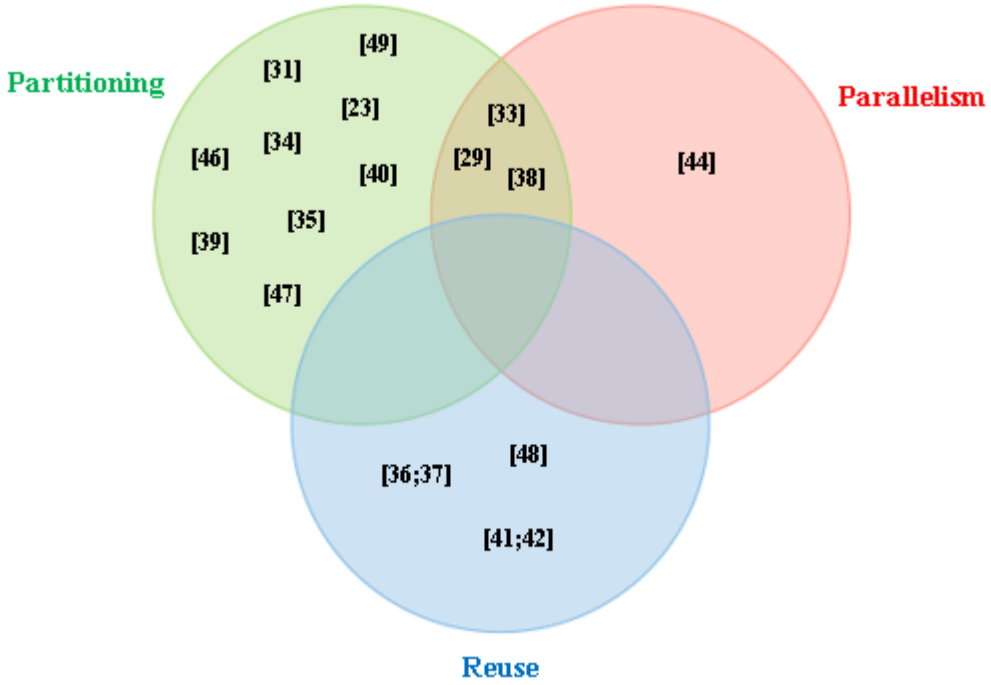

Figure 2.3. Classification of large-scale ontology matching tools.

From the above classification and summary, we conclude the following:

- Large-scale ontology matching still presents a real challenge because it is a time consuming and memory intensive process. Even though several systems aim at enhancing the performance of the ontology matching task, there are only few systems that can handle the heterogeneity between large ontologies. Due to this fact, much more work is required in this field.
- Partitioning large ontologies is the wide commonly used solution to deal with identifying semantic correspondences between different ontologies at the large scene. Ontology matching techniques that deal with this problem search to divide the large ontologies into small sub-ontologies in order to reduce the matching space.
  However, partitioning ontologies also suffers from interesting challenges:
  - Whatever the effectiveness of ontology partitioning, it may decrease the matching quality, owing to the fact that, several semantic links inside ontologies are expected to be lost in the matching process, while they actually exist.

- Moreover, the partitioning parameters (number of partitions, size of each partition, number of elements per partition, how to divide ontologies, how to align these divisions, …etc) are also challenging and affect the matching performance. For instance, ontology partitioning should produce partitions of optimum size, i.e., they should not be too small hence increasing the complexity of matching the produced partitions, nor too large hence not taking maximum benefits of ontology partitioning.
- Furthermore, ontology partitioning also suffers from the high complexity while creating partitions. Ontology partitioning algorithms should produce partitions that maintain the knowledge expressed by the original ontologies. Thus, a correct partitioning process require a high time and space complexity to be completely achieved. This has a direct impact on the efficiency of the ontology matching process.

- Parallel large-scale ontology matching is also used by ontology matching systems. However, it is not usually employed on pairs of concepts among ontologies but on ontology partitions. Thus, it is efficient, but still has strict demands for ontology partitioning techniques. It emerges as a complementary solution to the gaps of partitioning-based ontology matching.
- Unlike partitioning-based and parallelism-based ontology matching approaches, reusing the previous matching results for large-scale ontology matching is an independent category, i.e., algorithms of this matching type do not require partitioning or parallelism techniques. This is due to the fact that, this type of large-scale ontology matching provides both very high matching quality and very low matching complexity. However, the algorithms of this large-scale matching category are not autonomous and depend on other ontology matching tools. Also, they undergo some other difficulties, for instance, the candidate ontology matching results must be obtainable, expressed in the same format and have been evaluated on the same basis. For this, reusing the previous matching results is not very used in ontology matching.

2.5. Conclusion

In this chapter, we have presented a state-of-the-art on matching large ontologies. First, we describe the need for scaling the ontology matching process. Then, we present a detailed classification of the possible methods to deal with large-scale ontology matching. After that,

we present an overview of the existing ontology matching tools which have been developed to address the challenges of large-scale ontology matching. Finally, we have studied and discussed these tools with regard to their advantages and limits, and identified the current challenges which serve to outline our contributions.

This work seeks for addressing these challenges. For this, powerful computational mechanisms are required to fix the identified gaps of the existing large-scale ontology matching techniques. Deep learning techniques are very appropriate for dealing with large datasets. They have the ability to analyse and interpret massive amounts of data, that require effective and efficient computational tools. Thus, they have been widely used to solve complex tasks in many research axes. In the next chapter, we study deep learning techniques and present a state-of-the-art on their use for ontology matching.

Chapter 3

Deep Learning for Ontology Matching

Contents

3.1. Introduction

Large-scale ontology matching is still challenging. Such powerful computational mechanisms are required to address the posed challenges. Deep learning is a promising avenue of research and an important step toward artificial intelligence, making machines independent of humans, and emulating the human brain's mechanisms and ability to observe, learn, make decisions and analyze, especially for extremely complex problems. Deep learning algorithms have been particularly successful when dealing with high-dimensional and massive amounts of data. They have attracted much attention from researchers in recent years due to their performance and efficiency to solve complicated problems in many research domains such as computer vision, natural language processing, speech recognition and many others.

In this chapter, we present a state-of-the-art on employing deep learning techniques for ontology matching. It is organized into two major parts. In the first part, we describe the basic concepts of deep learning, and the second part is dedicated to the related literature. First, we introduce the challenges that have motivated deep learning. Next, we present the basic model family in deep learning which are artificial neural networks. We describe the biological inspiration of artificial neuron, the fundamental neural network architectures, activation functions and learning methods. Then, we present deep learning architectures with focus on auto-encoders and embedding models. After that, we review the existing ontology matching tools based on artificial neural networks, and particularly on deep neural networks. Finally, we discuss the presented techniques and provide an analytical summary which allows to outline contributions of this work.

3.2. Deep Learning Basics

Artificial intelligence field is actively growing and involving several applications and different challenges. Artificial intelligence aims at conceiving intelligent machines able to understand and solve problems that are difficult for human beings but not for machines.

Deep learning has made major advances in solving such complicated problems that have been exceedingly hard to fix by the artificial intelligence community for many years. It has produced very promising results for several tasks in many fields. In this section, we review the basic deep learning concepts that are necessary to describe these powerful computational models and their functionalities.

3.2.1. Challenges Motivating Deep Learning

Deep Learning has gained a lot of popularity and attention in the last years. This is due to the factors that we outline as follows.

- **Data availability.** With the significant evolution of the Internet, the number of its users and the content which they generate is exploding. This has resulted an increase of huge resources and large datasets that significantly help the learning procedure of deep learning paradigms.
- **Computational power.** Deep Learning algorithms require a considerable computational power (processor and memory) to efficiently run to solve complex tasks. Unlike bygone, users have the possibility to access more powerful computational resources. Particularly, the exploitation of GPU computing arises providing high computing power more than CPU. GPUs work more efficiently and make use of massively parallel processors in order to accelerate computations.
- **Improved learning algorithms.** In the last few years, novel training improvements have been proposed for increasing the performance of solving problems with deep learning techniques. These enhancements include more efficient activation functions, more robust optimizers and regularization techniques able to competently prevent overfitting [118].
- **Real-world impact.** Deep learning techniques have an impressive advancement in several tasks, such as object recognition, robotics, speech recognition, machine translation, …etc. Moreover, deep learning models have gained popularity to be used by many large technology companies in the world, like Google, Facebook, Microsoft, Apple, IBM and NVIDIA Corporation.

Deep learning provides technical, innovative and efficient solutions for a large variety of problems and domains. This would not have been possible without the advancement in the most dominant model family of neural networks that we introduce in the next section.

3.2.2. Artificial Neural Networks

Artificial Neural Networks (ANNs) are the fundamental model family in Deep Learning. Nowadays, neural networks have been applied to very different issues and in various fields due to their efficiency and ease of use.

3.2.2.1. Biological Inspiration

Artificial neural networks are based on intrinsic models of *biological* neural systems of information processing, which have led to the development of more intelligent computer systems, applicable in statistical problems and data analysis. Neural networks appeared during research on artificial intelligence, to replicate the "ability to learn" found in neural biological systems, by modeling the intrinsic structure of the brain.

The ***brain physiology*** shows that it is made up of a very large number of ***neurons*** (around one hundred billion), connected to each other by several thousand interconnections for each neuron. Each neuron is a specialized cell, capable of creating, sending and receiving electrochemical signals. Like all biological cells, neurons have a ***cell body*** (also called ***soma***), extensions providing information to the neuron (***dendrites***), and an extension which communicates the information collected by the neuron (***axons***). The axon of one cell is connected to the dendrites of another through a ***synapse***. When a neuron is activated, it sends an electrochemical signal through the axon. This impulse, which is of the order of 1 millisecond and its amplitude of about 100 millivolts, crosses the synapses to thousands of other neurons, which can in turn send and therefore propagate the signal to the whole of the biological brain. A neuron will only emit an impulse if the signal transmitted to the cell body by the dendrites exceeds a certain threshold called the ***trigger threshold*** [119]. Thus, neurons can create or accomplish various and extremely complex cognitive tasks such as learning.

3.2.2.2. Artificial Neuron

The artificial neuron is an information-processing model of the brain neuron. Figure 3.1 illustrates the biological inspiration of the artificial neuron. The main features of real neurons are retained in the definition of an artificial neuron: the inputs model the dendrites; the input impulses are weighted by synaptic coefficients and the emitted impulse (the output) obeys a threshold effect. Typically, the artificial neuron is the basic unit to process information, and it is composed of the following elements:

- The neuron receives signals from different sources. The *input nodes* are expressed as an n-dimensional vector $X \in \mathbb{R}^n$.

- The information flows over the *connection links* which are associated with importance values represented by *weights*. The greater the value of a weight *w*, the stronger the intensity of the incoming signal, and therefore, the more influential the corresponding input. Likewise, these weights (that can include both positive and negative values) are expressed as a real-valued vector $W \in \mathbb{R}^n$.
- A supplementary signal, called *bias*, can be added to the input. This parameter $b \in \mathbb{R}$, which is generally set to 1, has an important effect during the learning phase since it enables increasing or decreasing the neuron value.
- The *summing function* generates a weighted sum of all the received signals weighted by their connection strengths each. It can further select the maximum, minimum, product, majority or other normalizing algorithms.
- Finally, the *activation function* $a: \mathbb{R} \rightarrow \mathbb{R}$, is applied on the weighted sum in order to define the final output signal *Y*. This mathematical function, which is usually non-linear, can also scale and control the output value via thresholds [118].

Thus, it is possible to define an artificial neuron as:

$$Y = a(WX + b) = a(\sum_{i=1}^{n} (w_i x_i) + b) \tag{3.1}$$

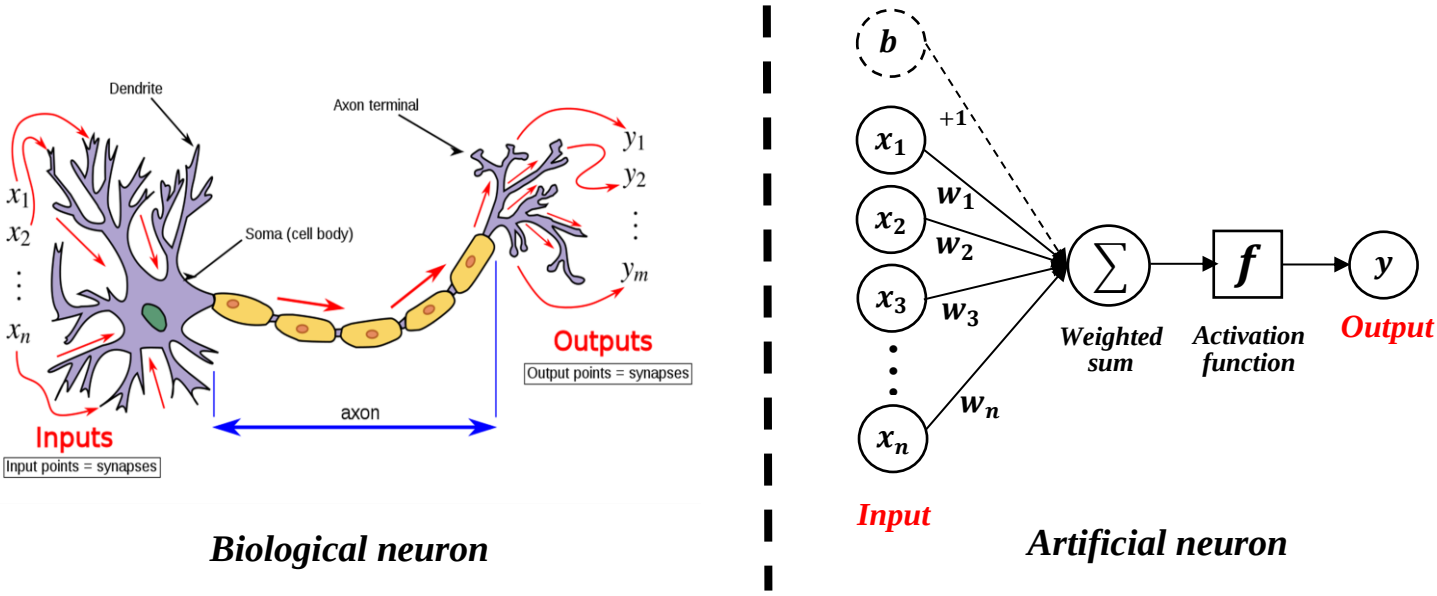

Figure 3.1. Biological neuron (adapted from[12]) Versus Artificial neuron.

[12] https://en.wikipedia.org/wiki/Biological_neuron_model

An artificial neuron is the basic processing unit of a neural network. It is connected to input information sources (other neurons for example) and returns output information. The neuron is the building block of more complex models.

Therefore, an artificial neural network is composed by a connection of artificial neurons denoting a weighted and directed graph. The nodes represent the neurons, and the connection links pass on the weights between neurons. The weights are first randomly initialized and then adjusted during the learning phase [118].

In the next sections, we describe the fundamental neural network architectures, the activation functions and the learning methods used to train the artificial neural network.

3.2.2.3. Fundamental Neural Network Architectures

An artificial neural network architecture is the overall structure of the network. Most neural networks are usually organized into groups of units called ***layers***. Neurons of the same layer mainly share the same pattern of connections with other neurons. The network can be either ***fully-connected*** if every node in each layer has a connection with every node in the adjacent forward layer, or ***partially-connected***, when the network is missing some connections [118]. And, according to the number of composed layers, neural network architectures can be classified into two major classes as follows.

1. Single-Layer Feedforward Neural Network

The single-layer feedforward network (Figure 3.2 (a)) is the simplest form of a layered network. It has one layer of connection weights. “Single-layer” is related to the presence of only one layer of computational nodes, which are the output neurons [118]. The nodes of the input layer receive the signals, and the nodes of the output layer compute and transform the input information. The term “feedforward” refers to the direction of the information flow from the input layer to the output layer (and not the opposite direction).

2. Multi-Layer Feedforward Neural Network

The multi-layer feedforward network (Figure 3.2 (b)) is composed of several layers which are introduced in the network in order to express more complicated transformations. It contains one or more hidden layers between the input layer and the output layer. This introduction of hidden layers enables the network to extract latent factors of variations from its input [118]. In addition, the network acquires the ability to capture an overall perspective in spite of its local connectivity, owing to the fact of the increased neural connections and the additional set of synaptic connections [120].

In this architecture, the input signal is fed to the first layer (input layer). Consequently, the computational nodes of the second layer (next layer which is the first hidden layer) put in the activation pattern on the input layer nodes. Then, the signals outputted from the second layer (first hidden layer) are used as inputs to the third one (second hidden layer), and so on for the rest layers of the network. The nodes of a layer receive information only from the nodes of the preceding layer. The output signals of the last (output) layer constitute the final output of the network in response to the first (input) layer.

Figure 3.2 illustrates two typical examples of the neural network architecture types and the difference between them. The left side presents an example of a single-layer feedforward neural network where the input and output layers are composed of five and two nodes respectively. The right side presents an example of a multi-layer feedforward neural network where the input, hidden and output layers are composed of five, three and two nodes respectively. Both examples depict fully-connected networks.

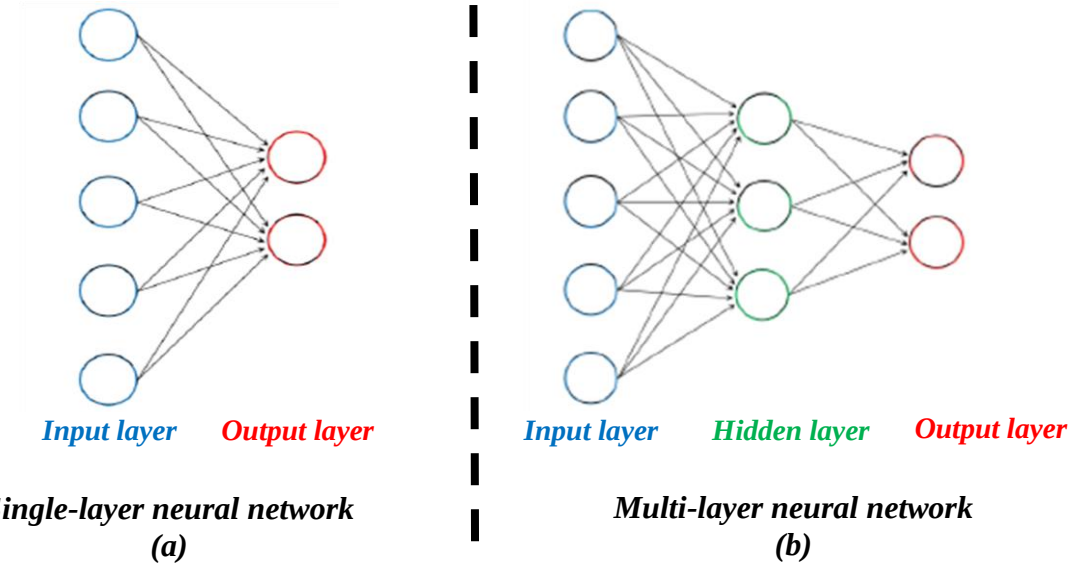

Figure 3.2. Single-layer feedforward neural network (a) Versus Multi-layer feedforward neural network (b).

3.2.2.4. Activation Functions

The fundamental operation of an artificial neuron implies summing up its weighted input signals and then applying an activation function. Typically, the same activation function is applied for all neurons of the same layer in the network. Activation functions must be differential and continuous so as to allow the error correction during the training phase [118]. There are many possible activation functions.

The activation function is a function that must return a real close to 1 when the "good" input information is given and a real close to 0 when it is "bad". Functions with values in the real interval [0,1] are generally used.

If the activation function is linear, the neural network would reduce to a simple linear function. However, nonlinear activation functions are often employed for more realistic results. These nonlinear transformations may significantly help when the input data are not linearly separable in the input space. They provide a new representation space in which the transformed data may be linearly separated. Therefore, the use of the neural network is however much more interesting when using nonlinear activation functions.

3.2.2.5. Learning Methods

As presented above, most problems require neural networks with an architecture of several fully-connected nonlinear layers. The neural networks will then be hard to interpret. For this reason, that deep learning is generally related to the term “black box”, as it is not possible to track the internal computations inside the network. Moreover, the increase in the number of hidden layers makes the learning process more complex and computationally expensive. In this section, we describe the different algorithms used for training neural networks. Then, we outline the most popular optimization algorithms and objective functions commonly adopted for training neural network models and solving the afore-mentioned issues.

3.2.2.5.1. Training Algorithms

The high commonly used method for training artificial neural networks is the **back-propagation** method. It consists of two phases of different directions:

- A ***forward phase***, where the signal x is propagated through the neural network layers from the input layer to the output layer. In this phase, weights of connections between neurons are fixed. The output value computed by the output layer is compared with the desired value using a loss function. The difference score is then used as error signal in the backward phase.
- A ***backward phase***, where the computed error signal is propagated through the network in the opposite direction layer-by-layer from the output layer to the input layer. The weights of connections between neurons are then adjusted so as to minimize the error value, i.e., so that the computed output and the desired output would be more similar. These computations of adjustments are much more challenging in the hidden layers.

3.2.2.5.2. Optimization algorithms

Training neural networks is such a complex task. Training a single instance may take time from days to months on hundreds of machines. As a result, several optimization methods have been proposed so as to solve these issues of increased complexity and high computational costs. In what follows, we outline the most efficient modern optimization algorithms used while training deep neural networks. They are adapted from [118].

- **Gradient descent** is the basic optimization algorithm for training artificial neural networks. It updates the model parameters at each step of the iterative process in the gradient direction of the objective function.
- **Stochastic gradient descent (SGD)** [121] is a stochastic approximation of the gradient descent optimization. It also follows an iterative process to minimize the objective function.
- **Stochastic gradient descent with Momentum**. The momentum method [122] can provide considerable improvements over SGD, speeding up the training process and avoiding the unstable oscillatory problems caused by selecting high values of learning rates.
- **Stochastic gradient descent with Nesterov Momentum** is proposed in 2013 [123] to improve the momentum method, and inspired from the Nesterov's accelerated gradient method [124]. The Nesterov Momentum method evaluates and continuously corrects the gradient computation, and prevents the model from increasing the responsiveness.
- **AdaGrad** [125] updates learning rates by scaling them inversely proportional to the square root of summing all the historical squared values of the past gradients. Differently from the previous optimization methods, this algorithm considers a different learning rate value for each step.
- **RMSProp**. The main weakness of Adagrad is that, it may lead to an algorithm unable to acquire additional knowledge [126] due to the fact that the squared gradients accumulate and keep growing up during the training phase. RMSProp method is proposed [127] in order to solve this issue. It consists on discarding the extreme historical gradient values and accumulating only near past gradients.
- **Adam** [128] computes adaptive learning rates for each parameter. It prevents the model from high biases and results as a robust response to the selection of hyperparameters.

- **Adadelta** is proposed in 2012 [129] as an extension of Adagrad in order to overcome many issues such as the problem of continual decay of learning rates during the training phase. It is robust to a large variety of configuration choices, demonstrating a high disposition to be applied as optimization method for training neural networks.

3.2.2.5.3. Objective functions

Most deep learning models involve an optimization phase, that usually consists on minimizing the objective function named as the *loss function* [118]. The objective function typically returns the distance between the output computed by the neural network and the expected output of the training data. The optimization process aims for identifying the best set of parameters which minimizes this distance.

Regularization refers for taking as input the parameters of a neural network and returns as output a score representing their complexity. Thus, the training algorithm aims for minimizing both the loss function and the parameters complexity.

Various loss functions and regularization techniques have been proposed and the possibilities of their combination lead to diverse learning algorithms. In what follows, we introduce the most commonly used loss functions and regularization methods used for training artificial neural networks (adapted from [118]).

1) Loss functions

- **Mean Squared Error (MSE)** squares the prediction error and averages over all instances of the training set.
- **Hinge loss** is the loss function ordinarily adopted for binary classification purposes, and can be extended for dealing with multi-classification problems [130].
- **Log loss** is a continuous loss function which displays a related convergence rate as the hinge function.
- **Binary cross-entropy loss** [131] is the appropriate loss function for binary classification purposes with conditional probability outputs.

2) Regularization

- **L2 regularization** [132] is one of the simplest regularization techniques. It computes the sum of squares of the network weights.
- **L1 regularization** [133] is the sum of absolute values of the network weights. It is recommended in particular for sparse solutions.
- **Elastic net** [134] is the combination of L2 and L1 regularizers.

- **Dropout** [135] randomly drops units from the neural network during the training phase. It is less complex compared to the previously presented regularization techniques, and generally leads to significant results.

3.2.3. Deep Learning Architectures

A good number and varied architectures are used in deep learning, such as convolutional neural networks, recurrent neural networks, long-short term memory and others. This thesis focuses on the deep learning research lines that we present in the following.

3.2.3.1. Auto-Encoders

Auto-encoders are the most adopted deep learning models for unsupervised representation learning [118]. They act as a dimensional reduction method, where the input layer is copied to the output layer and the hidden layer between them represents then the latent factor of the data. This representation has proved its efficiency in facilitating the visualization, communication, classification and storage of data [136]. Thus, auto-encoders have achieved successful results in a variety of applications, and attracted a lot of attention in recent years as very effective unsupervised models.

The core component of an auto-encoder is a neural network which tries to reconstruct its input layer at its output layer. Figure 3.3 illustrates the general architecture of auto-encoder, with three hidden layers of three, two and three nodes respectively.

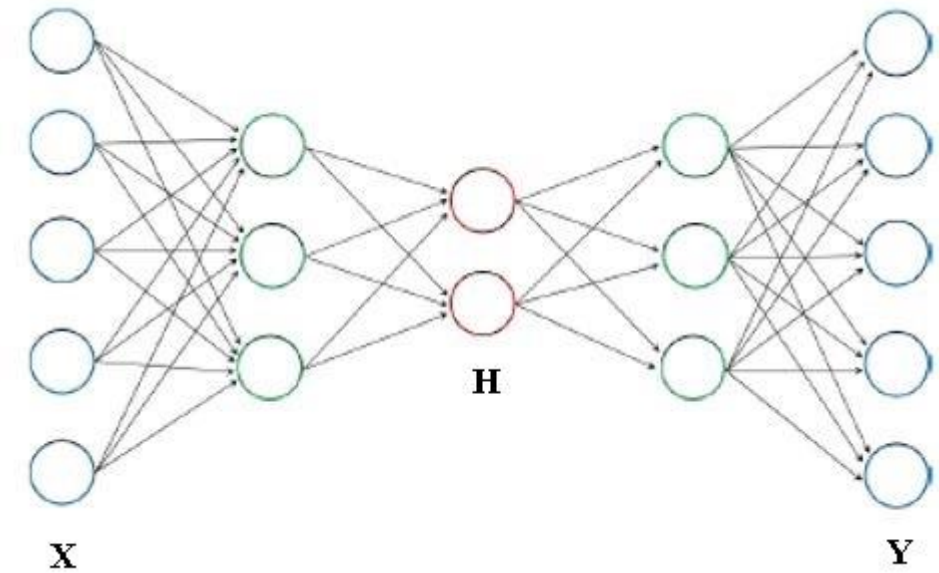

Figure 3.3. General architecture of auto-encoder.

An Auto-encoder consists initially of two crucial components.

1. The **encoder** function, denoted as f, allows an efficient and straightforward feature extraction from an input set of data $X = \{x_1, x_2, \dots, x_n\}$, and represent it as a feature vector $H = \{h_1, h_2, \dots, h_m\}$. The encoder function can then be defined as:

$$H = f(X) \tag{3.2}$$

2. The **decoder** function, denoted as g, maps the feature space back into the input space, producing a reconstruction set $Y = \{y_1, y_2, \dots, y_n\}$. The decoder function can be defined as:

$$Y = g(H) \tag{3.3}$$

Formally, an auto-encoder can be expressed as a multi-layer artificial neural network as:

$$\begin{cases} H = f(X) = a_f(WX + b_f) = a_f(\sum_{i=1}^{n}(w_i x_i) + b_f) \\ Y = g(H) = a_g(W'H + b_g) = a_g(\sum_{j=1}^{m}(w'_j h_j) + b_g) \end{cases} \tag{3.4}$$

where: a_f and a_g are the encoder and decoder activation functions; b_f and b_g are the encoder and decoder bias vectors; *W* and *W'* are the encoder and decoder weight matrices.

The auto-encoder training process consists on finding the set of parameters which minimizes the reconstruction error. Stochastic gradient descent methods are usually employed for error minimization while training auto-encoders [118]. The choice of the activation and optimization functions largely depends on the domain nature of the input data.

3.2.3.2. Embedding Models

The unsupervised generation of embeddings are one of the recent successes of the artificial neural network models. The term '*embedding'* is usually used in machine learning to refer for representing objects in a real number vector space.

Embeddings allows performing complex analysis tasks on new types of data since machine learning models work essentially on numerical data. Moreover, they retain characteristics of object and then reduce complex models to fewer dimensions.

Embedding algorithms rely on the notion of "neighbourhood" [137]. They work in a way that the geometric relationship between two vectors represents the semantic relation

between the corresponding entities. In what follows, we outline some basic models for embedding data into low-dimensional vector spaces:

- **Word Embeddings**

  The word can be seen as the atomic unit of natural language processing. Treating words as vocabularies suffers from sparsity and high-dimensionality. Word embedding works on finding new representations of words, which are dense, lower-dimensional and easily manageable by machine learning models. These algorithms are neural network based models trained on a large text corpus, and produce as output a vector space, aiming for representing each word in the corpus by a real valued vector.
- **Node Embeddings**

  Node embedding consists on mapping nodes to a high dimensional vector space so as to maximize the likelihood of preserving node neighbourhoods [137].
- **Sentence Embeddings**

  Words can be combined in exceedingly many ways. Unlike word embeddings where words represent semantic units, the idea of semantic embeddings is to consider words as a continuous representation in a sentence.
- **Knowledge Graph Embeddings**

  Embedding models have also spilled into the field of knowledge graphs. They are mainly used in *statistical representation learning*, where graphs are compressed into low-dimensional representations which may be used by reasoning systems, and in *knowledge base completion*, where embeddings are used to predict new relations between graph's entities [138].

3.3. Review of the Literature

In this section, we review the use of deep learning models in the ontology matching field. First, we present the ontology matching tools based on artificial neural networks proposed in the literature. Then, we present the existing approaches that have made use of deep neural networks for ontology matching.

3.3.1. Ontology Matching with Artificial Neural Networks

In the last decades, using machine learning techniques, particularly neural networks, to match heterogeneous ontologies has attracted much attention from research teams.

Following, we present the different approaches which make use of artificial neural networks for ontology matching with a chronological order.

- The first application of neural networks in general mapping were in 1989 [139], where the authors proved that any continuous mapping can be approximately realized by multilayer neural networks with at least one hidden layer with sigmoid output functions.
- Authors in [140], proposed an efficient learning method to approximate non-linear mappings and their derivatives, whose input-output relations are represented by neural networks.
- SEMINT (SEMantic INTegrator) [141] [142] is a system prototype for semantic integration in heterogeneous databases using neural networks. In 1993 [143], authors presented three techniques for automating the process of matching to integrate heterogeneous database systems. The study in [144] presents a procedure using a classifier to categorize attributes according to their field specifications and data values, and then trains a neural network to recognize similar attributes. In 2000 [145], authors represented attributes in different databases with their metadata as discriminators.
- The work in [146] presents an integrated ontology mapping approach. It determines similarity through rules which have been manually formulated by ontology experts.
- APFEL (Alignment Process Feature Estimation and Learning) [147] is a machine learning approach that explores the user validation of initial alignments for optimizing alignment methods, which are based on extensional and intentional ontology definitions.
- The study in [148] presents an automatic ontology alignment method based on the recursive neural network model that uses ontology instances to learn similarities between ontology concepts.
- In work [149], a new supervised learning based method for compound metric creation is proposed. A training set is used to create a neural network model, performs sensitivity analysis on it to select appropriate metrics among a set of existing ones, and finally constructs a neural network model to combine the result metrics into a compound one.

- The work in [150] presents a Knowledge Source Discovery (KSD) agent, which guides knowledge requirements towards distributed ontology domains in the Semantic Web through a neural network model.
- OAANN (Ontology Alignment by Artificial Neural Networks) [151], [152] uses artificial neural network to align biological ontologies. It consists of learning and adjusting contributing weights for the different semantic aspects of ontologies.
- MALFOM-SVM [153] uses multiple concept similarity measures for the ontology mapping problem. It organized this problem into a standard machine learning framework.
- X-SOM is a flexible and extensible ontology mapping and integration tool first presented in 2007 [154]. It combines various matching algorithms by means of a feed-forward neural network. It exploits logical reasoning and local heuristics to improve the quality of mappings while guaranteeing their consistency. The architecture of the X-SOM Ontology Mapper is composed by three subsystems: Matching subsystem, Mapping subsystem and Inconsistency Resolution subsystem. The work in [155] summarizes its results in the OAEI 2007 campaign. A nested and double classification approach for missing value imputation are presented in [156] and [157].
- OMNN (Ontology Mapping Neural Network) [158] [159] is proposed in order to learn and infer correspondences among ontologies. It extends the Identical Elements Neural Network's ability to represent and map complex relationships among ontologies. The learning dynamics of simultaneous training of similar tasks interact at the shared connections of the networks. The output of one network in response to a stimulus to another network can be interpreted as an analogical mapping. OMNN has proved its performance on ontology mapping by participating to several OAEI benchmark test cases.
- PRIOR+ (Profile pRopagation and InfOrmation Retrieval techniques) [160] [161] is a generic and adaptive ontology mapping approach, based on propagation theory, information retrieval techniques and artificial intelligence. The approach consists of three major modules, the IR-based similarity generator, the adaptive similarity filter and weighted similarity aggregator, and the neural network based constraint satisfaction solver. PRIOR+ first measures both linguistic and structural similarity of

ontologies in a vector space model using classic information retrieval techniques, and aggregates them using an adaptive method based on their harmonies. Then, the interactive activation and competition neural network is selectively activated to solve the constraint satisfaction problem in the context of ontology mapping. The work in [162] and [163] summarizes the results of PRIOR and PRIOR+ for OAEI 2006 and 2007 campaigns. Authors used the interactive activation network [164], then integrated the interactive activation and competition neural network in ontology mapping [165]. In 2008 [166], they treated the neural network based constraint satisfaction in ontology mapping. The work in [167] presents a harmony based adaptive ontology mapping approach.

- MoTo (Mapping ontology To ontology) [168] [169] is an automated ontology matching system for recovering uncertain mappings through structural validation and aggregation, supported by various machine learning techniques.
- The work in [170] presents an Artificial Neural Network based ontology matching model for improving web knowledge resource discovery on the Semantic Web based on recently developed intelligent techniques. This method takes into account both schema-level and instance-level information from ontologies, and semantic annotations, and combines agent-based technologies with an artificial neural network based classifier to propose a solution to the ontology-matching problem.
- CIDER (Context and Inference baseD alignER) is a schema-based ontology alignment algorithm with usage of neural networks. It compares each pair of ontology terms by extracting their ontological contexts and combining different elementary ontology matching techniques. Its participations at the OAEI campaigns are presented in [171], [172] and [173]. CIDER first extracts the ontological contexts for each ontology terms pair up to a certain depth and enriches it by applying lightweight inference rules, and then combines the different elementary ontology matching techniques using artificial neural networks in order to generate alignments between ontologies. CIDER-CL is the evolution of CIDER for Cross-Lingual matching.
- In 2012 [174], authors presented an ontology mapping system which computes various types of similarities based on metadata and instances and combines them using neural network learning.

- X-Map (eXtended Mapping) is a structural approach for aligning OWL ontologies first defined in 2010 [175], presenting an automatic method to learn how to combine the linguistic and structural affinity. XMap++ [176] [177] exploits WordNet as a background knowledge sources. In 2012 [178] [179], authors introduced artificial neural network in the ontology alignment process to combine multiple similarity measures into a single aggregated metric. In 2013 [180], they aimed for improving the large-scale ontology alignment quality. XMapGen and XMapSig [181] are two variants of XMap++. Authors in [182] proposed a novel approach using context-based measure for matching large-scale ontologies. XMap++ and XMap took part in several editions of OAEI where their results and performance are described [183] [184] [185] [186] [57].
- The approach proposed in [187] tackles the ontology alignment task by proposing a matching process based on the usage of Weightless Neural Network (WNN) model. A WiSARD classifier is built and used to estimate a distribution-based similarity measure among the concepts of the several ontologies being matched. New patterns can be learned without the need to retrain the complete neural network, and names of classes are taken into account in order to obtain a more significant alignment.

3.3.2. Deep Learning Solutions for Ontology Matching Tasks

In this section, we study the use of deep learning techniques in the ontology matching field. The related work include:

- ERSOM (Entity Representation and Structure based Ontology Matching) is proposed in 2015 [188]. It is an ontology matching system which mainly uses the deep neural network model to learn the high-level abstract representations of classes and properties from their descriptions. And it uses an iterative similarity propagation method based on more abundant structure information of the ontology for ontology matching in an unsupervised way. In 2017 [189], authors added a supervised learning step when training data is available to refine the learned representation, and then allowed to learn the representation of ontology entity in the cases the training data exists or not.
- The study of Nkisi-Orji et al. [190] introduced a random forest classifier for ontology alignment which integrates semantic similarity features, string-based similarity features and semantic context features, using word embedding. It completes

alignment in two stages. It first selects a set of candidate alignments using basic matching techniques. After that, a machine classifier determines the true alignments from entity pairs of the candidate alignments, using feature vectors that are generated from a variety of direct and indirect similarity indicators.

- The approach proposed by Chandrashekar et al. [191] aims to discover the relationships between concepts from the analysis of semantic features across multiple ontologies, and identify the abstractions of the ontological relationships through mapping between features to the ontologies. The ontology mapping is performed through ontology search, feature extraction and word embeddings.
- The work in [192] presented a novel ontology mapping system called HISDOM, which uses comprehensive factors like concept names, attributes, instances, and structural similarities to determine the similarity of ontology. And then dynamically derives the weight of those different factors in the overall ontology similarity proportional to the amount of information of each factor in the ontology, to determine whether the two ontologies have mapping relationships. HISDOM also uses a convolutional neural network to extract and calculate the comment and annotation semantics and find their similarity according to the extent of annotation.
- Dhouib et al. [193] proposed a new ontology alignment approach inspired by an existing proposal [194]. It combines the radius measure and word embedding. They consider word embedding to get a vector representation of the concepts to be matched, and use it to compute hierarchical relations between concepts.

3.3.3. Analytical Summary

Deep learning algorithms have motivated numerous researchers in many fields to employ them in order to solve different and complex problems. Previously, we reviewed a state-of-the-art on ontology matching approaches and systems that make use of artificial neural networks in general, then of deep neural networks in particular. In this section, we study and discuss these matching tools in order to identify their advantages and limits, and outline contributions of this study.

A comparative review of existing ontology matching tools based on neural networks is given in Table 3.1. They are classified chronologically according to their last evolution year where neural networks are employed. ***Matching Strategy*** column specifies the technique adopted for finding equivalences between ontologies. ***Neural Network Usage***

columns describe the use of these networks in the matching process by the reviewed approach, including the ***purpose*** of this application, the ***structure*** of the network, as well as the ***learning method***. ***Large-Scale*** column considers whether the matching tool can tackle the issue of large-scale ontology matching or not.

This chapter is partly related to our published paper [195], that aims to figure out the best way to use neural networks in ontology matching. It provides a survey on the different ontology matching approaches based on neural networks, seeking for clearing the way for researchers in this domain. Readers are referred to [195] for further studies and for more details about the input required by each tool, its output type, matching interactivity, the results of its evaluation in terms of precision, recall and F-measure, participation at OAEI campaigns, …etc.

Matcher	**Matching Strategy**	**Neural Network Usage**			**Large-Scale**
		Purpose	**Structure**	**Learning Method**	
– 1989 [139]	Neural networks	Approximation	Multilayer feed-forward neural network Sigmoid as output activation function	Backpropagation learning	×
1993 [140]	Non linear differentiable mapping; Neural networks	Approximation	Multi-layer feedforward neural network + its adjoints Non linear activation function	Supervised learning Backpropagation method with the steepest descent algorithm	×
Semint 2000 [145]	DBMS parsers; Clustering, Self-organizing map algorithm (unsupervised learning); Neural networks	Category learning and recognition	3 layer neural network	Supervised learning, forward propagation, error calculation, backward propagation	×
– 2004 [146]	Terminological similarity; Intentional similarity; Heuristics; Machine learning techniques	Learn weights of n different similarity methods classify mappings into: equal or not equal	3 layer neural network consisting of 1 linear input layer, 1 hidden layer with a *tanh* function, and a *sigmoid* output function	-	×

APFEL 2005 [147]	Terminological; Extensional; Intentional; Heuristics; Machine learning techniques	Optimize the representation of the alignment scheme Aggregate different similarity measures using weighting schemes classification	**Input:** set of feature/similarity combinations generated before training **Output:** classification of being aligned or not **Training data:** validated alignment pairs processed with the automatically generated collection of features and similarities	Supervised learning	×
– 2005 [148]	Terminological similarity; Structural similarity, graph-based; Extensional similarity; Neural networks	Estimate distribution-based similarity measure	Recursive 2 layer neural network classifier Sigmoid as output activation function	Supervised learning, backpropagation algorithm	×
– 2006 [149]	Different similarity metrics; Machine learning techniques	Approximation Select appropriate metrics for combination, and select their appropriate weights	Multilayer neural network	Supervised learning	×
– 2007 [150]	Terminological similarity, string-based; Extensional similarity, instance-based; Neural networks	exploit the information contained in ontology instances classify the instances of each concept: positive or negative matching	3 layer multilayer perceptron model (1 hidden layer) Linear activation functions for hidden layer neurons	Supervised learning: standard backpropagation algorithm	×
OAANN 2008 [152]	Rule-based + learning-based; Terminological similarity; Intentional similarity; Extensional similarity; Neural networks	Learn weights for concept's semantic aspects	2 layer neural network (with 3 input neurons and 1 output neuron)	Supervised learning, Gradient-descent	×
MALFOM-SVM 2008 [153]	Word similarity, string-based (prefix, suffix, Edit distance, n-gram), knowledge-based (Wordnet, synset, Wu & palmer, description, Lin); Word list similarity, maximum word similarity, word edit distance; Concept hierarchy similarity; Structure similarity	Classification	-	Supervised machine learning	×

X-SOM 2010 [157]	Logical reasoning; Local heuristics; Consistency checking; Language-based similarity, Jaro, Levenshtein, Wordnet, Leacock Chodorow; Structural similarity, graph-based; Semantic similarity; Neural networks	Aggregate several similarity maps	Feed-forward neural network	Supervised learning, standard backpropagation learning algorithm	×
OMNN 2010 [159]	Identical Elements Neural Network	learn relationship mapping and infer correspondences among ontologies	4 Sub multilayer neural networks	Cross training	×
PRIOR 2010 [161]	Linguistic similarity, Edit distance, IR techniques, TF-IDF, Cosine; Structural similarity; Neural networks;	Solve Constraint Satisfaction Problem (CSP) in context of OM	Interactive Activation and Competition (IAC) neural network	Propagation theory	√
MoTo 2011 [169]	Linguistic similarity, Wordnet, relational affinity, linguistic quantifiers; Structural similarity, IC, Jaccard, Dice, Ochiai, Gower Legendre Extensional using Pellet reasoner; Machine learning techniques (4 base learners+1 meta learner to produce 1 K-Nearest Neighbor Classifier+1 ANN+2 Bayesian Classifiers)	Classify instances and estimate their probability distributions	-	Unsupervised learning	×
– 2012 [170]	Schema-level + instance-level; Semantic annotations; Agent-based technologies; Machine learning classifier	Classification	Multilayer artificial neural network Sigmoid as activation function	Supervised learning, Levenberg-Marquardt training algorithm	×

– 2013 [174]	Terminological similarity, Levenshteins distance, TF-IDF; Structural similarity; Extensional, instance-based, attributes, Dice; Neural networks	Learn the different weights for instance - based and metadata measures	3 layers feed forward neural network (8 units in the input layer, 12 units in the hidden layer and one output unit) Weights initialized randomly Training set prepared manually		×
CIDER 2013 [173]	Context extraction, inference rules, semantic reasoner; Terminological similarity, Levenshtein; Structural similarity, VSM; Neural networks	Combine features of extracted ontological contexts	2 3-layer perceptrons (each of 5 input neurons, 3 neurons in hidden layer, 1 output neuron and 2 bias) Sigmoid as activation function	Supervised learning	×
ERSOM 2017 [189]	Terminological, Cosine Similarity; Structural, Similarity Propagation, Intentional, Induced propagation graph; Extensional, Scaled Levenstein; Kullback-Leibler divergence; Neural networks	Learn the high level abstract representation for ontology entities	Deep neural network: Multi layer learning model; Auto-encoder (1 hidden layer with large number of units); Stacked Auto-Encoder (Multiple hidden layers with large number of units) Softmax regression classifier Training data by domain experts	Supervised + Unsupervised representation learning	×
- 2017 [187]	Weightless neural networks; A WiSARD classifier is built to estimate a distribution-based similarity measure among the concepts of ontologies	Classification	WiSARD neural networks	Supervised learning	×
XMap 2018 [57]	Terminological similarity, string-based, language-based Structural similarity; Schema-based; Semantic similarity, Wordnet Context-based; Neural networks	Extract the optimal model of compound metrics	Feedforward neural network: Multi layer perceptron (3 layers: input, hidden, and output layer (with 3, 4, and 1 neurons respectively)+2 bias for input and hidden layers) Sigmoid as activation function	Supervised machine learning: Resilient Propagation Training	√
- 2018 [190]	Random forest classifier; Word embedding; String-based similarity; Semantic similarity	Classification	Deep neural networks	Supervised machine learning	×

- 2018 [191]	Semantic mapping; Ontology search; Feature extraction; Word embeddings; Natural Language Processing, TF-IDF	Learning word embeddings & producing vector space for a large corpus of ontological properties	Skip-gram model of Word2Vec (two-layer neural network)	Unsupervised learning	×
HISDOM 2019 [192]	Multi-dimensional similarity, name-based, attribute-based, instance-based, structure-based, comments-based; Hybrid similarity based on dynamic weights	Extracting comment and annotation semantics and calculating their similarity	Convolutional neural networks	-	×
- 2019 [193]	Word embedding; Radius measure	Representing concepts of ontologies and computing equivalence and hierarchical relations between concepts	Pre-trained word vectors	Unsupervised learning	×

Table 3.1. Summary of ontology matching tools based on neural networks.

Figure 3.4 illustrates the publication activity and the evolution of the previously described researches on ontology matching using artificial neural networks.

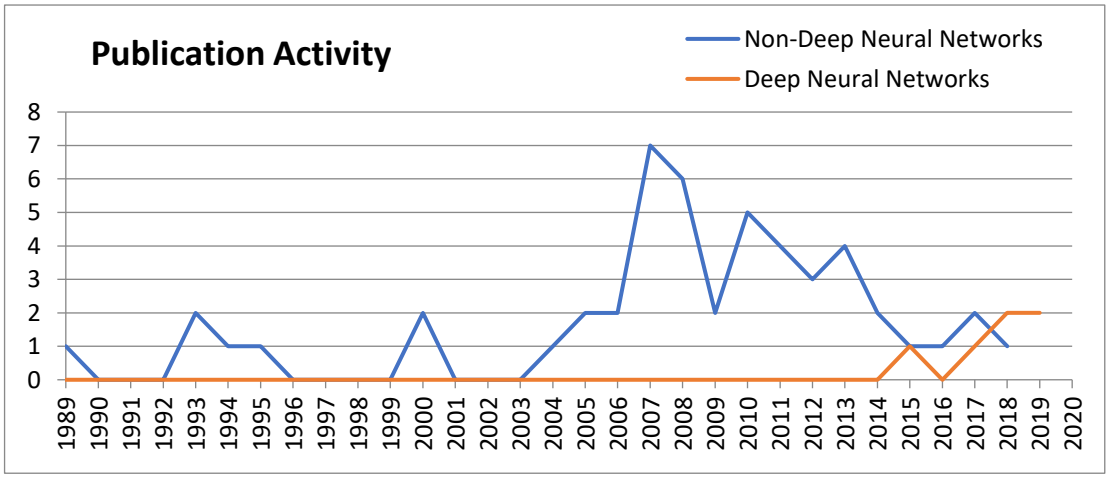

Figure 3.4. Publication activity on using artificial neural networks for ontology matching.

An analytical look at the above comparative review reveals the unfolded conclusions:

- The use of artificial neural networks for ontology matching started from 1989, reached its maximum value which is equal to 7 published works in 2007, and continues to 2018. Using deep learning models for matching heterogeneous ontologies has not attracted much attention from researchers. It started in 2015 and continues to this day with a low publication rate.

- The study of the researches presented previously highlights the fact that the main difference between them resides in the strategy used in matching the heterogeneous ontologies, like general ontology matching approaches, and particularly in the purpose of applying neural networks in ontology matching.
- The most commonly used learning technique is the back-propagation learning method, and a very high percentage of them favour supervised learning.
- The structure is generally a multilayer feed-forward neural network.
- Only Prior and XMap tackle the issue of large-scale ontology matching.
- Artificial neural networks have been widely used in different areas of research, particularly in the field of ontology matching. However, the use of deep neural networks has not attracted much attention from research teams to match heterogeneous ontologies, despite the fact that ontology matching is an active field of current research.
- Although deep learning models are very appropriate for dealing with large datasets, they are not commonly used to address the problem of large-scale ontology matching. Moreover, the few works employing these models that can be found in the literature aim at enhancing the performance of the ontology matching task, and not at handling the heterogeneity between large ontologies. Besides, they tested their methods on ontologies of small sizes.

Further conclusions can be additionally derived from [195]:

- Regarding matching strategies, the most similarity type used is terminological measures, due to their ease of implementing. Structural measures are also used compared to extensional ones. Usually, semantic measures are not too much used because of their need of complex processes. However, they intervene in a large number of approaches in this literature review because of the fact that, those involving semantic aspects only by using artificial neural networks are included.
- Mainly, there are two major purposes of employing neural networks in ontology matching. The most used one is approximation, where the approaches use artificial neural networks mostly to find optimal weights and define functions that provide ontology similarity between entities of ontologies being matched, while others aim for learning representation of ontology components. Classification is the second one, where these machine learning models are employed to classify concepts of ontologies

in some researches, and to classify ontologies patterns in others. Some other ontology matching techniques applied these machine learning models for particular purposes.

- The big majority of the approaches take as input OWL ontologies and produce 1:1 output mapping result.
- About 85 percent of the approaches are fully automatics. Excepting SEMINT which intervene the user to check and confirm output results, and APFEL to validate initial alignments.
- The analysis of their matching results of the different approaches presented in their papers in terms of precision, recall and f-measure, shows a high accuracy of matching. This proves the efficiency of artificial neural networks in the field of ontology matching.

3.4. Conclusion

In this chapter, we have presented a state-of-the-art on matching ontologies with deep learning. First, we introduce the challenges motivating deep learning. Next, we present artificial neural networks, their biological inspiration, their fundamental architectures and activation functions. The learning methods are also detailly described. Then, we present deep learning architectures along with the most important deep learning models related to this study which are auto-encoders and embedding models. After that, we review the existing ontology matching techniques that have made use of artificial neural networks, and then particularly of deep neural networks. And finally, we discuss the presented tools so as to figure out the current challenges that allow to outline contributions of this work.

In chapter 1, chapter 2 and chapter 3, we have presented a wide overview of the state-of-the-art on the existing ontology matching tools in general, in the large-scale in particular and on those employing artificial and deep neural networks respectively. Apart from studying all these techniques and identifying their advantages and limits that we figure out the current challenges and address them by proposing our methodology for large-scale ontology matching using deep learning techniques which will be detailly described in the chapters of the next part.

Chapter 4

Reuse-based Semantic Approach for Large-Scale Ontology Matching

Contents

4.1. Introduction

Large-scale ontology matching is still challenging for its long-time processing and large-memory consumption. In the previous chapters, we have presented a large overview of the state-of the-art on the existing ontology matching tools, particularly on the large-scale ontology matching systems and those employing artificial and deep neural networks.

In this chapter, we present NeuralOM, an artificial neural networks-based solution that we propose to address the large-scale ontology matching challenges. We first overview the proposed solution and outline its contributions. Then, we detailly describe its technical steps. After that, we present the evaluation framework where we conduct our experiments on twelve different test cases from the OAEI initiative at both small and large scales. We present and analysis the results of these experiments and discuss the performance of NeuralOM.

4.2. Brief Overview

This chapter is related to our published work [196] [197] where we propose NeuralOM for large-scale ontology matching. Figure 4.1 presents an overview of the proposed matching approach. NeuralOM consists of combining the mappings of the most effective ontology matching systems through a linear perceptron in order to define a matching function that leads to generate the ideal set of correspondences between ontologies. We aim by training their neural network to adjust a weight for each matching tool according to its importance. The final mapping is obtained after a threshold filtering. This combination strategy using neural networks serves to increase the quality of the matching task, and then leads to have optimal matching results.

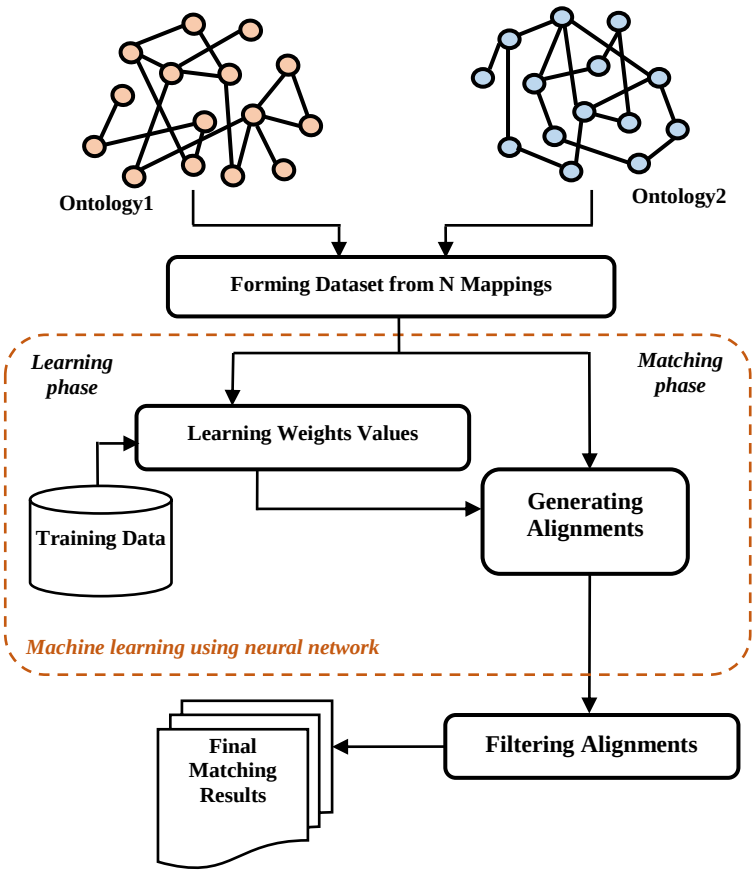

Figure 4.1. NeuralOM Overview.

The main contributions of NeuralOM are:

- It reuses and combines, according to a very detailed state of the art on the existing ontology matching techniques, the results of the best matching systems that have been validated through various test cases. As we aim, not just to generate alignments between ontologies, but to ideally match them, we refine these results to achieve a perfect ontology matching.

  A large number of works that address the ontology matching issue can be found in the literature. That allows thinking of benefiting from the existing matching techniques. Thus, refining and reusing different effective matching results give the impression of being interesting. Moreover, the process of matching ontologies (Sect.1.3.3) takes as parameter an initial alignment which is intended to be completed by the matching process. NeuralOM considers as initial alignment the different mappings generated by the selected candidate matching systems. Working on refining ontology matching results denotes working on a higher and more precise level than working on matching ontologies.

- Ontology matching is initially based on computing similarities between ontologies. Basing on one similarity type is not enough actually. The majority of ontology matching techniques combine several similarity measures. For preference, NeuralOM combines several alignments that have been generated through complicated processes and validated by various tests. That obviously can give finer and more precise matching results.
- The refining strategy is based on artificial neural networks which are very appropriate for combination, because of their structures of numerous inputs and outputs.
- Regarding the classification of ontology matching techniques (Sect.1.3.4), NeuralOM benefices from different matching strategies since the candidate matchers work differently. Therefore, it acquires a great chance to provide a maximal number of possible and especially of correct correspondences.

4.3. Neural Ontology Matching

In this section, we present ***NeuralOM***, the neural networks-based ontology matching approach that we propose for matching large-scale ontologies. The processing flow of NeuralOM is illustrated in Figure 4.2.

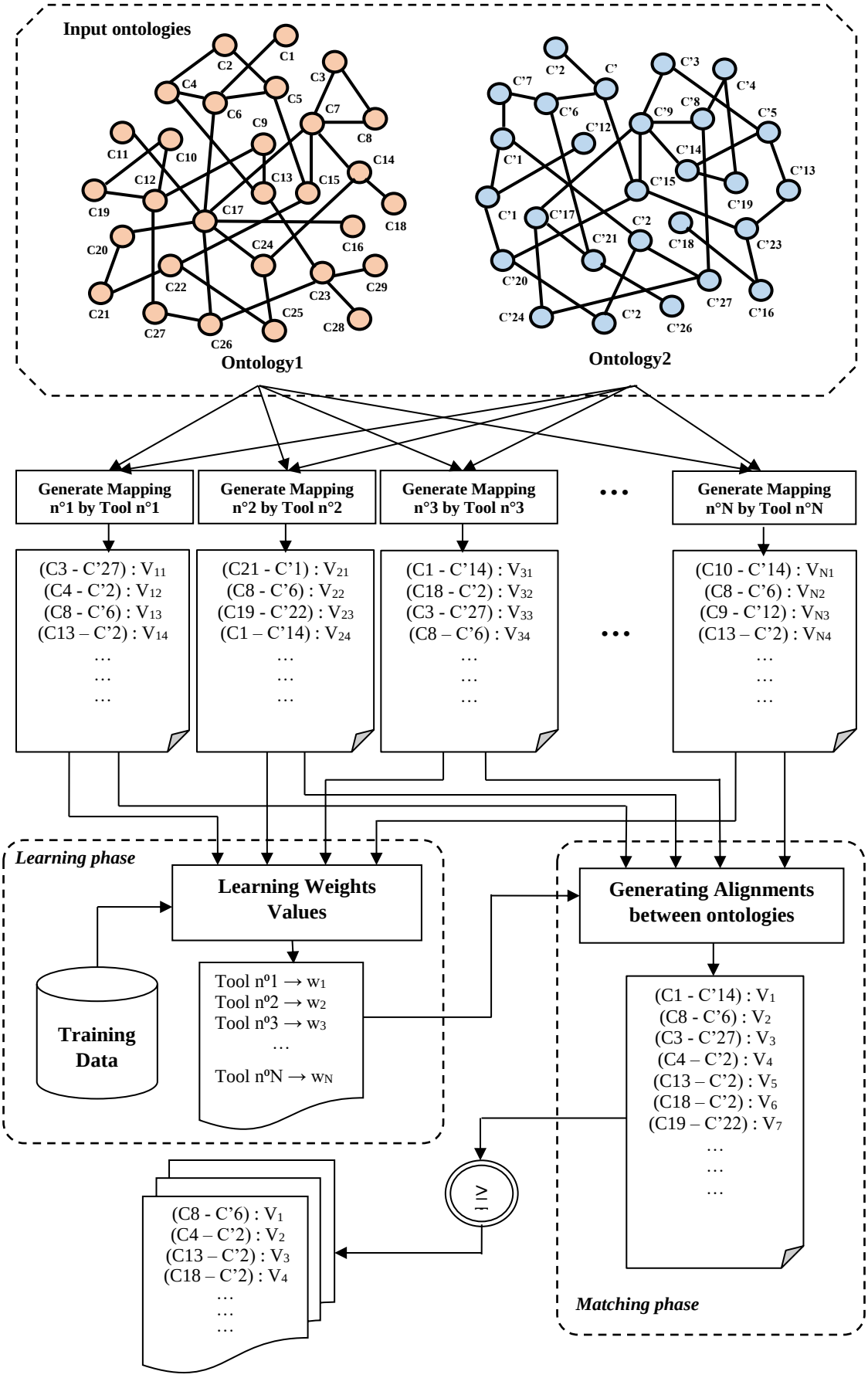

Figure 4.2. Processing flow of NeuralOM.

As we are dealing with an ontology matching issue, the input is two ontologies to be matched: *Ontology1* and *Ontology2*, and the output is an alignment (a set of correspondences) between them. We define such a correspondence by a triplet as:

$$A = \{C, C', V\} \tag{4.1}$$

Where: C is a concept from *Ontology1*; C' a concept from *Ontology2*; and V the similarity value between C and C' given by our technique.

4.3.1. Constructing the Dataset

The first step of NeuralOM consists on constructing the matching dataset by generating N alignments between the two input ontologies to be matched: *Ontology1* and *Ontology2*, by means of the most effective matching tools aiming to refine their results. Each set of correspondences is generated by an ontology matching tool applying its own specific matching technique. Combining such results is not that simple. Chosen systems, which are not all available, are of different inputs and especially of different outputs. Choice criteria and number of chosen systems are additional challenges. Therefore, these tools are chosen according to a very detailed state of the art on the different ontology matching systems developed in the scientific literature. N depends on choice criteria.

Next, the N alignments are combined and refined according to the environmental conditions of our approach for the aim of perfectioning the matching process. For example, an alignment value which exceeds the upper bound of the interval defined for correspondences should be set equal to 1.0.

After that, the N different mappings are combined to get the whole dataset for matching. The method of combination has a big impact on the resulted alignments. It determines the size of the dataset and then of the resulted mapping set. However, combining such results is quite challenging; Chosen tools, which are not all available, are of different inputs and especially of different outputs. Choice criteria and number of chosen systems are additional challenges.

4.3.2. Network Training

This step is the core process of the proposed matching approach. It consists of applying a supervised learning procedure based on neural networks, in order to learn the matching function that allows generating correspondences between *Ontology1* and *Ontology2*.

We aim by this step at adjusting a weight for each matching tool. For that, we train a neural network for each pair of concepts from the training dataset in order to fix, for each system, a value which reflects its importance.

The trained neural network consists of a simple linear perceptron of N inputs and one output. Figure 4.3 illustrates this network. Inputs correspond to the different matching systems whereas the output represents the pretended similarity value between the concerned concepts.

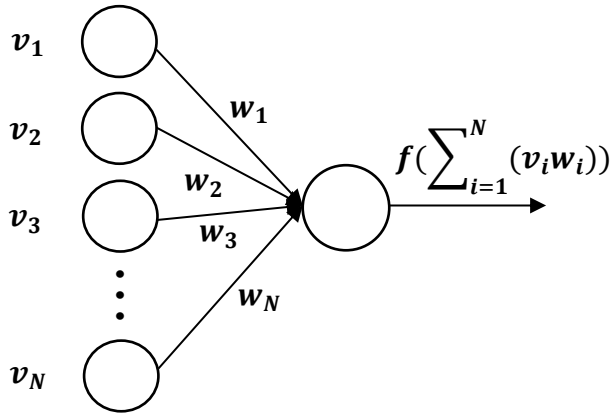

Figure 4.3. Neural Network Structure.

In neural networks, the output is built on the type of target variable. In our network, the output is in fact a similarity value, i.e., it should be a in the range [0,1]. Thus, we use Sigmoïd as activation function accordingly. Sigmoid is a mathematical function which is used excessively in neural networks. It is defined by:

$$Sig(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}} \quad ; \quad x \in [0,1] \tag{4.2}$$

For network learning, we use the back-propagation method following the learning algorithm described below. It uses a gradient decent procedure to modify weights so as to minimize the error between the desired output and the output computed by the perceptron.

As we aim by this network to fix an importance value for each tool, the N weights are first initialized according to the choice procedure of the matching tools. They are fixed according to a detailed analysis of our state of the art on the different matching systems of the scientific literature. Then, they are updated for each sample of the training dataset S, applying the gradient decent method aiming at minimizing the error between the desired output and the output computed by the perceptron.

Each sample of the training set comprises two parts: {input, output}; the first one is composed by N similarity values obtained from the precedent mappings for the two concepts in question, and the second one affords the reference alignment value between the two concepts. This value is obtained from OAEI plus an expertise touch. The learning rate ε is fixed by trial and test. As results of this step, the final N weights values are adjusted after that the execution of the learning algorithm is completed.

Algorithm 4.1. *Training the neural network to learn tools' weights*

Input: perceptron P of N inputs and 1 output defined by the weights vector $w^{®}=(w_1,...,w_n)$;
training set $S=(v^{®}, out)\in R^n\times\{0,1\}$;
learning rate ε;

Begin

initialization of weights w_i for i from 1 to n
initialization of ε
Repeat
take an example $(v^{®j}, out^j)$ from S
compute the output $o^j = f(v^{®j}, w^{®})$ of P for j
$error \leftarrow \varepsilon * (out^j - o^j)$
// update weights
for i from 1 to n
$w_i \leftarrow w_i + error * v_i$
End Repeat

End

Output: P defined by $w^{®}=(w_1,...,w_n)$.

4.3.3. Matching Ontologies

The supervised learning process performed in the precedent step basing on neural networks allows to learn the matching tool's weights, and then to define the matching function that leads to generate semantic correspondences between *Ontology1* and *Ontology2*, using the different mapping results of the first step. The output similarity value between each pair of concepts from the input ontologies is computed as:

$$V = \sum_{i=1}^{N} (v_i w_i) / \sum_{i=1}^{N} w_i . \qquad (4.3)$$

Where: $v_i \in [0,1]$ is the similarity value given by $tool_i$ between the two concepts; w_i is the weight which reflects its performance.

Finally, we filter the generated alignments so as to get the final mapping. For that, we define a threshold T (fixed by trial and test) which permits to extract the final alignment from the results obtained previously. Our aim behind this step at perfecting the matching process and ameliorating its precision, by eliminating irrelevant alignments, and keeping only the most appropriate ones.

4.4. Evaluation Framework

Aiming to study the effectiveness of NeuralOM, we evaluate it according to various campaigns of the Ontology Alignment Evaluation Initiative (OAEI). More details about this initiative and the evaluation challenges that it provides are given in Sect.6.4.1. We compare the results of NeuralOM, by its three different variants and for each test case, with the results of all OAEI participant systems for the same test challenge, and adopting the same cross-validation procedure.

This comparison is done for the global dataset in terms of the five standard evaluation metrics defined previously (Sect 1.3.5.3): *precision*, *recall* as well as three variants of F_b-*measure* ($F_{0.5}$-measure, F_1-measure and F_2-measure).

In the following, we refer by ***NeuralOM-I***, ***NeuralOM-M*** and ***NeuralOM-U*** to our neural networks-based matching approach by *intersection*, *majority* and *union* dataset construction method respectively (more details about these versions are provided in Sect.6.3.4).

In order to choose the most efficient systems to be used in the first step of NeuralOM, we based on F_1-measure, because it is the harmonic mean of precision and recall where both of them receive equal weight. We pick up the systems that maximize this score according to the results of OAEI for each test case. Their initial weights are those F_1-measure values, and their number is fixed by trial and test. These tools have marked their efficiency over years by participating in several editions of the OAEI since their realizations.

We adopt a cross-validation to effectively control the network while training and testing. More details about the cross-validation procedure are given in Sect.6.4.1. The number of cross-validation partitions, is fixed by trial and test. This latter is fixed to 2 partitions for both of Anatomy, Phenotype-HP-MP and LargeBioMed-FMA-NCI test cases, to 3 for LargeBioMed-SNOMED-NCI, to 5 for LargeBioMed-FMA-SNOMED and to 6 partitions for Phenotype-DOID-ORDO challenges.

We aim for performing tests on both small and large ontologies. Thus, we conducted our experiments in two main sketches: *small-scale evaluation* and *large-scale evaluation*. Moreover, as we really interest in matching complexity, particularly in time processing, we analyse the matching Runtime required by NeuralOM compared to all participant matching systems for each test case in the large-scale.

4.4.1. Small-Scale Evaluation

In this section, we present the results of our experimental procedure at the small scale.

4.4.1.1. Evaluation for Conference Track

Figure 4.4 illustrates the results of evaluating NeuralOM according to **OAEI'2018-CONFERENCE Track *(Conference'18)***, and summarises their comparison with OAEI matching systems.

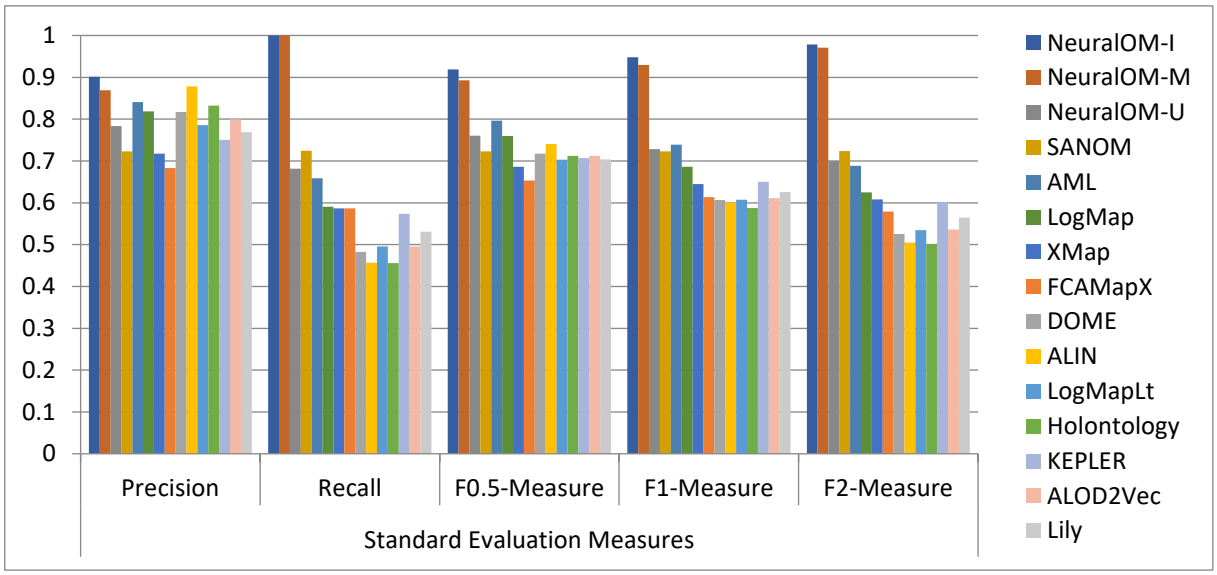

Figure 4.4. Evaluation results of NeuralOM against OAEI systems for Conference'18 track.

The results of the global dataset hail from those of the several partitions. As shown in Figure 4.4: 1. NeuralOM gives very good results (the minimum value is 0.681 given for recall by NeuralOM-U, which gives better good values for the four other metrics), especially Intersection and Majority versions that present complete values of recall (1.0) and excellent values of precision and the three F-measures. Thus, they present by far the best matching results. 2. Also for us, NeuralOM-I gets better performance than NeuralOM-M and better than NeuralOM-U. Contrary to the latter, the two other versions get high values of precision than of recall. 3. OAEI matching systems present almost a descending order of their recall

values and no order of their precision values, thus, their F-measures values are also ordered decreasingly but less sharply. 4. As expressed by F_1-measure, which better shows the real quality of the matching results, NeuralOM-I results are slightly better than those of NeuralOM-M, but they are roughly better than the others (the difference exceeds 0.2), including NeuralOM-U of which the results are close to those of AML and competitive to those of all the other matching systems.

4.4.1.2. Evaluation for Biodiversity and Ecology Track

Plots in Figure 4.5 and Figure 4.6 illustrate the results of evaluating NeuralOM according to **OAEI'2018- BIODIV Track *(BioDiv'18)*** for FLOPO-PTO and ENVO-SWEET Sub-Tracks respectively, and summarises their comparison with OAEI matching systems.

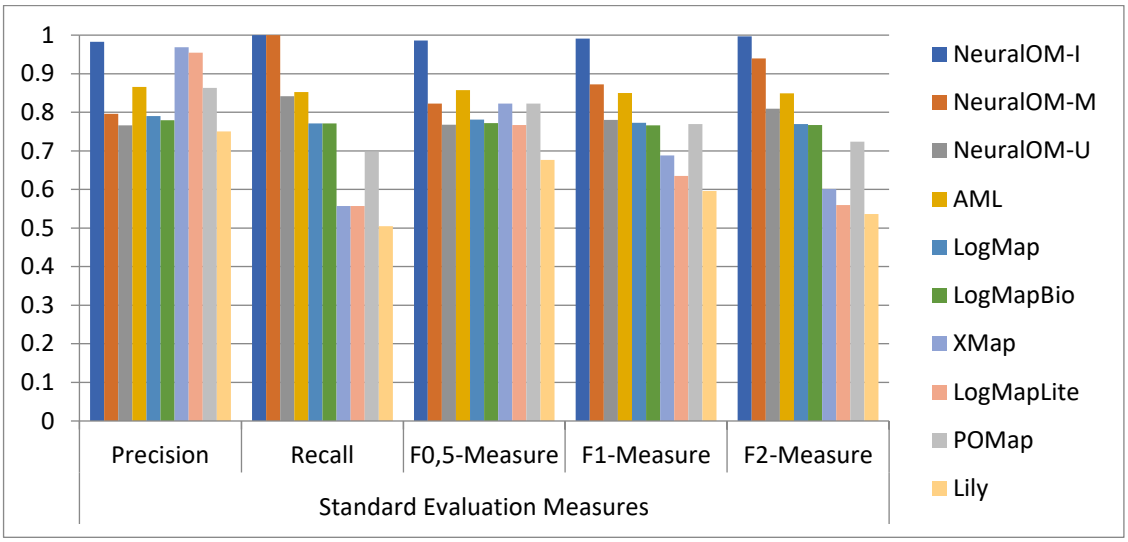

Figure 4.5. Evaluation results of NeuralOM against OAEI systems for BioDiv-FLOPO-PTO'18 track.

From Figure 4.5, it can be seen that: 1. The best precision values are given by NeuralOM-I, then by XMap and LogMapLite, then by AML and POMap, and then come the other systems with the best value obtained by NeuralOM-M. 2. Constantly, NeuralOM-I and NeuralOM-M achieved complete global recall values. A very good value is obtained by NeuralOM-U as well. Then, the OAEI systems are ordered decreasingly, except POMap which is out of order. 3. For $F_{0.5}$-measure, NeuralOM-I has by far the higher score. The other matching systems have close values around 0.8. For F_1-measure and F_2-measure, NeuralOM-I has the highest values followed by NeuralOM-M. The other systems are of a descending order starting from AML and NeuralOM-U, and more sharply for F_2-measure.

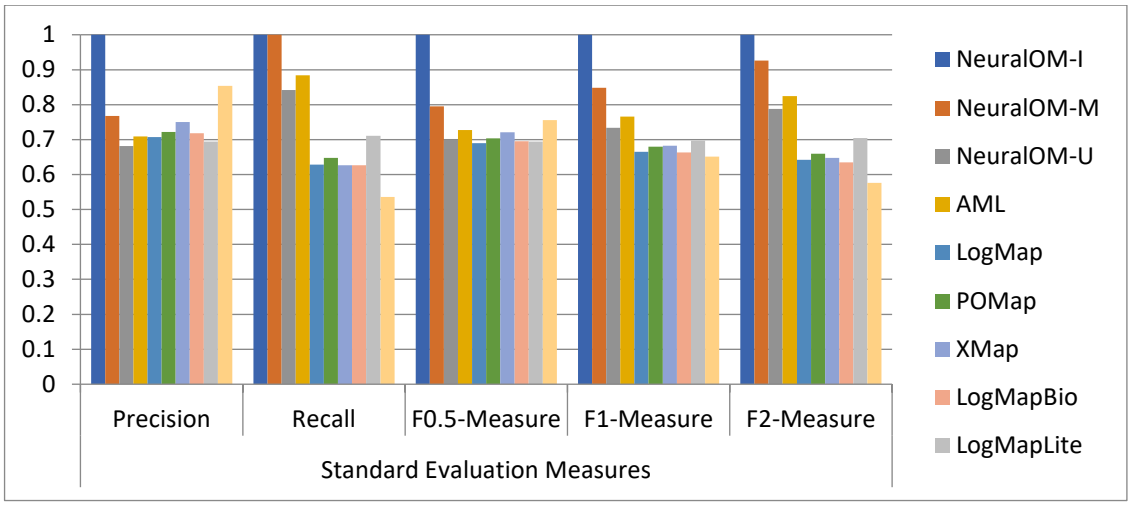

Figure 4.6. Evaluation results of NeuralOM against OAEI systems for BioDiv-ENVO-SWEET'18 track.

It is clear from Figure 4.6 that: 1. Globally, NeuralOM-I has complete scores for all the five evaluation measures. The next best results are given by NeuralOM-M (complete value of recall). Then, AML and NeuralOM-U obtain good results, and then come the other matching systems with acceptable results, presenting a slight descending order, excepting Lily and LogMapLite which are out of this order with higher scores for precision and recall respectively. 2. For F-measures, the OAEI values are closer in $F_{0.5}$-measure than in F_1-measure than in F_2-measure.

4.4.1.3. Evaluation for Process Model Matching Track

Plots in Figure 4.7 and Figure 4.8 illustrate the results of evaluating NeuralOM according to **OAEI'2017-PM Track *(PM'18)*** for UA and BR respectively, and summarises their comparison with OAEI matching systems.

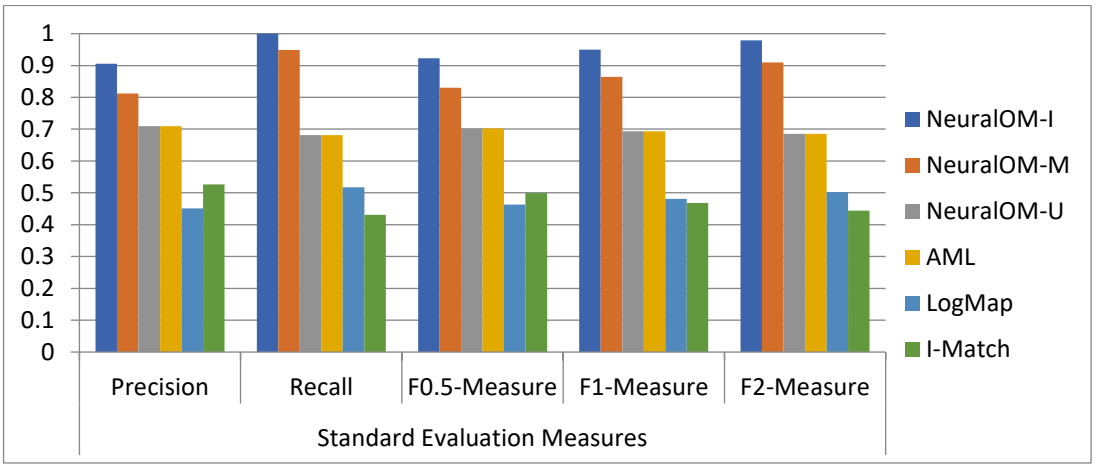

Figure 4.7. Evaluation results of NeuralOM against OAEI systems for PM-UA'17 track.

We can notice from Figure 4.7 that: Globally, systems have the same systems disparities for all evaluation measures. NeuralOM-I gets the highest results (with a complete recall value). NeuralOM-M gives excellent scores, followed by NeuralOM-U and AML with good and so close values. Log-Map and I-Match have the smallest performance with scores around 0.5.

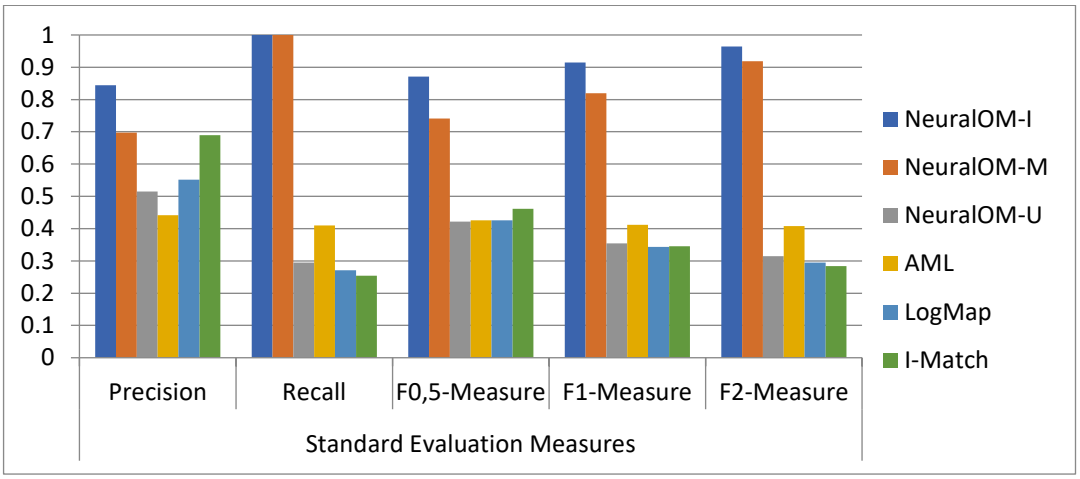

Figure 4.8. Evaluation results of NeuralOM against OAEI systems for PM-BR'17 track.

From Figure 4.8, it is clear that: 1. The best precision score, given by NeuralOM-I, exceeds 0.8. The next one is obtained by NeuralOM-M, then by I-Match. The other matching systems give values around 0.5. 2. Once again, NeuralOM-I and NeuralOM-M get the complete performance for recall with values equal to 1.0. Whereas the other matching systems give very small values, the score of AML is higher, but there is still a huge difference between their results and those of NeuralOM-I and NeuralOM-M. Those systems (contrary to NeuralOM-I and NeuralOM-M) have worse results for F_2-measure than for F_1-measure and than for $F_{0.5}$-measure. 3. For F-measures, the results obtained by NeuralOM-I and NeuralOM-M are excellent and by far better than the other matching systems, of which the $F_{0.5}$-values variances depend on those of precision and F_2-measure variances depend on those of recall whereas F_1-measure results are balanced and didn't achieve the average. 4. The distance between the two groups (NeuralOM-I and NeuralOM-M, and the other systems) is smaller for precision than the other evaluation measures.

4.4.1.4. Evaluation for Ontology Alignment for Query Answering Track

Figure 4.9 illustrates the results of evaluating NeuralOM according to **OAEI'2015- OA4QA Track *(OA4QA'18)***, and summarises their comparison with OAEI matching systems.

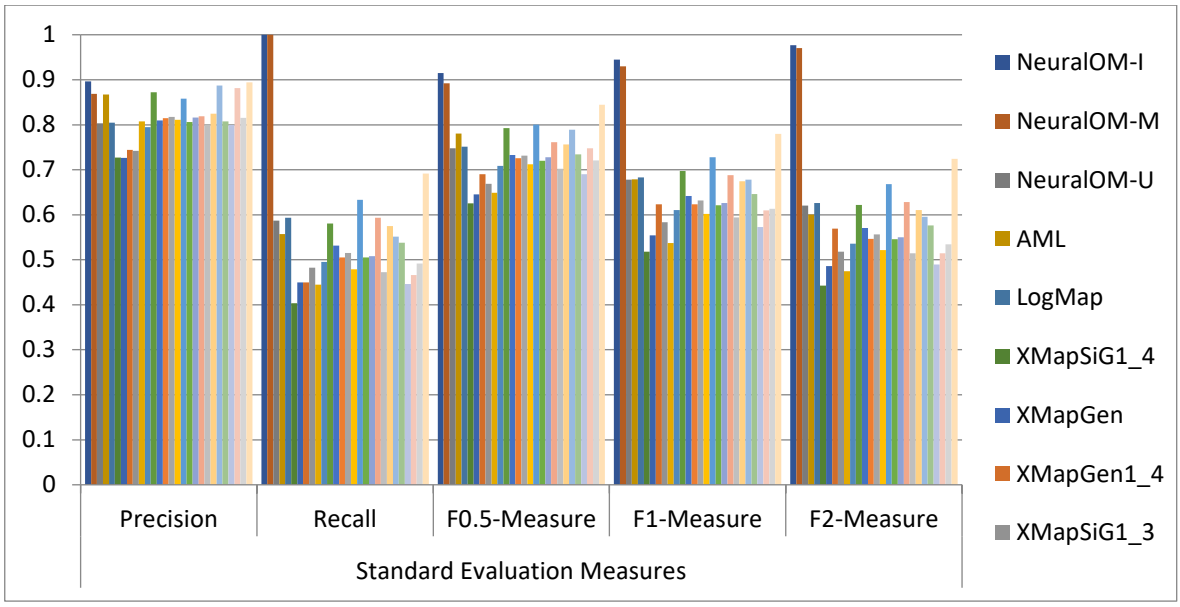

Figure 4.9. Evaluation results of NeuralOM against OAEI systems for OA4QA'15 track.

From Figure 4.9, we can observe that: 1. Precision results of all matching systems are better than recall values (around 0.8), excepting NeuralOM-I and NeuralOM-M which have complete values of recall. Thus, their $F_{0.5}$-measure results are also better than those of F_1-measure and better than F_2-measure results, but with sharp-less differences. 2. Recall values, excluding NeuralOM-I and NeuralOM-M with values equal to 1.0, and YAM++ and MaasMatch with a value equal to 0.7 and 0.63 respectively, did not exceed 0.6. 3. Since precision scores are more balanced and closer to each other than recall results, the variances of the three F-measures depend on those of recall, but with different intensities, sharply for F_2-measure than F_1-measure than $F_{0.5}$-measure. 4. In total, the best performances belong to NeuralOM-I and NeuralOM-M with high and far scores. After-ward, NeuralOM-U is among the best 9 systems (from 23 systems) which obtain good results.

4.4.1.5. Discussion of Results

According to the previous results, we can conclude the following:

- The challenge with the maximum average ***precision*** values is *BioDiv-FLOPO-PTO* (around 0.85). The next one is OA4QA with values around 0.8. Then, Conference has precision values around 0.75 and BioDiv-ENVO-SWEET's precision scores exceed 0.7. Finally, comes PM where the minimum average precision results belong to its *PM-BR* sub-track.

Precision values are separated in PM track and close to each other in the other test cases, less closely in BioDiv-FlOPO-PTO and OA4QA. In BioDiv-ENVO-SWEET, systems values are close and lower and far from our precision results.

All maximum precision values of all the six test cases of this experimental study are given by the intersection version of NeuralOM. The highest one is equal to 1.0 for BioDiv-ENVO-SWEET and the lowest one is equal to 0.84 for PM-BR. The highest minimum precision value is equal to 0.75 and given by Lily for BioDiv-FLOPO-PTO. The lowest minimum precision value in this experimental procedure is equal to 0.44 and given by AML for PM-BR tests.

- The track with the maximum average ***recall*** values is *BioDiv* with its two sub-tracks (around 0.75). The next task is PM-UA with values around 0.73 and then Conference with values around 0.6. PM-BR average results are slightly higher than 0.5. Finally, *OA4QA* has the poorest average recall values (around 0.5).

  Recall values are separated in all tracks excepting PM-BR and OA4QA, where recall values are somewhat close to each other but the gap between the results of NeuralOM-I and NeuralOM-M and those of the other systems is big.

  The maximum recall value in all this experimental procedure is equal to 1.0 and given by both NeuralOM-I and NeuralOM-M for all test cases, excluding PM-UA, where this complete value is obtained only by NeuralOM-I, and NeuralOM-M has a slightly lower but still excellent score. The highest minimum recall score is equal to 0.53 and given by Lily for BioDiv-ENVO-SWEET, and the lowest minimum recall value is equal to 0.25 and given by I-Match for PM-BR sub-track.

- The track with the highest average ***$F_{0.5}$-measure*** values is *BioDiv-FLOPO-PTO* (around 0.82), and next, Conference with average values equal to 0.75. Then, BioDiv-ENVO-SWEET and OA4QA have average $F_{0.5}$-measure results superior to 0.7. Finally, *PM-BR* has the lowest average scores (around 0.54).

  $F_{0.5}$-measure values are separated only in PM-UA and close to each other in the other test cases. In BioDiv-ENVO-SWEET, PM-BR and OA4QA, the $F_{0.5}$-measure results of our intersection and majority versions (even union in the 1st test case) are positively far from the other systems results.

  All the maximum $F_{0.5}$-measure values in these experiments are given by NeuralOM-I. The highest one is equal to 1.0 for BioDiv-ENVO-SWEET and the lowest one is equal to 0.87 for PM-BR. The highest minimum $F_{0.5}$-measure value is equal to 0.68

and given by LogMap for BioDiv-ENVO-SWEET, and the lowest minimum $F_{0.5}$-measure value is equal to 0.42 and given by NeuralOM-U for PM-BR sub-track.

- ***F_1-measure*** shows the real quality of matching. The task which has the maximum average F_1-measure results is *BioDiv-FLOPO-PTO* (around 0.8). BioDiv-ENVO-SWEET has a good one as well (around 0.74). The next track is Conference with values around 0.7. Then, PM-UA and OA4QA have average results so close to 0.7. Finally, *PM-BR* has the lowest results (around 0.56).

  F_1-measure values are separated in BioDiv-FLOPO-PTO and PM-UA and close to each other in the other tasks. Also, in BioDiv-ENVO-SWEET, PM-BR and OA4QA, F_1-measure results of our NeuralOM-I and NeuralOM-M versions (even NeuralOM-U in the 1st test case) are positively far from the other systems results.

  All the six maximum F_1-measure values of this experimental study are given by NeuralOM-I. The highest one is equal to 1.0 for BioDiv-ENVO-SWEET and the lowest one is equal to 0.91 for PM-BR. The highest minimum F_1-measure value is equal to 0.65 and given by Lily for BioDiv-ENVO-SWEET. The lowest minimum F_1-measure value is equal to 0.34 and given by LogMap for PM-BR sub-track.

- The challenge that has the maximum average ***F_2-measure*** results is *BioDiv-FLOPO-PTO* (around 0.77). The other sub-track average results are not far as well. The next one is PM-UA with values around 0.71. Then, OA4QA and Conference have average values superior to 0.6. And finally, *PM-BR* has the poorest scores (around 0.58).

  F_2-measure values are close to each other in all test cases, excepting PM-BR and OA4QA where the F_2-measure results of our intersection and majority versions are positively far from the other matching systems results.

  Also for this evaluation measure, all the maximum F_2-measure values are given by NeuralOM-I. The highest one is equal to 1.0 for BioDiv-ENVO-SWEET and the lowest one is equal to 0.96 for PM-BR. The highest minimum F_2-measure value is equal to 0.57 and given by Lily for BioDiv-ENVO-SWEET task. And, the lowest minimum F_2-measure value is equal to 0.28 and given by I-Match for PM-BR task.

4.4.2. Large-Scale Evaluation

In this section, we present the results of experimental procedure performed at the large scale.

4.4.2.1. Evaluation for Anatomy Track

Figure 4.10 illustrates the results of evaluating NeuralOM according to **OAEI'2018-ANATOMY-Track (Anatomy'18)**, and summarises their comparison with OAEI systems.

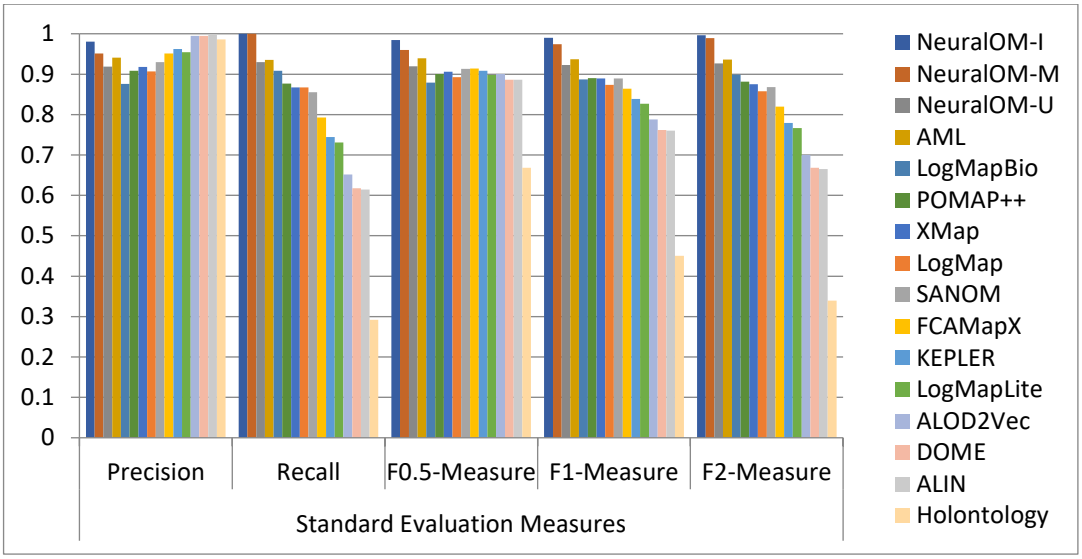

Figure 4.10. Evaluation results of NeuralOM against OAEI systems for Anatomy'18 track.

As can be seen from Figure 4.10: 1. Precision results present globally a decreasing order from NeuralOM-I down to LogMapBio (with a good value equal to 0.87), then an increasing one up to Holontology. 2. Recall values are decreasing, starting by NeuralOM-I and NeuralOM-M with a complete value equal to 1.0. NeuralOM-U value is slightly lower but still an excellent one. Recall values of the other matching systems present a descending order from 0.93 (of AML) to 0.29 (of Holontology). 3. The best F-measures values belong to NeuralOM-I and NeuralOM-M with values so close to 1.0. AML and NeuralOM-U also give excellent values. The other matching systems results are good as well, balanced for $F_{0.5}$-measure (excluding Holontology), and decreasing for F_1-measure and F_2-measure.

Figure 4.11 illustrates the Runtime evaluation results for OAEI'2018-ANATOMY-Track.

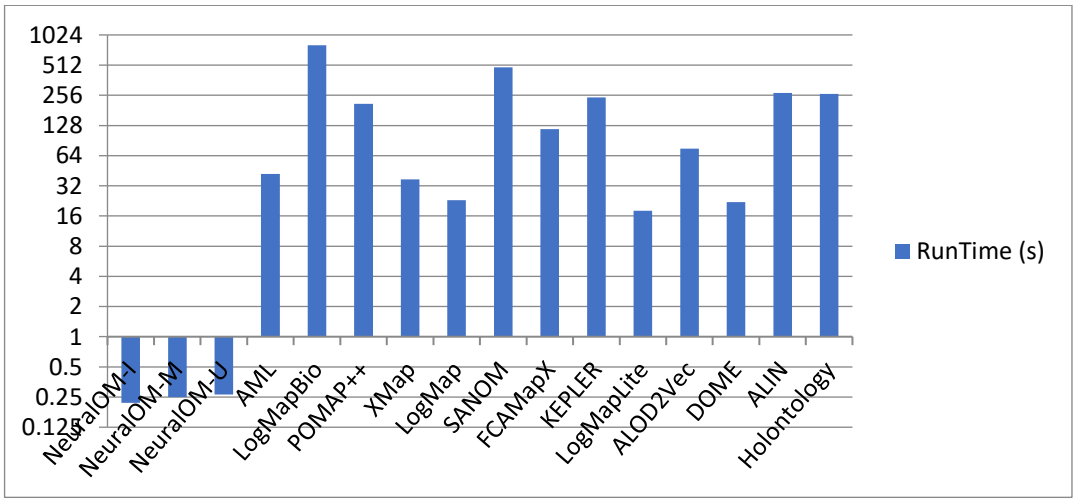

Figure 4.11. Runtime analysis of NeuralOM and OAEI systems for Anatomy'18 track.

It is clearly seen from Figure 4.11 that: 1. Required Runtime by OAEI matching systems is approximately confined between 16s (this minimum value belongs to LogMap) and 810s (this maximum value is given by LogMapBio). A disparity of values is inside, but the Runtime of the majority exceeds 60s. 2. The Runtime values of NeuralOM are roughly smaller; they did not even exceed 0.27s. There is a huge difference between these two classes (810/0.3=2700).

4.4.2.2. Evaluation for Disease and Phenotype Track

Figure 4.12 and Figure 4.13 illustrate the results of evaluating NeuralOM according to **OAEI'2018-PHENOTYPE-Track (Phenotype'18)** for **HP-MP** and **DOID-ORDO** Sub-Tracks respectively, and summarises their comparison with OAEI matching systems.

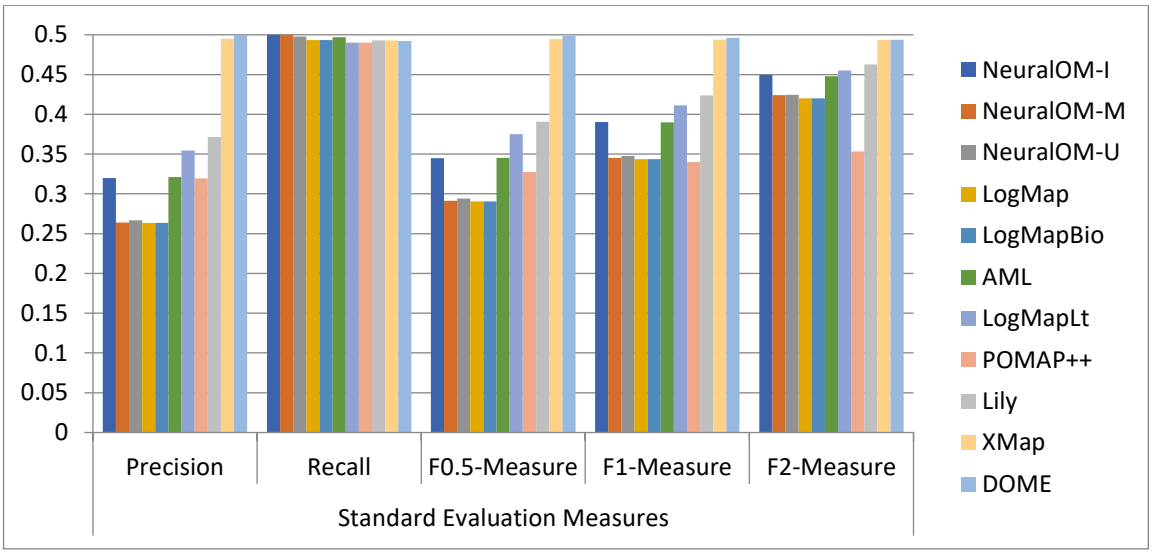

Figure 4.12. Evaluation results of NeuralOM against OAEI systems for Phenotype-HP-MP'18 sub-track.

From Figure 4.12, it is clear that: 1. DOME and XMap achieved the best precision results. Then, NeuralOM-I and four other systems are competitive with each other. Then, NeuralOM-M, NeuralOM-U and two other systems give medium results. 2. Recall results are balanced between the matching systems. The best scores belong to NeuralOM-I and NeuralOM-M which get a value equal to 0.5. Next, the other matching systems have competitive and medium results as well. 3. F-measures variances follow mostly precision results variation, especially $F_{0.5}$-measure, systems values are closer to each other for F_2-measure. Globally for these three measures, the best results are achieved by DOME and XMap, then by four systems including NeuralOM-I, then come the other matching systems.

4. The low scores of this test case is due to the fact that all matching systems give null values of the five evaluation measures for the 2nd partition.

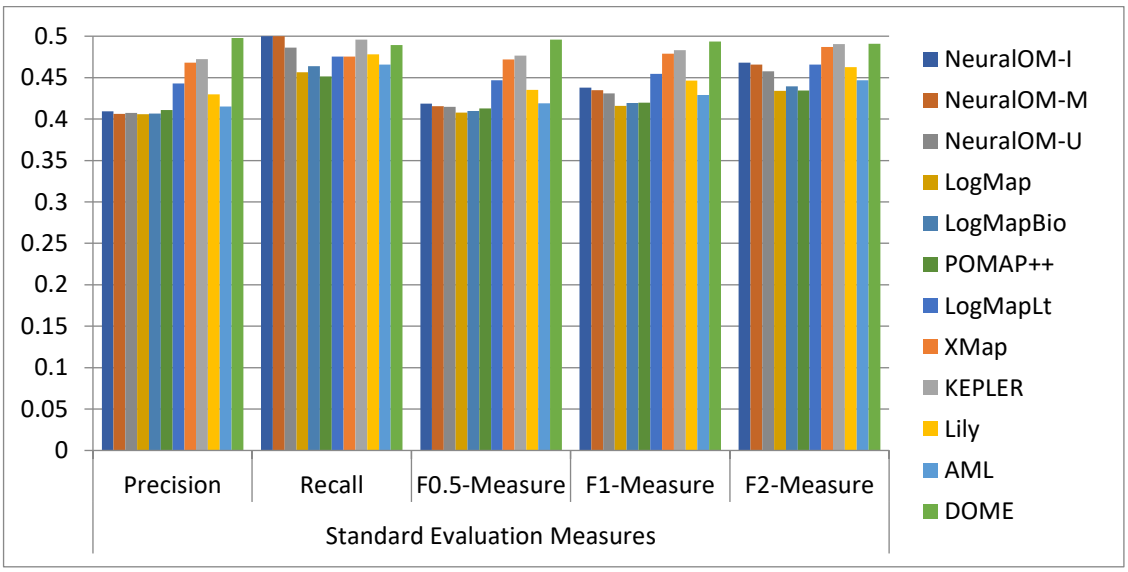

Figure 4.13. Evaluation results of NeuralOM against OAEI systems for Phenotype-DOID-ORDO'18 sub-track.

We can see form Figure 4.13 that: 1. Global evaluation measures values are decreased. This is due to the null values of the last three partitions. Partition1 and Partition2 present excellent results (those of the 2nd one are slightly better), Partition3 has average results, and zeros of the last three partitions modified the values average as if we are working on a superior limit of 0.5 instead of 1. 2. All matching systems, including the three variants NeuralOM give better results of recall (complete values by NeuralOM-I and NeuralOM-M and excellent ones by NeuralOM-U and the others) than of precision, excepting DOME which gets similar values of the two metrics, thus of the three others. 3. The best precision value belongs to DOME. XMap and KEPLER give good results. LogMapLt and Lily values are somewhat higher than the average whereas the last other matching systems ones are lower. 4. NeuralOM-I and NeuralOM-M get medium values of recall. KEPLER, XMap, NeuralOM-U, DOME, LogMap and LogMapLt give lightly lower recall results, then, close ones are given by the other matching systems. 5. F-measures results have the same variances as precision results but with slightly higher values. Globally, there is not a big difference between systems results, except for, XMap, KEPLER and DOME that are lightly better.

Figure 4.14 illustrates the Runtime evaluation results for OAEI'2018-PHENOTYPE-Track.

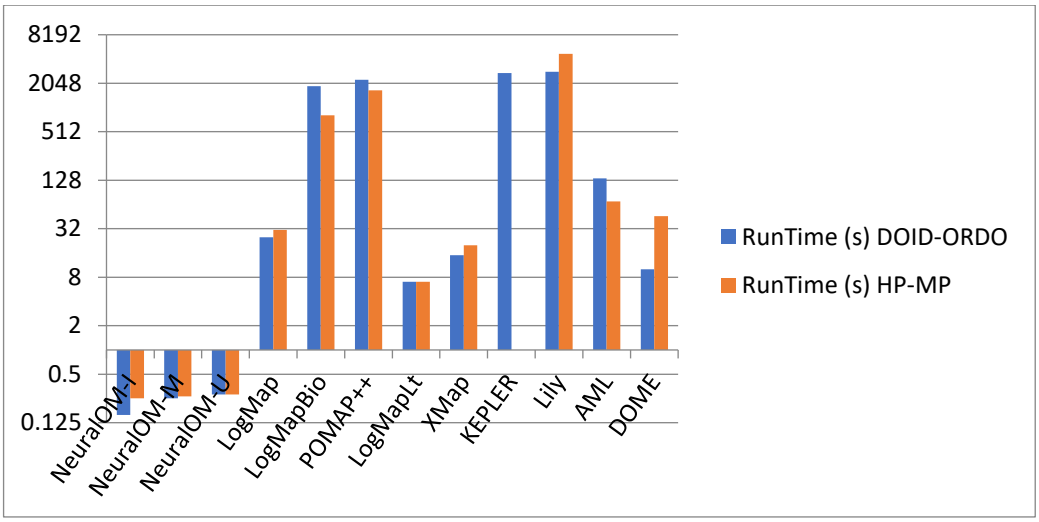

Figure 4.14. Runtime analysis of NeuralOM and OAEI systems for Phenotype'18 track.

From Figure 4.14, it is clearly seen that: 1. Required Runtime by OAEI matching systems is approximately confined between 7s (this minimum value is belonging to LogMapLite) and 4750s (this maximum value is Lily). Inside, LogMapBio, POMAP++ and KEPLER took significantly more time than the others. 2. There is not a huge disparity between HP-MP and DOID-ORDO in terms of Runtime, except by KEPLER which did not even participate at the second test case. 3. The Runtime values of NeuralOM are hardly smaller, not even exceeding 0.3s (a great difference comparing 4750 with 0.3). NeuralOM-M and NeuralOM-U have almost the same Runtime values for the two test cases. But, NeuralOM-I's Runtime for HP-MP is higher than that of DOID-ORDO by 0.1s.

4.4.2.3. Evaluation for Large Biomedical Ontologies Track

Plots in Figure 4.15, Figure 4.16 and Figure 4.17 illustrate the results of evaluating NeuralOM according to **OAEI'2017-LARGEBIOMED-Track (LargeBioMed'18)** for **FMA-NCI**, **FMA-SNOMED** and **SNOMED-NCI** Sub-Tracks respectively, and summarises their comparison with OAEI matching systems.

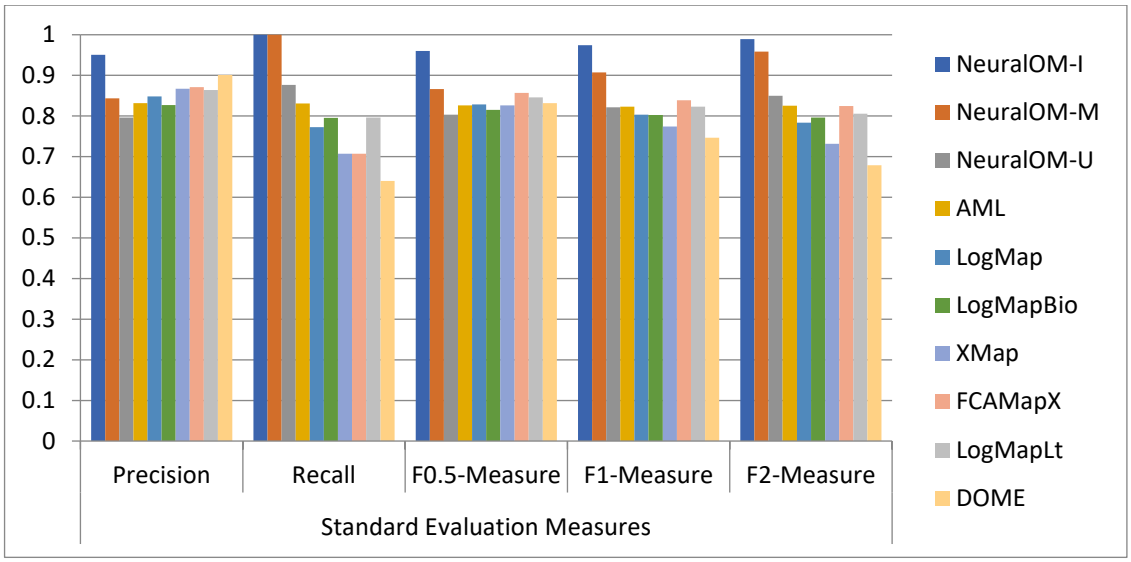

Figure 4.15. Evaluation results of NeuralOM against OAEI systems for LargeBioMed-FMA-NCI'18 sub-track.

From Figure 4.15, it is observed that: 1. NeuralOM-I, followed by NeuralOM-M, has achieved the best performance for all evaluation metrics (same performance for recall where both of them obtain 1.0), NeuralOM-U obtains a very good value of recall (0.87) and lower but still good value of precision. 2. OAEI matching systems are slightly increasing for precision, clearly decreasing for recall, also decreasing for F_2-measure and F_1-measure (less sharply especially for the 2nd measure) and balanced for $F_{0.5}$-measure. FCAMapX and LogMapLt are somewhat out of this order.

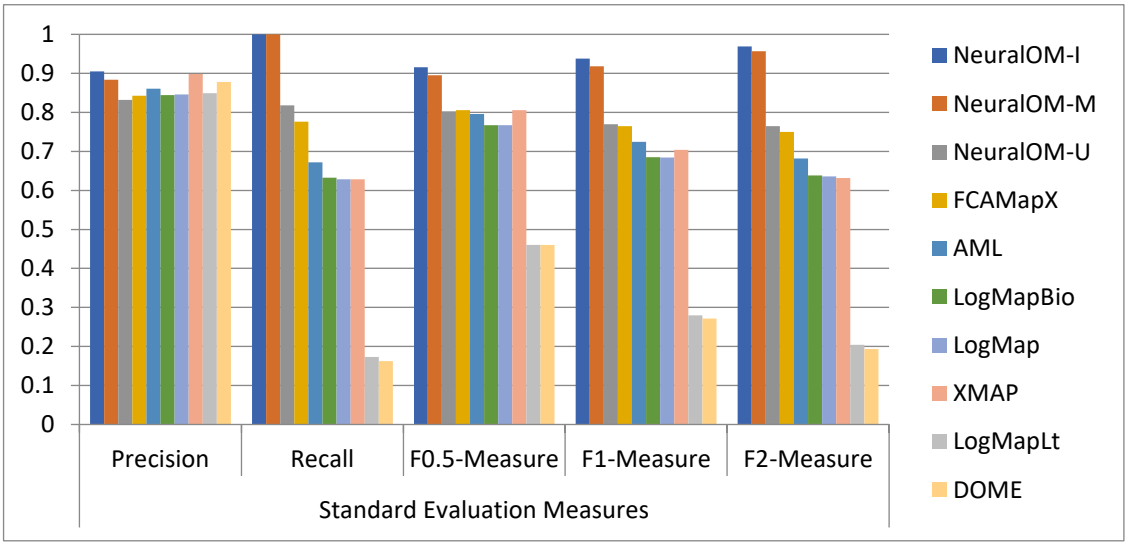

Figure 4.16. Evaluation results of NeuralOM against OAEI systems for LargeBioMed-FMA-SNOMED'18 sub-track.

From Figure 4.16, it is clearly seen that: 1. All participating systems, including the three proposed variants, have global precision values higher than 0.8. 2. For the other four measures, matching systems can be classified into three groups according to their results from the best to the worst performance; Excellent values by NeuralOM-I and NeuralOM-M (complete values of recall); NeuralOM-U and five OAEI systems from FCAMapX down to XMap, with different variances between measures (the most sharply is for recall); LogMapLt and DOME have the smallest values. 3. The best performance belong to NeuralOM-I and NeuralOM-M for all evaluation measures.

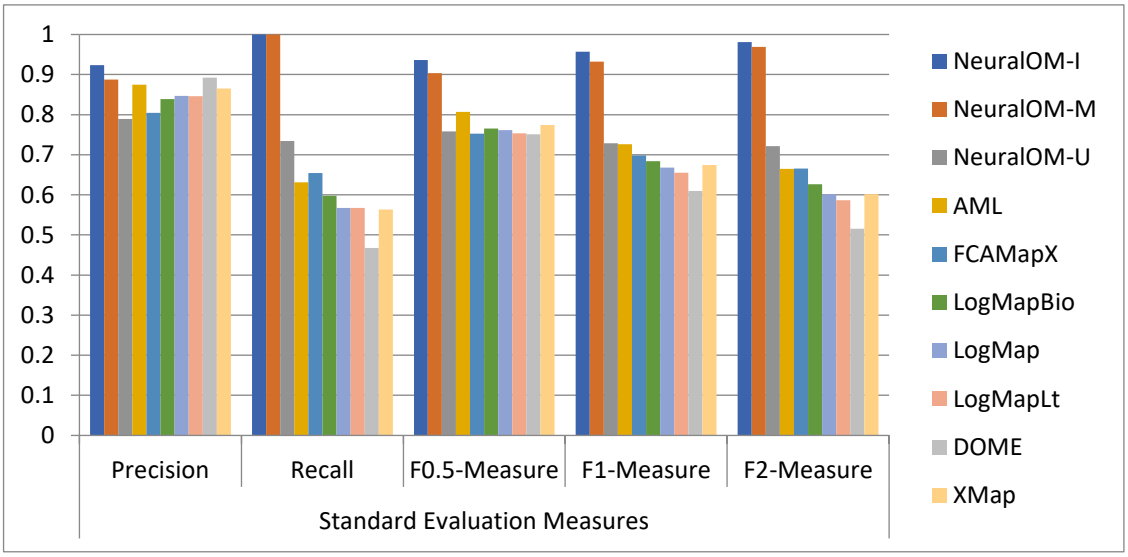

Figure 4.17. Evaluation results of NeuralOM against OAEI systems for LargeBioMed-SNOMED-NCI'18 sub-track.

Figure 4.17 shows that: 1. Complete recall values are offered by NeuralOM-I and NeuralOM-M, then, good values are given by NeuralOM-U, then come the other systems in a descending order. 2. Global precision values are somewhat close. The best ones are related to NeuralOM-I and NeuralOM-M, then to the other systems, which present also equilibrium for $F_{0.5}$-measure and a decline for F_1-measure and F_2-measure, stronger for the 2nd one. 3. For the five evaluation measures, NeuralOM-I and NeuralOM-M get by far the higher scores.

Figure 4.18 depicts the Runtime evaluation results for OAEI'2018-LARGEBIOMED-Track.

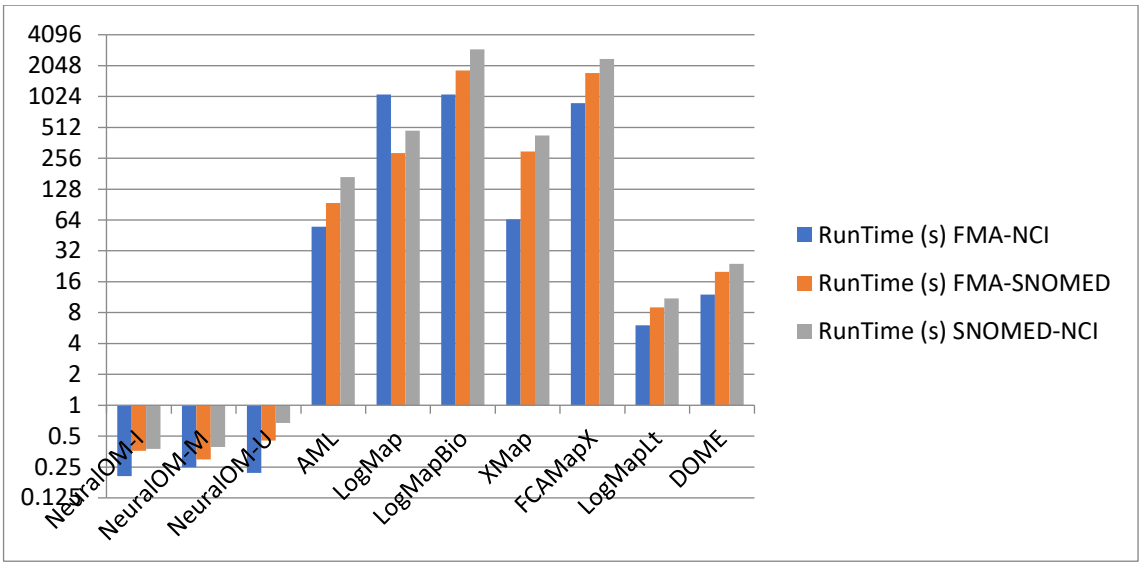

Figure 4.18. Runtime analysis of NeuralOM and OAEI systems for LargeBioMed'18 track

From Figure 4.18, it is clear that: 1. All OAEI matching systems took time in generating alignments for FMA-NCI less than FMA-SNOMED and less than SNOMED-NCI, with a same difference rate approximately, except XMap whose Runtime values of the first two test cases are far from that of the third one, and LogMap, which took more time for FMA-NCI. 2. Comparing systems shows that LogMapBio (with more than 2900s for SNOMED-NCI) and FCAMapX are the slowest systems. LogMap gives the next higher value. Then, XMap and AML have the following values. The smallest Runtime belongs to DOME and LogMapLt (6s for FMA-NCI). 3. As for NeuralOM, test cases are ordered samely. But, differences in NeuralOM-U values are bigger than those of NeuralOM-M, and Runtime's value of NeuralOM-I for FMA-NCI is far from the two other matching tasks of which values are so closed. 4. The nine Runtime values of NeuralOM are roughly smaller than those of OAEI matching systems (comparing parts of one with thousands of seconds!). All values of the proposed work did not even exceed 0.7s for the three LargeBioMed sub-tracks.

4.4.2.4. Discussion of Results

According to the previous results, we can conclude the following:

- The highest maximum ***precision*** value in all this experimental study is equal to 0.99 and given by ALIN for Anatomy. The lowest maximum precision value is equal to 0.5 and given by DOME for Phenotype-HP-MP. The highest minimum precision value is equal to 0.87 and given by LogMapBio for Anatomy. The lowest minimum precision value in this experimental procedure is equal to 0.26 and given by LogMap for Phenotype-HP-MP tests.

Among these six test cases, NeuralOM-I has tackled the maximum precision value three times. DOME has gotten it twice and the last one is achieved by ALIN. For the minimum precision value, it was given by NeuralOM-U in three test cases, by LogMap in two, and by LogMapBio in one test case.

- The highest maximum ***recall*** value in all this experimental procedure is equal to 1.0 and given by both NeuralOM-I and NeuralOM-M for all test cases. The lowest maximum recall value is equal to 0.5 and also given by NeuralOM-I and NeuralOM-M for Phenotype-HP-MP. The highest minimum recall score is equal to 0.63 and given by DOME for LargeBioMed-FMA-NCI, and the lowest minimum recall value is equal to 0.16 and given by DOME for LargeBioMed-FMA-SNOMED sub-track.

  NeuralOM-I and NeuralOM-M have achieved the maximum recall score for all tracks. For the minimum recall score, it was obtained by DOME 3 times, by POMAP++ twice, and by both of Holontology and LogMapLt once.

- The highest maximum ***$F_{0.5}$-measure*** value is equal to 0.98 and given by NeuralOM-I for Anatomy, and the lowest maximum $F_{0.5}$-measure value is equal to 0.49 and given by DOME for Phenotype. The highest minimum $F_{0.5}$-measure value is equal to 0.75 and given by DOME for LargeBioMed-SNOMED-NCI, and the lowest minimum $F_{0.5}$-measure value is equal to 0.15 and given by RSDLWB for LargeBioMed-FMA-NCI sub-track.

  Four of the six $F_{0.5}$-measure maximum scores have been achieved by NeuralOM-I. The two others have been obtained by DOME. The $F_{0.5}$-measure minimum score has been given by both of LogMap and DOME in two test cases and by both of Holontology and NeuralOM-U in one challenge.

- ***F_1-measure*** shows the real quality of matching. The highest maximum F_1-measure value is equal to 0.99 also given by NeuralOM-I for Anatomy. The lowest maximum F_1-measure value is equal to 0.49 and given by DOME for Phenotype. The highest minimum F_1-measure value is equal to 0.74 and given by DOME for LargeBioMed-FMA-NCI test case. The lowest minimum F_1-measure value is equal to 0.27 and given by DOME for LargeBioMed-FMA-SNOMED sub-track.

  NeuralOM-I has achieved the maximum F_1-measure score for four test cases. DOME has gotten it for the two others. The minimum F_1-measure value has been given by DOME 3 times and by both of Holontology, POMAP++ and LogMap once.

- The highest maximum ***F_2-measure*** value is equal to 0.99 and given by NeuralOM-I for Anatomy. The lowest maximum F_2-measure value is equal to 0.49 and given by DOME for Phenotype. The highest minimum F_2-measure value is equal to 0.67 and given by DOME for LargeBioMed-FMA-NCI sub-track. And, the lowest minimum F_2-measure value is equal to 0.19 and given by DOME for LargeBioMed-FMA-SNOMED sub-track.

  Among the six test cases of this experimental study, NeuralOM-I has obtained the maximum F_2-measure results for four ones, and DOME has gotten it for the two others. The minimum F_2-measure value is given by DOME in three test cases, by both of Holontology, POMAP++and LogMap in one matching task.

- The matching ***Runtime*** values of NeuralOM are so small compared to those of the OAEI systems that we were obliged to use a logarithmic scale base 2 to see the difference in the Runtime graphs presented above; otherwise, their bars are negligible and won't figure in the graphs. There is such a great difference comparing parts of one second with thousands of seconds. Therefore, the proposed approach is not time-consuming, and it will not complicate the process of matching such huge ontologies.

4.4.3. Experimental Summary

The matching systems present better results for precision than for recall in seven test cases: Conference (excepting NeuralOM-I, NeuralOM-M and ALIN), BioDiv-FLOPO-PTO, PM-BR, OA4QA, Anatomy (excepting NeuralOM-I and NeuralOM-M), LargeBioMed-FMA-SNOMED (with a slight difference) and LargeBioMed-SNOMED-NCI. Recall results are higher than precision results in the four matching tasks: ENVO-SWEET (with a slight difference), PM-UA, Phenotype-HP-MP (excepting XMap and DOME) and Phenotype-DOID-ORDO. In the challenge LargeBioMed-FMA-NCI, precision and recall results are similar. These variances are reflected on F-measures, where $F_{0.5}$-measure is mostly affected by precision and F_2-measure by recall, whereas F_1-measure combines them evenly.

For the proposed approach, the results of NeuralOM-I are better than those of NeuralOM-M and better than those of NeuralOM-U for all matching challenges, excepting Phenotype-HP-MP where NeuralOM-U presents better results than NeuralOM-M. In Conference, PM-BR, OA4QA, LargeBioMed-FMA-SNOMED and LargeBioMed-SNOMED-NCI, NeuralOM-U results are far from those of NeuralOM-I and NeuralOM-M. In Phenotype-HP-MP, NeuralOM-I results are far from those of the two others.

All in all, the detailed experimental procedure that we performed on twelve test cases shows that, the three variants of NeuralOM present better results than the OAEI matching systems, especially the Intersection and Majority variants which present the best matching performance. That is clear from the results of all the five evaluation measures adopted, especially from F_1-measure results. We can exclude only Phenotype (with null values in the last partitions by all systems), where the results of NeuralOM are similar with those of the OAEI systems in DOID-ORDO, and lower than the results of five systems and higher than three systems in HP-MP. This is because of the low initial weights of the three chosen systems for this track. We should also notice that this track is the track with the lowest matching performance.

All the maximum scores of all the evaluation measures adopted for all test cases of this study have been achieved by NeuralOM-I. NeuralOM-M has also presented excellent results positively far from the OAEI systems results. And NeuralOM-U has given very good results which are competitive with the two best OAEI matching systems. This high performance achieved by NeuralOM is due to two main reasons. First, the initial candidate matchers of which the generated mappings are required as input have been meticulously selected. Second, each step of NeuralOM is developed with a very high level of accuracy. The best value is fixed for each parameter by performing very detailed sets of tests.

4.5. Conclusion

In this chapter, we have presented NeuralOM, an automatic solution for large-scale ontology matching basing on artificial neural networks. We present an overview of the proposed solution and describe its contributions as well as its technical steps. Then, we present the very detailed experimental procedure that we performed on twelve test cases of different domains as well as of different dataset sizes from the OAEI initiative. The results of these experiments show that the proposed approach has proven its efficiency in front of all OAEI matching systems. NeuralOM has perfectly tackle the large-scale ontology matching challenges which have motivated this research.

In the next chapter, we present another different solution to the large-scale ontology matching issue which is an unsupervised method based on deep neural networks.

Chapter 5

Deep Embedding Learning with Auto-Encoder for Large-Scale Ontology Matching

Contents

5.1. Introduction

Ontology matching is an efficient method to support interoperability and remove heterogeneity among ontologies. As previously stated, large-scale ontology matching is still challenging for its long-time processing and large memory space consumption. Deep learning techniques are powerful computational models very appropriate for dealing with large datasets.

In the previous chapter, we have presented a reuse-based solution to the large-scale ontology matching issue. In this chapter, we present DeepOM, another different solution that we propose to deal with the large-scale heterogeneity problem using deep learning techniques. First, we provide an overview of the proposed ontology matching system as well as its main contributions. Then, we present the detailed workflow of DeepOM. After that, we describe the evaluation of DeepOM, conducted on the Anatomy track from the 2020 campaign of the OAEI Initiative. We present the results of these experiments and discuss the performance of our system.

5.2. DeepOM Overview

This chapter is related to our published work [198], where we propose the system DeepOM for automatically matching large ontologies without partitioning and basing on deep learning techniques. Figure 5.1 presents an overview of the proposed matching system. DeepOM first extracts the requisite ontological information from input ontologies and pre-process it. A reference ontology is then used to transform ontological concepts into numerical vectors that deep learning models can use as input. Auto encoders are common deep learning models. They are great at representation learning. Once the semantic embeddings for concepts are created, they can be used to train an auto-encoder, in order to output finer and smaller representations for ontological concepts. After that, the cosine similarity is used to compute similarities between the compact vectorial representations of concepts. Finally, a filtering process is applied using a defined alignment threshold, in order to keep only the most appropriate correspondences that compose the final mapping.

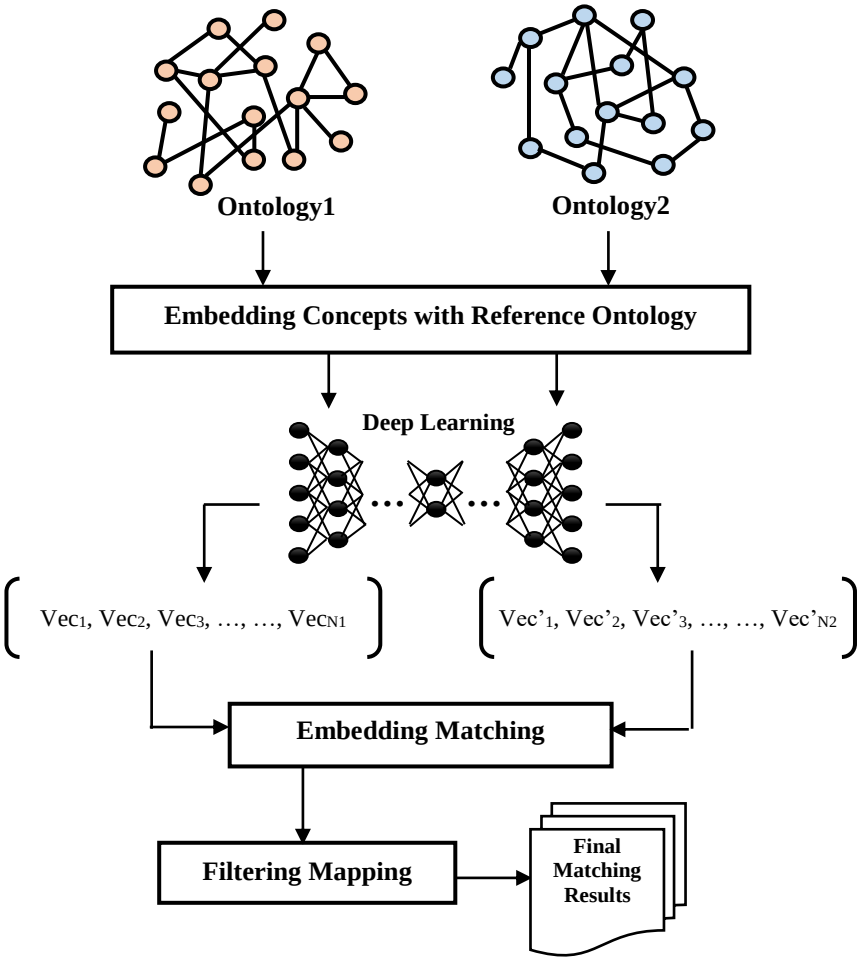

Figure 5.1. DeepOM Overview.

The main contributions of this system are:

- Employing deep learning techniques in order to effectively match large-scale ontologies without partitioning them, and at the lowest time process and memory space cost. Deep learning techniques are very appropriate for dealing with huge amounts of data. A massive dataset is adequate for learning for the reason that the model encounters and learns from a good enough number of examples.
- Representing the concepts of input ontologies in a multi-dimensional embedding space, using a smaller and well selected reference ontology. That aims for perfecting the matching performance and reducing its complexity.

  The aim of these representations is for transforming our data into vectors that deep learning models can use. Moreover, the obtained vectors represent the concepts in a richer and more precise way, since a concept is represented by a high number of dimensions according to the size of the reference ontology. For instance, if the reference ontology is of size 100, then, each concept will be represented by 100

values. i.e., concepts will be represented in a 100-dimensional vector space. Furthermore, passing the ontological concepts to vector representations reduce the complexity of matching them. This is due to the fact of manipulating float value vectors instead of manipulating a concept with several components of different types. Even if this transformation is a supplementary process, its complexity is still not considerable in regard with directly matching the huge input ontologies.

- The use of a reference ontology has a great impact on turning the ontological form of each concept from the input ontologies into a multi-dimensional numerical vector. Beside the fact that, this ontology is well selected, of the same domain as input ontologies and close to each of them equinely, the use of an ontology rather than other data structures is very expressive. i.e., all a semantic of the same domain is exploited for the purpose of embedding concepts.
- Training an auto-encoder on the concepts' embeddings, in order to learn more accurate and more compact representations for input concepts. That leads to better performance and less complexity as well.

  In one hand, this dimensionality reduction serves for improving the matching quality. Training the auto-encoder on the previous vectorial representations of concepts keeps the most important attributes of the input vectors in the compressed representations. That provides finer and more accurate representations for ontological concepts. In the other hand, as the auto-encoder compresses the input data into lower dimensional representations. This dimensionality reduction helps to reduce the complexity of ontology matching. In addition, the auto-encoder works in an unsupervised way that does not require a learning base, which necessitates a delicate process to prepare.

5.3. Deep Ontology Matching

In this section, we present ***DeepOM***, a system that we propose to address the ontology matching challenges at the large scale. The idea behind DeepOM is to automatically treat the large-scale ontology matching issue in two stages. At each stage, it seeks for providing more representative and less dimensional real-valued vectors for concepts of input ontologies.

- First, it creates semantic embeddings for ontological concepts, basing on the semantic similarity between them and the concepts of a smaller and well selected

reference ontology. That perfects the matching process and reduces the matching complexity.

- Second, DeepOM trains an auto-encoder on the generated concepts' vectors, in order to learn high-level and more compact embeddings for input concepts. This learning process also leads to better matching performance and decreases the complexity of large-scale ontology matching.

The processing workflow of DeepOM is illustrated in Figure 5.2. It could be summarized in four major phases: *Pre-Matching Phase*, *Embedding Phase, Deep Learning Phase* and *Matching Phase.*

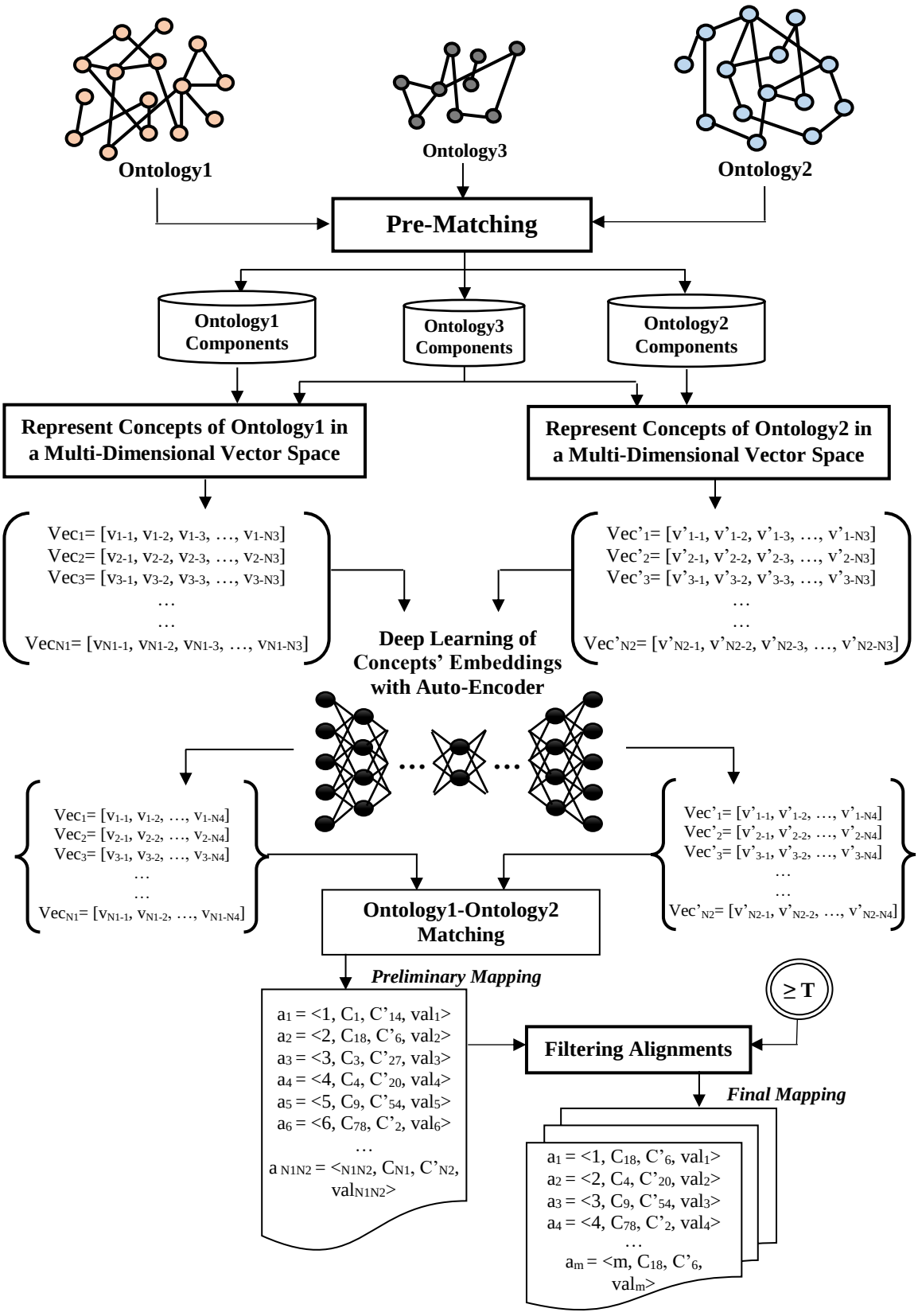

Figure 5.2. Processing Workflow of DeepOM.

Matching two given ontologies *Ontology1* and *Ontology2* is the process of finding a set of *m* correspondences (alignment) $A = \{a_1, a_2, a_3, \dots, a_m\}$. Each correspondence $a_{i,\ (i=1\text{-}m)}$

is defined by a quadruple as: a_i = < id_i, C, C', val_i >. Where: id_i is the correspondence identifier; C is a concept from *Ontology1*; C' is a concept from *Ontology2*; and val_i the correspondence value between C and C' provided by DeepOM. This latter is in range [0,1], and reflects the similarity measure between the linked concepts.

5.3.1. Pre-Matching

The first step aims for preparing ontologies for matching. It is such an important process, since the input ontologies are heterogeneous and different in their components' availability and entities' lexicon for our interests. We pre-match input ontologies in two main sub-steps:

5.3.1.1. Extracting Ontological Information

This step consists on loading the ontologies needed for matching, and extracting their components which are necessary for generating alignment. An ontological concept is defined by its semantic triangle (see Figure 5.3). The vertices of this 3-dimensional shape represent the three main aspects of the concept:

1. ***Term:*** expresses the concept in language. It is the linguistic representation of a given concept.

   For example, as shown in Figure 5.3, to express the concept that brings together the different vehicle objects, terms such as "*Vehicle*", "*Automobile*", "*Car*", "*Auto*", "*Motor vehicle*" or even "*Wheels*" can be used.
2. ***Intention:*** is the set of its qualitative or functional properties which constitutes its meaning. A property of a concept can be of one of this two types:
   - ***Attributes:*** represent the *internal structure* of the concept. i.e., the features or characteristics that objects of this concept can have.

   As shown in the example of the figure below, "*Model*", "*Body mass*", "*Registration number*", "*Suspension stiffness*", "*Axle hop frequency*", "*Tire stiffness*" and "*Color*" are attributes of the class Vehicle.

   - ***Relations:*** represent the *external structure* of the concept. i.e., the relations of this concept with other concepts of the ontology.

   For the presented example, the vehicle concept is in relationship with the concept "*Person*" by the relation "*own*", with "*Support*" by "*supply*", with "*Steering Device*" by "*control*", with "*Transport*" by "*is-a*", with "*Driver*" by "*conduct*", with "*Vehicle Component*" by "*has-part*", with "*Time*" by "*damage-at*", with "*Vehicle Function*" by "*has-function*" and many others.

3. ***Extension:*** is the set of the objects denoted by the concept. For example, "*Mercedes*", "*Audi*", "*BMW*", "*Ferrari*", "*Lamborghini*", "*Volkswagen*" are entities falling into the category of vehicle.

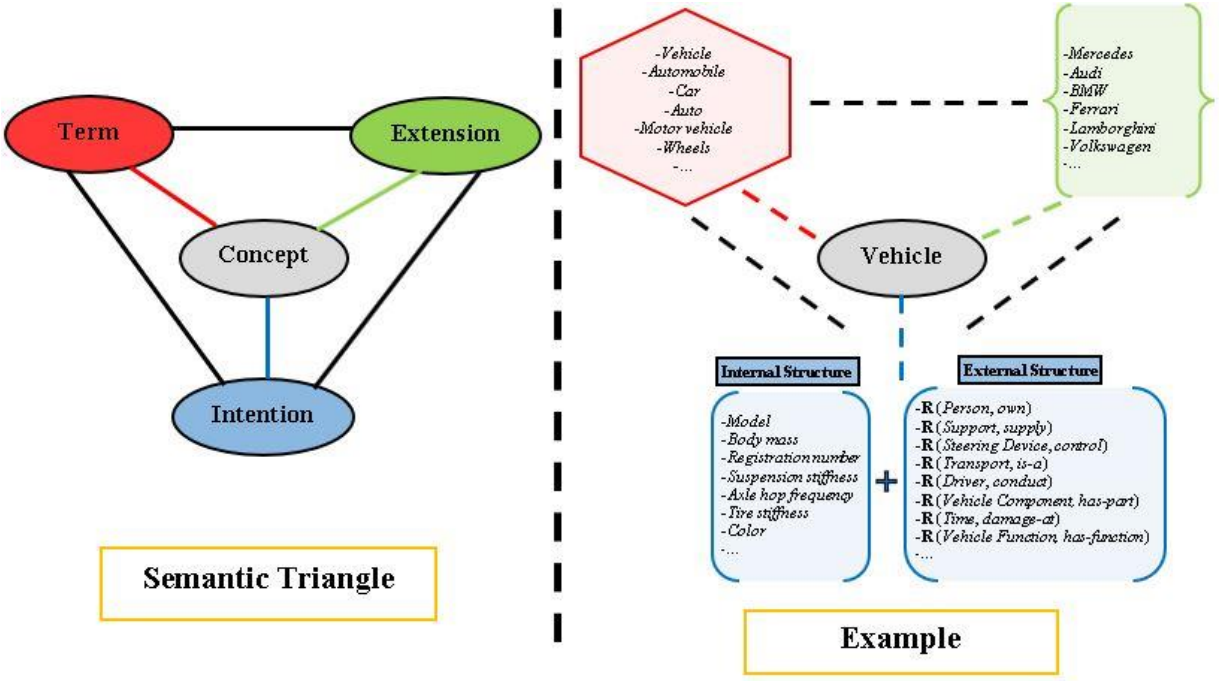

Figure 5.3. Ontology concept by three dimensions. The left side presents the semantic triangle of the concept; The right side presents an example.

Therefore, to keep the semantics carried by a concept in an ontology, it should be defined by the three elements cited above. In this study, as we aim for performing the matching process in a complete way, we care about the semantics of input ontologies and we cover this 3-dimensional view of their concepts. Thus, we extract for each concept *C* from *Ontology1* and *Ontology2* its:

- ***Lexical label***, which is the representative term used to describe this concept;
- ***Related concepts***, which are the concepts from the same ontology that are related to the concerned concept. Concepts of ontologies are related to each other with distinct types of relationships. The most basic type of relations in an ontology is the *subsumption* relation, also known as *is-a* relation. It provides the tree-like taxonomy of the ontology. By this relation, an ontological concept has principally a ***parent-concept***, a ***child-concept*** and a ***sibling-concept***. According to this structure, three main types of related concepts are extracted:

- ***Ancestors***, which are the elements of the set of concepts composed by, the parents of *C*, and the parents of their parents, along the path to root. i.e., we extract all parent-concept levels of the concept in question;
- ***Descendants***, which are the direct child-concepts of *C*. As a concept has a significant number of child-concepts compared with parent-concepts, we find that the first child-concept level is sufficient to have related descendants' concepts;
- ***Siblings***, which are the direct child-concepts of the direct parent-concepts of *C*. i.e., concepts of the first child-concept level of the first parent-concept level of the concerned concept are extracted;

- ***Individuals***, which represent the instance-level of the concept, described by its concrete objects.

5.3.1.2. Pre-Processing of Ontological Components

In this sub-step, we mainly interest in pre-processing the lexical information extracted from input ontologies. Thus, once the ontological components are extracted, we analyze and process, for each concept, its label, individuals' names, as well as the labels of its related concepts (ancestors, descendants and siblings). Considering an extracted textual information *T* to be pre-treated. The pre-processing task outputs a set of processed terms. It is performed as present the following points:

- *Tokenization:* consists on segmenting *T* to a set of tokens according to space (' ') and two types of dashes ('-' and '_');
- *Removing stop words:* consists on removing the commonly used words which do not carry useful information for matching. For that, we use the English **nltk**[13] stop-words list;
- *Denoising:* aims to get rid of unhelpful elements of the textual information. In the case of this study, it consists on lowercasing all characters, removing tokens of length 1 (excepting numbers) and removing punctuations marks as well as all special and non-ASCII characters.

[13] http://www.nltk.org/

5.3.2. Creating Semantic Embeddings for Concepts

This step consists on transforming the concepts of input ontologies into vectorial representations that deep learning models can use as input. We use another ontology, called ***reference*** ontology, in order to represent each concept from *Ontology1* and *Ontology2* in a numerical multi-dimensional vector space.

For the following, the set of *Ontology1*'s concepts is defined by $\boldsymbol{C} = \{\boldsymbol{c_{i,\ i=1-N1}}\}$, the set of *Ontology2*'s concepts by $\boldsymbol{C'} = \{\boldsymbol{c'_{i,\ i=1-N2}}\}$, and the set of concepts of the reference ontology by $\boldsymbol{C''} = \{\boldsymbol{c''_{i,\ i=1-N3}}\}$, where N1, N2 and N3 denote the number of concepts of *Ontology1*, *Ontology2* and the reference ontology respectively.

Algorithm 5.1 demonstrates the task of this step. It is performed based on computing similarities between concepts of the reference ontology and elements of C and C'. We represent each concept from *Ontology1* and *Ontology2* by a vector of N3 numerical values. Each value is the similarity score between the concerned concept and a concept from the reference ontology. Since the size of the reference ontology is N3, all vectorial representations of concepts of *Ontology1* and *Ontology2* are of length N3 for each vector. i.e., concepts of input ontologies are represented in a N3-dimensional vector space. As results of this step, the embedding representations of ontologies' concepts are created after that the execution of the algorithm is completed.

Algorithm 5.1. *Creating concepts' embeddings for input ontologies*

Input: Ontology1's concepts: $\boldsymbol{C} = \{\boldsymbol{c_{i,\ i=1-N1}}\}$;
Ontology2's concepts: $\boldsymbol{C'} = \{\boldsymbol{c'_{i,\ i=1-N2}}\}$;
Reference ontology's concepts: $\boldsymbol{C''} = \{\boldsymbol{c_{i,\ i=1-N3}}\}$.

Begin

// Vectorial representations for elements of C
Initialization of V = { }
for i from 1 to N1
    Initialization of vec_i = []
    for j from 1 to N3
        $vec_i[j]$ = Semantic similarity value between c_i and $c"_j$
    append vec_i to V
// Vectorial representations for elements of C'
Initialization of V' = { }
for i from 1 to N2
    Initialization of vec'_i = []

for j from 1 to N3
vec'$_i$[j] = Semantic similarity value between c'$_i$ and c"$_j$
append vec'$_i$ to V'
End
Output: Embeddings of C: V = (vec$_i$, $_{i = 1\text{-}N1}$), vec$_i$ = [v$_j$, $_{j = 1\text{-}N3}$];
Embeddings of C': V' = (vec'$_i$, $_{i = 1\text{-}N2}$), vec'$_I$ = [v'$_j$, $_{j = 1\text{-}N3}$].

The accuracy of the generated embeddings is highly dependent upon two major factors: *defining the reference ontology* and *similarity measurement*.

5.3.2.1. Defining Reference Ontology

The reference ontology has a great impact on the performance of the matching process. Thus, its determination is such a delicate and careful task. It depends on input ontologies and should be:

- Of the same domain as *Ontology1* and *Ontology2*, and semantically close to them. Otherwise, the embedding vectors would be overpowered by zeros. Thus, they would not provide the real representations for concepts.
- In the-middle-of-the-road between *Ontology1* and *Ontology2*. i.e., it should be neutral and balanced between them, so as to afford fair concepts' representations.
- Of an appropriate size. i.e., it must not be very small, not useful, nor larger than input ontologies, so matching gets more complicated.

5.3.2.2. Similarity Measurement

The adequacy of the numerical values of the concepts' embeddings relies on how similarities are computed between input ontologies and the reference ontology. As this is a very exact task, we perform it on a high level of accuracy, exploiting the three main aspects of ontological concepts. Therefore, we proceed and combine several matchers:

5.3.2.2.1. Terminological Matcher

The terminological matcher exploits semantics inside concepts' lexicon. It measures both *context-based* similarity and *syntactical* similarity. And, it combines them in a way that, the weight assigned to each element reflects its accurate need proportion in that current case. We propose the following formula to compute the terminological similarity between two concepts C_1 and C_2:

$$TerSim(C_1, C_2) = \frac{2 \times |Label_1 \cap Label_2| + D_1 + D_2}{|Label_1| + |Label_2|} \quad (5.1)$$

Where:

Label₁ is the pre-processed label of C_1; *Label₂* is the pre-processed label of C_2; D_1 is the similarity report of $(Label_1 - Label_2)$ compared to $(Label_2 - Label_1)$; D_2 is the similarity report of $(Label_2 - Label_1)$ compared to $(Label_1 - Label_2)$. For each pair of individual terms, we take the maximum similarity between the two values provided by an ***external knowledge resource*** and ***Jaro measure***.

5.3.2.2.2. Structural Matcher

The structural matcher measures the similarity between concepts basing on their structure, which refers to their related concepts. We use the following formula to compute structural similarities between ancestors, descendants and siblings of C1 and C2:

$$\boldsymbol{StrSim(C_1, C_2) = \frac{2 \times |RC_1 \cap RC_2| + D_1 + D_2}{|RC_1| + |RC_2|}} \quad (\mathbf{5.2})$$

Where:

RC₁ is the set of concepts related to C_1; *RC₂* is the set of concepts related to C_2; D_1 is the similarity report of $(\boldsymbol{RC_1 - RC_2})$ compared to $(\boldsymbol{RC_2 - RC_1})$; D_2 is the similarity report of $(\boldsymbol{RC_2 - RC_1})$ compared to $(\boldsymbol{RC_1 - RC_2})$. This similarity report is computed using the terminological similarity equation (Equation.1) for each pair of individual related concepts.

Related concepts (*RC₁* for C_1 and *RC₂* for C_2) refers to sets of ancestors, descendants and siblings of C_1 and C_2 for each case.

5.3.2.2.3. Extensional Matcher

We measure the similarity between instances of C_1 and C_2 using the Jaccard similarity, given by formula:

$$\boldsymbol{ExtSim(C_1, C_2) = \frac{|Inst_1 \cap Inst_2|}{|Inst_1 \cup Inst_2|}} \quad (\mathbf{5.3})$$

Where:

Inst₁ is the set of instances of C_1 and *Inst ₂* is the instances set of C_2.

Combining the individual similarity measures is necessary in order to get the final semantic similarity between C_1 (from C+C') and C_2 form (C"). Unlike other ontology matching techniques, which give equal weights for similarity values, DeepOM combines them in an additive way that, the lack individuals and related elements for some concepts would not affect the matching performance. Algorithm 5.2 demonstrates this process.

Algorithm 5.2. *Measuring semantic similarity between two given concepts*

Input: C_1: Concept from Ontology1+Ontology2: Set of N1 concepts: C + Set of N2 concepts: C', $\boldsymbol{C} = \{\boldsymbol{c_{i,\ i=1-N1}}\}$, $\boldsymbol{C'} = \{\boldsymbol{c'_{i,\ i=1-N2}}\}$;
C_2: Concept from Reference ontology: Set of N3 concepts: C", $\boldsymbol{C''} = \{\boldsymbol{c''_{i,\ i=1-N3}}\}$;
Sim_Threshold.

Begin

// Computing similarities between C_1 and C_2

Ins_1 = list of c_1 Instances
Anc_1 = list of c_1 Ancestors
Des_1 = list of c_1 Descendants
Bro_1 = list of c_1 Siblings
Ins_2 = list of $c"_2$ Instances
Anc_2 = list of $c"_2$ Ancestors
Des_2 = list of $c"_2$ Descendants
Bro_2 = list of $c"_2$ Siblings
TerVal = TerSim(c_1,$c"_2$)
AncVal = StrSim(c_1,$c"_2$) \\ RC_1==Anc_1; RC_2==Anc_2
DesVal = StrSim(c_1,$c"_2$) \\ RC_1==Des_1; RC_2==Des_2
BroVal = StrSim(c_1,$c"_2$) \\ RC_1==Bro_1; RC_2==Bro_2
ExtVal = ExtSim(c_1,$c"_2$) \\ $Inst_1$==Ins_1; $Inst_2$==Ins_2

// Combining similarities between C_1 and C_2

```
StrVal = Average(AncVal,DesVal,BroVal)
If (StrVal>Sim_Threshold) and (ExtVal>Sim_Threshold):
        SemVal = Average(TerVal,StrVal,ExtVal)
Else: If (StrVal>Sim_Threshold):
            SemVal = Average(TerVal,StrVal)
        Else: If (ExtVal>Sim_Threshold):
                    SemVal = Average(TerVal,ExtVal)
              Else:
                  SemVal = TerVal
```

End

Output: SemVal: Semantic Similarity value between *two given* concepts;

5.3.3. Deep Ontology Matching with Auto-Encoder

This step consists on using deep learning techniques to learn high-level embeddings for concepts of the two ontologies. This task aims to provide ***more accurate*** and ***less dimensional*** representations for concepts. That perfectly represents the input ontologies in an unsupervised way.

Auto-encoders seem to be very appropriate for such purposes. They are capable of creating sparse representations of the input data. Therefore, they can be used to compress the concepts' vectors resulted from the previous step, and represent them in a latent space.

Figure.5.4 illustrates the architecture of the auto-encoder model of DeepOM. It is a deep neural network with multiple layers. The output layer has the same dimension as the input layer. And, the architecture between them is mirrored. The model has two components: ***Encoder*** and ***Decoder***. The encoder compresses the input data into a lower dimension. Then, the decoder uses the compact representations to recreate the original input.

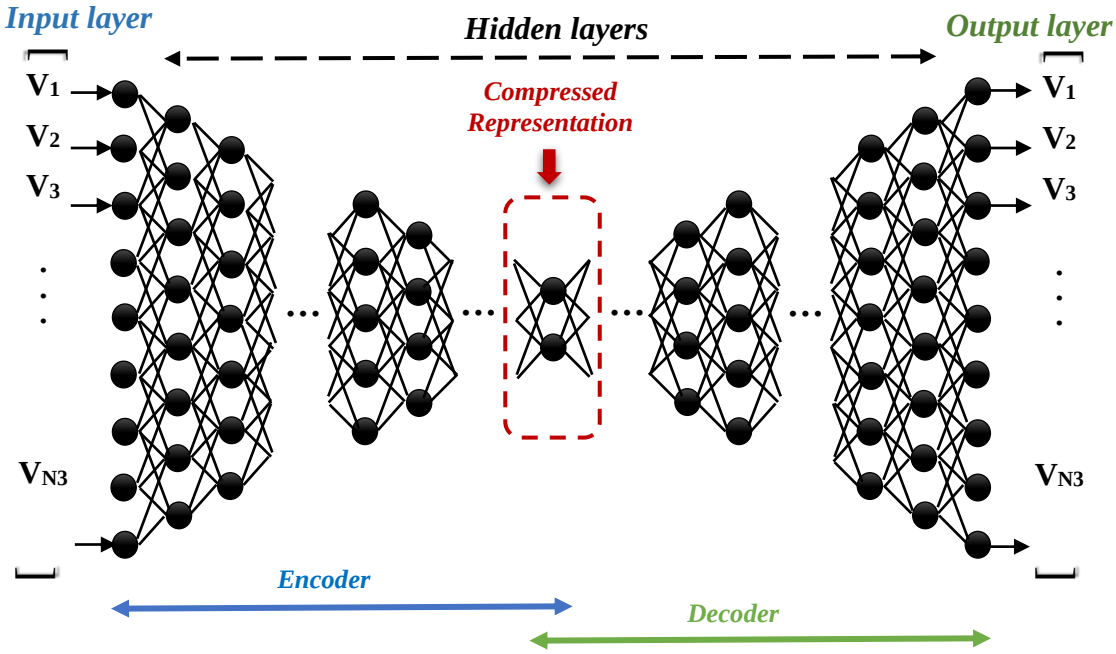

Figure 5.4. Architecture of the Auto-Encoder Model.

The number of layers of the deep network, the number of nodes per layer, the number of training epochs, as well as the other auto-encoder parameters are fixed by trial-and-error. For training the auto-encoder model, we use the back-propagation learning method. We take the current concepts' vectorial representations as input. Then, we train the model to learn weights so as to compress down these vectors into a lower dimensional space. The final learned representations (of size N4 as shown in Figure 5.2) keep the most important features of ontological concepts.

5.3.4. Generating Ontology1-Ontology2 Alignment

This final step is performed in two sub-steps:

5.3.4.1. Measuring Embeddings Similarity

Generating alignment between *Ontology1* and *Ontology2* requires computing similarities between their concepts. Since those concepts are represented by numbers in a vector space, the matching process consists on computing similarities between the corresponding vectors. For that, we use the cosine similarity measure defined by:

$$Cos(\vec{x}, \vec{y}) = \frac{\vec{x}.\vec{y}}{\|\vec{x}\|.\|\vec{y}\|} \quad (5.4)$$

5.3.4.2. Pruning Generated Alignment

Once similarities between concepts of the input ontologies have been measured, they undergo a filtering procedure, so as to keep only significant correspondences. For example, we do not consider a concepts' pair of which the value is equal to 0.0 as a valid correspondence. For that, we define an alignment threshold T, fixed by trial-and-error, to extract the final mapping, for which the similarity values exceed T. We aim by this task at improving the matching accuracy by removing irrelevant correspondences of low similarity scores, and keeping only the most appropriate correspondences.

5.4. Evaluation Framework

In this section, we describe the experimental procedure that we proceed for evaluating our ontology matching system.

5.4.1. Experimental Design

Aiming for studying the efficiency of DeepOM, we evaluate it according to the Anatomy track of the Ontology Alignment Evaluation Initiative (OAEI). More details about this international initiative and its evaluation challenges are given in Sect.6.4.1. As we look for passing the ontology matching task to the large scale, Anatomy track proposes test ontologies of appropriate sizes. The OAEI'2020 Anatomy track[14] comprises a single real-world test case about matching two fragments of biomedical ontologies describing the human anatomy and the anatomy of the mouse. The ontologies to be matched are the **human** and **mouse** OWL ontologies with 3304 and 2744 classes respectively. The task is situated in a domain where we find large and carefully designed ontologies which are described in technical terms. The evaluation is based on a manually curated reference alignment.

[14] http://oaei.ontologymatching.org/2020/anatomy/index.html

For evaluating the proposed system, we use the standard evaluation measures: precision, recall as well as their harmonic mean F-measure, against the reference alignments of the Anatomy test case. They are previously defined and described in (Sect 1.3.5.3). They are the most widely used criteria for such evaluation.

For the external resource, which is required for measuring the terminological similarity, we use **BioWordVec [199]**. It is an open set of biomedical word embeddings of 2,324,849 distinct words. It combines sub-word information from unlabeled biomedical text with MeSH[15], a widely-used biomedical controlled vocabulary.

As reference ontology, we use the Anatomical Entity Ontology: **AEO[16]**, an OWL ontology of anatomical structures with 250 classes. It expands the Common Anatomy Reference Ontology (CARO)[17], which is an upper-level ontology to facilitate interoperability between existing anatomy ontologies for different species. AEO is intended for being useful in increasing the knowledge amount of anatomy ontologies, facilitating annotation and in enabling interoperability across anatomy ontologies.

The structure of the trained auto-encoder model is [250-200-150-100-150-200-250]. The size of the reference ontology is 250. That generates a vectorial representation of 250 numbers for each concept. It is token as input and output to the network.

5.4.2. Experimental Results

In order to study the efficiency of DeepOM, we compare its results with the results[18, 19, 20] of the new systems participant to the same matching challenge for the three most recent OAEI campaigns. The results of the performed evaluation are summarized in Table 5.1. The best scores for each evaluation measure are marked in bold. The OAEI systems with which DeepOM is compared are: ATBox [200], OntoConnect [201], ALOD2Vec [202], FCAMap-KG [11], DOME [12], DESKMatcher [13], Holontology [14] and AGM [203]. StringEquiv is a baseline of OAEI that generates alignment basing on exact string matching of concepts' labels. They are classified in Table 5.1 decreasingly according to their F-measure results.

[15] https://www.ncbi.nlm.nih.gov/mesh/
[16] http://www.obofoundry.org/ontology/aeo.html
[17] http://www.obofoundry.org/ontology/caro.html
[18] http://oaei.ontologymatching.org/2018/results/anatomy/index.html
[19] http://oaei.ontologymatching.org/2019/results/anatomy/index.html
[20] http://oaei.ontologymatching.org/2020/results/anatomy/index.html

System	Standard Evaluation Measures			Runtime(s)
	Precision	Recall	F-Measure	
ATBox	0.987	0.671	0.799	192
DeepOM	**0.994**	**0.665**	**0.797**	**149**
OntoConnect	0.996	0.665	0.797	248
ALOD2Vec	0.996	0.648	0.785	75
FCAMap-KG	0.996	0.631	0.772	25
DOME	0.997	0.615	0.761	22
DESKMatcher	0.472	0.623	0.537	391
Holontology	0.976	0.294	0.451	265
AGM	0.152	0.195	0.171	628
StringEquiv	0.997	0.622	0.766	6

Table 5.1. Evaluation results of DeepOM for OAEI-Anatomy'20 track.

As the aim, by DeepOM, is to both maximize the matching quality and minimize the large-scale matching complexity, we evaluate the proposed ontology matching system at two stages:

5.4.2.1. Evaluate the Matching Quality

Graphs in Figure 5.5 outline the evaluation results in terms of the three standard evaluation metrics defined in Sect.1.3.5.3.

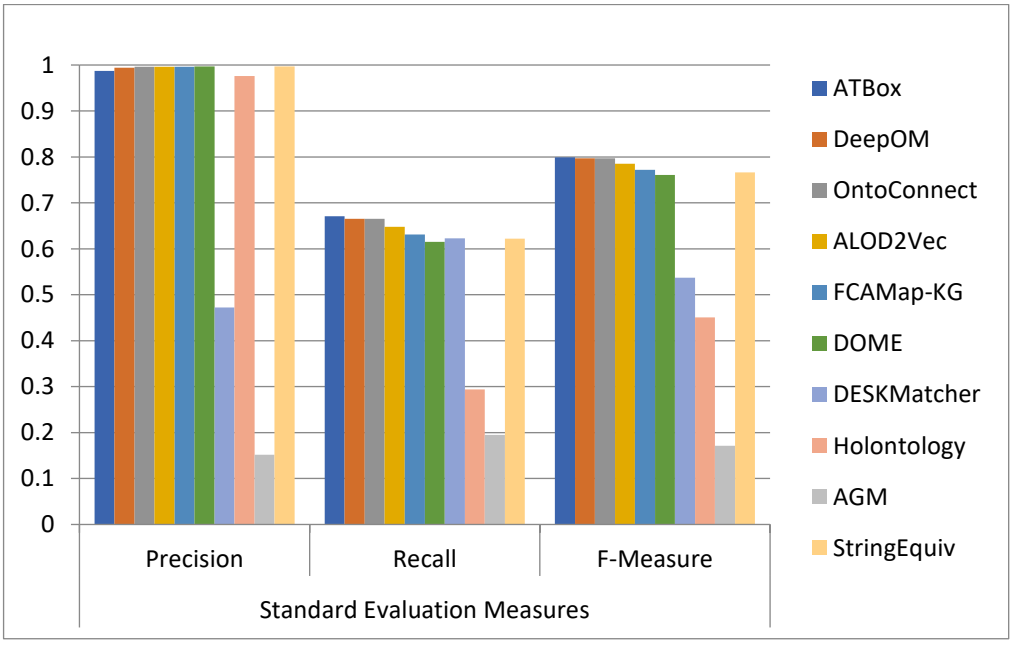

Figure 5.5. Matching Quality evaluation results of DeepOM for OAEI-Anatomy'20 track.

We can see from Figure 5.5 that, 1. DeepOM presents an excellent precision score (0.994). It is among the five top ranked systems of which the values exceed 0.99, and outperforms the other matching systems in terms of precision. 2. Regarding recall, DeepOM achieved a good value (0.665). It presents (with OntoConnect) the second-best score after ATBox with a slight difference of 0.006, and outperforms the other systems. 3. For F-measure, which combines precision and recall evenly, the system proposed in this work presents a high score (0.797), greater than the baseline (StringEquiv) which is based on normalized string equivalence. DeepOM is among the top three ranked systems that exceed 0.79. The other matching systems present competitive results excepting DESKMatcher, Holontology and AGM.

5.4.2.2. Evaluate the Matching Complexity

As we really interest in reducing the matching complexity, we compare the matching runtime required by our system with the other participant matching systems. Figure 5.6 illustrates this comparison.

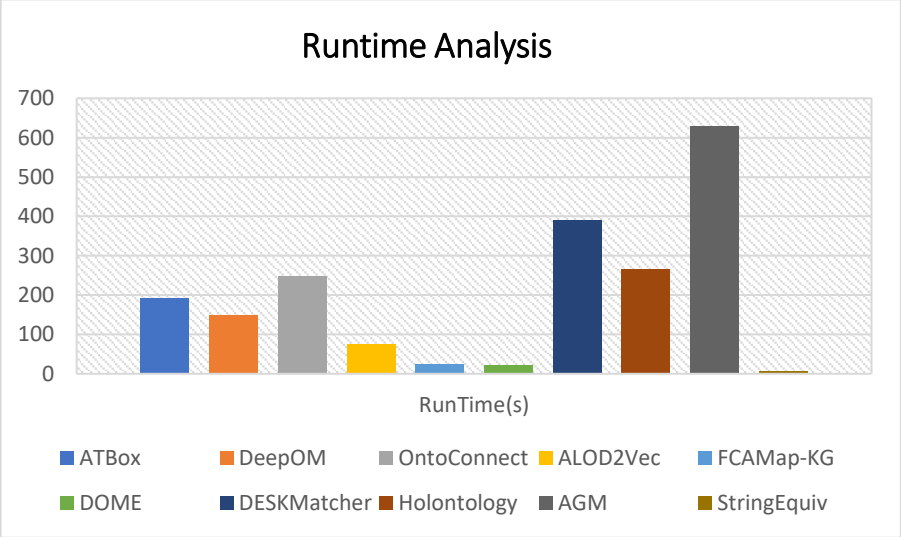

Figure 5.6. Matching Complexity evaluation results of DeepOM for OAEI-Anatomy'20 track.

We can observe from Figure 5.6 that, DeepOM performs the ontology matching process at short notice comparing with the other matching systems. It is among the 4 systems out of 9 that are able to achieve the matching task in less than 150 seconds. These are DeepOM, ALOD2Vec, FCAMap-KG and DOME which has the shortest runtime.

5.4.3. Experimental Summary

According to the previous results, we can conclude the following. As we aim by DeepOM to both increase the matching quality and decrease the matching complexity, we analyse results of F-measure, which reflects the real quality of matching, with respect to the matching runtime.

- The preliminary results of DeepOM are very promising. It has achieved a score of 0.8 for F-measure, which is a very good value for the ontology matching task at least as a start.
- Comparing DeepOM with different matching systems which have participated to the OAEI-Anatomy track, it outperforms them in terms of F-measure, excluding ATBox and OntoConnect.
- As for these two systems, DeepOM has matching results similar to OntoConnect. The results of ATBox are a bit higher but with a very slight difference (0.002). Regarding complexity, DeepOM has the shortest runtime.

The conducted evaluation of DeepOM demonstrates that, concepts' embeddings using the reference ontology and learning them with auto-encoder have improved the performance of the ontology matching task. Moreover, representing ontological concepts by numerical values in a vector space has efficiently solved the large-scale ontology matching problem.

The experimental results of DeepOM show a very good matching performance at lowest cost. This means that we have achieved the two objectives of large-scale ontology matching, which are improving the matching performance and reducing its complexity. Therefore, we have effectively, by DeepOM, tackled the challenges that have motivated this study.

5.5. Conclusion

In this chapter, we have presented DeepOM, a large-scale ontology matching system that we propose basing on deep learning techniques to deal with the large-scale heterogeneity problem. We describe the proposed matching system and its contributions. And, we present the experimental procedure that we have performed on the Anatomy track from the 2020 campaign of the OAEI Initiative.

The key novelty of DeepOM is to use a reference ontology to create semantic embeddings for ontological concepts, which are used to train an auto-encoder in order to learn more accurate and less dimensional representations for concepts. The results of its evaluation on large OAEI ontologies show that, DeepOM has proven its efficiency against the participant matching systems to effectively match large-scale ontologies and tackle the challenges which have motivated this work.

In the next chapter, we present another solution to the large-scale ontology matching issue, which represents the global methodology of this research.

Chapter 6

Matching Big Ontologies Semantically by Combining NeuralOM and DeepOM

Contents

6.1. Introduction

Large-scale ontology matching is still challenging for its large-memory consumption and long-time processing. In the previous chapters of the contribution part, we have proposed two different solutions for the issue of matching heterogeneous and large ontologies.

In this chapter, we propose the global methodology of this research, aiming for addressing the challenges of the existing large-scale ontology matching systems as well as our proposed solutions. We first provide an overview of this matching methodology and discuss its main contributions. Next, we describe its detailed workflow. Then, we present its evaluation that we have conducted on the Anatomy track from the most recent campaign of the OAEI Initiative. We present the results of these experiments and discuss the performance of our matching methodology against OAEI matching systems as well as our previously proposed solutions.

6.2. SemBigOM Overview

In this chapter, we present SemBigOM, the solution that we propose for tackling the challenges of large-scale ontology matching and overcoming the limits of the solutions that we have previously proposed; NeuralOM and DeepOM.

Comparing NeuralOM with DeepOM reveals the following conclusions:

- They are both original methods which respond to the objectives of this study. They address the challenges of ontology matching, particularly large-scale ontology matching and without partitioning. Their evaluation results show high quality and low complexity of matching.
- NeuralOM is marked by its excellent matching results. However, it requires initial mappings as input, so it is related to mappings of other matching systems.
- DeepOM is totally independent. Its evaluation results are very satisfying but poorer than the results of NeuralOM.

As we seek for ideal matching results, we look forward to benefit from the advantages of NeuralOM and DeepOM, and to handle the drawbacks of both of them. In other words, we attempt to achieve excellent matching results independently. Therefore, we combine the two proposed solutions in order to get the global methodology of this research, ***SemBigOM***,

aiming at a better solution which addresses the challenges of large-scale ontology matching and achieves all objectives of this study.

The core idea of this conjunction is that, it makes use of DeepOM to generate initial mappings that NeuralOM requires as input to be reused so as to provide the final matching results. More expressively, we use different versions of DeepOM instead of different ontology matching systems in order to provide the initial alignments that NeuralOM needs as input. In other words, we refine the matching results of DeepOM by NeuralOM so as to perfectly and independently achieve the large-scale ontology matching process. Figure 6.1 presents an overview of the proposed matching methodology.

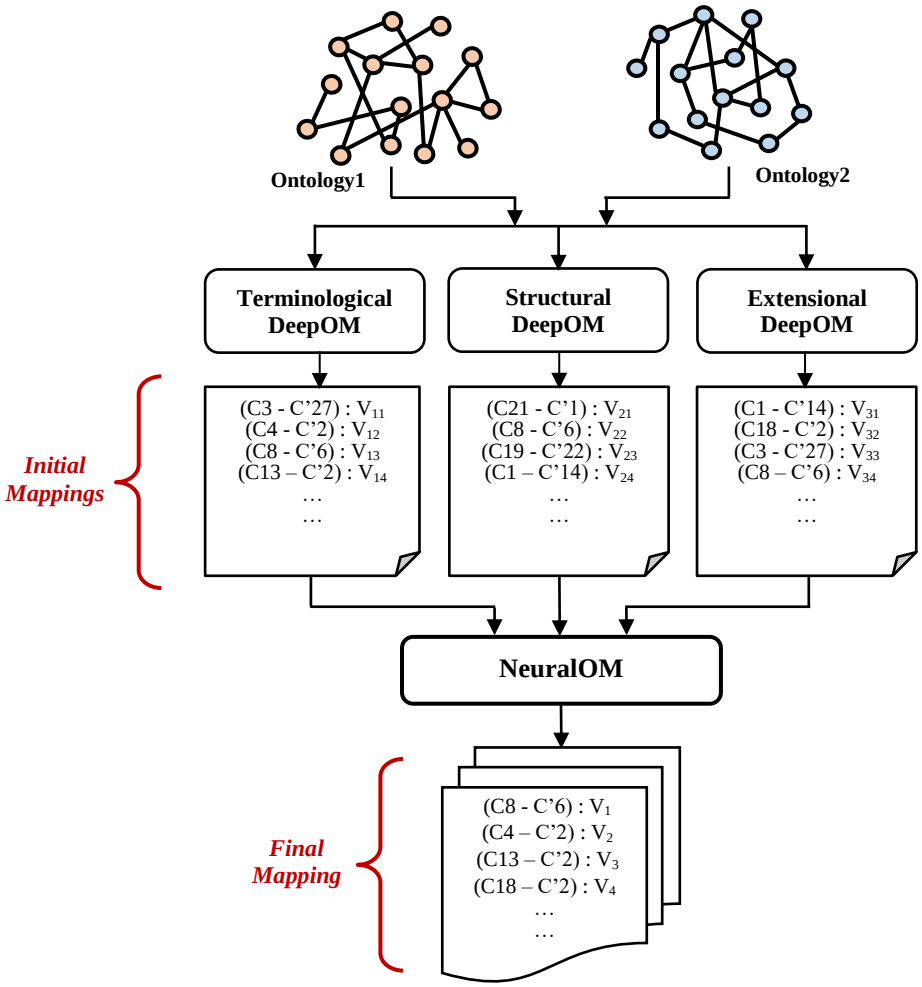

Figure 6.1. SemBigOM Overview.

SemBigOM consists of extracting three different matchers from DeepOM, to output three different mappings that are reused by NeuralOM in order to generate the final mapping. These initial mappings should be:

- *All requisite*, otherwise, some of them could be neglected;

- *Independent*, otherwise, they could cover each other and they could not work in parallel;
- *Have the same importance*, otherwise, their initial weights would not be equivalent.

The ontological concept is the basic unit of an ontology. It is defined by a semantic triangle with three dimensions: *terminological dimension*; *structural dimension* and *extensional dimension* (more details can be found in Sect.5.3.1.1). Working on these aspects exploits semantics inside ontologies in one hand. In the other hand, it leads to a high quality of matching since the three dimensions are precise and complementary. With regard to the matching evaluation measures, a high precision and a high recall produce a high f-measure. Thus, they lead to a correct and complete alignment.

Ontology matching is based on computing similarity between ontologies. On this basis, SemBigOM first generates three different mappings between input ontologies by three matchers using three different versions of DeepOM: Terminological-based DeepOM, Structural-based DeepOM and Extensional-based DeepOM.

The different matchers differ in measuring similarity for representing concepts of ontologies in a numerical multi-dimensional vector space. A reference ontology is used for that purpose. Next, an auto-encoder is trained in order to transform the produced representations into finer and smaller numerical vectors for concepts. Then, final similarities between the compact vectorial representations of concepts are computed using the cosine similarity and revised by a pruning process.

After that, the three generated mappings are refined by NeuralOM in order to provide an ideal mapping between input ontologies. They are combined to construct the matching dataset which is used, first through a linear perceptron to adjust an importance weight for each matcher, then to define the matching function which leads to generate the final alignment after a threshold filtering procedure.

SemBigOM benefits from the advantages of both NeuralOM and DeepOM and addresses their limits. It provides the following contributions:

- Exploiting semantics inside the concerned ontologies. SemBigOM benefits from covering all aspects of the concepts of input ontologies. In addition, it exploits background knowledge resources in ontology matching, and uses an already trained

deep learning model for measuring the required similarities. Thus, SemBigOM is advantaged of providing a correct mapping.

- It deals with the large-scale ontology matching issue at two stages. At each stage, it aims for perfecting and simplifying the ontology matching process.
    - First, SemBigOM involves deep learning methods, which are very appropriate for treating huge amounts of data, in order to create semantic embeddings for concepts of input ontologies. This is basing on computing similarities with concepts of a smaller and well selected reference ontology. That perfects the matching process, because of the fact that, each concept is represented in a vector space of a wide number of dimensions, where each value adds more precision to the concerned concept. And, it reduces the matching complexity, since, manipulating vectors of real values instead of ontological concepts is much less complicated.

      Then, an auto-encoder is trained on the generated vectors in order to provide better matching performance, since the auto-encoder keeps the most representative values for each concept. Also, training the model and compressing concepts embeddings to a lower-dimensional vector space decreases the large-scale matching complexity.
    - Second, as we seek for an ideal ontology matching, we perform a refining procedure that combines the different mappings generated by SemBigOM in order to provide an ideal alignment, using artificial neural networks which are, with their structures of numerous inputs and outputs, very appropriate for combination. This procedure serves to perfect the quality of the matching process with negligible matching complexity.
- Supporting parallelization, which emerges as a complementary solution for matching large-scale ontologies. The different matchers of SemBigOM works in parallel. Moreover, SemBigOM parallelizes any multiple tasks that can run in parallel, aiming for reducing the processing time of large-scale ontology matching.

6.3. Semantic Big Ontology Matching

In this section, we study ***SemBigOM***, the research framework that we propose for addressing the ontology matching challenges at the large scale. The processing workflow of SemBigOM is illustrated in Figure 6.2. It could be summarized in the following phases: *Pre-Matching*

Phase, Deep Embedding Phase, Matching Phase, Mapping Reuse phase and *Post-Matching Phase.*

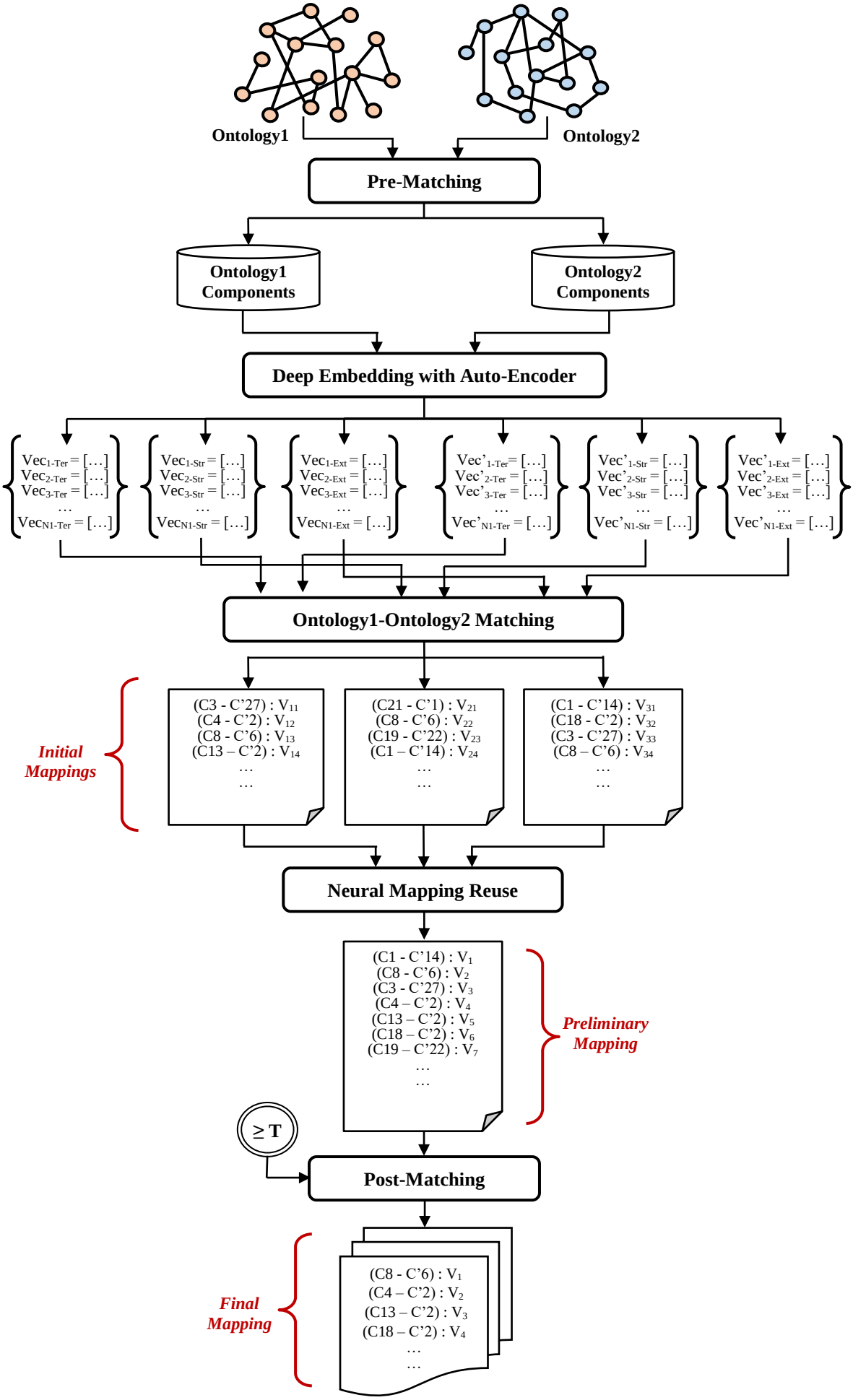

Figure 6.2. Processing Workflow of SemBigOM.

SemBigOM takes as input two given ontologies, *Ontology1* and *Ontology2*, and outputs an alignment A between them. A is a set of correspondences defined each by an identifier, a concept source (from *Ontology1*), a concept target (from *Ontology2*) and a similarity value. In what follows, we describe the matching process of SemBigOM and discuss its detailed steps.

6.3.1. Pre-Matching

At first, SemBigOM pre-matches input ontologies aiming for preparing them for matching. This preliminary phase consists of extracting the required information from *Ontology1* and *Ontology2*. i.e., we extract their components which are necessary for generating alignment.

As previously defined (Sect.5.3.1.1), an ontological concept is defined by three main aspects: ***Term***, ***Intention*** and ***Extension***. Respecting this 3-dimensional shape, we extract for each concept from *Ontology1* and *Ontology2* its:

- ***Lexical label***
- ***Related concepts*** basing on the subsumption relationship, that are:
    - Its ***Ancestors*** **(**parent-concepts along the path to root**)**
    - Its ***Descendants*** **(**child-concepts of the first level**)**
    - Its ***Siblings*** **(**direct child-concepts of the direct parent-concepts**)**
- ***Individuals***

After that, we pre-process the lexical information of the extracted components. i.e., we process the *label*, *related concepts' labels* and *individuals' labels* of each concept. The pre-processing procedure outputs a set of processed terms. It consists mainly on ***tokenizing***, ***denoising*** and ***removing stop words*** from the extracted textual information.

6.3.2. Deep Embedding of Concepts

This phase aims for preparing the concepts of input ontologies for deep learning. It transforms their concepts into vector representations that deep learning models can use. It is illustrated with the next phase in Figure 6.3. This transformation is performed using a **reference** ontology, which should be of the same domain as input ontologies, semantically and neutrally close to them, and of an appropriate size.

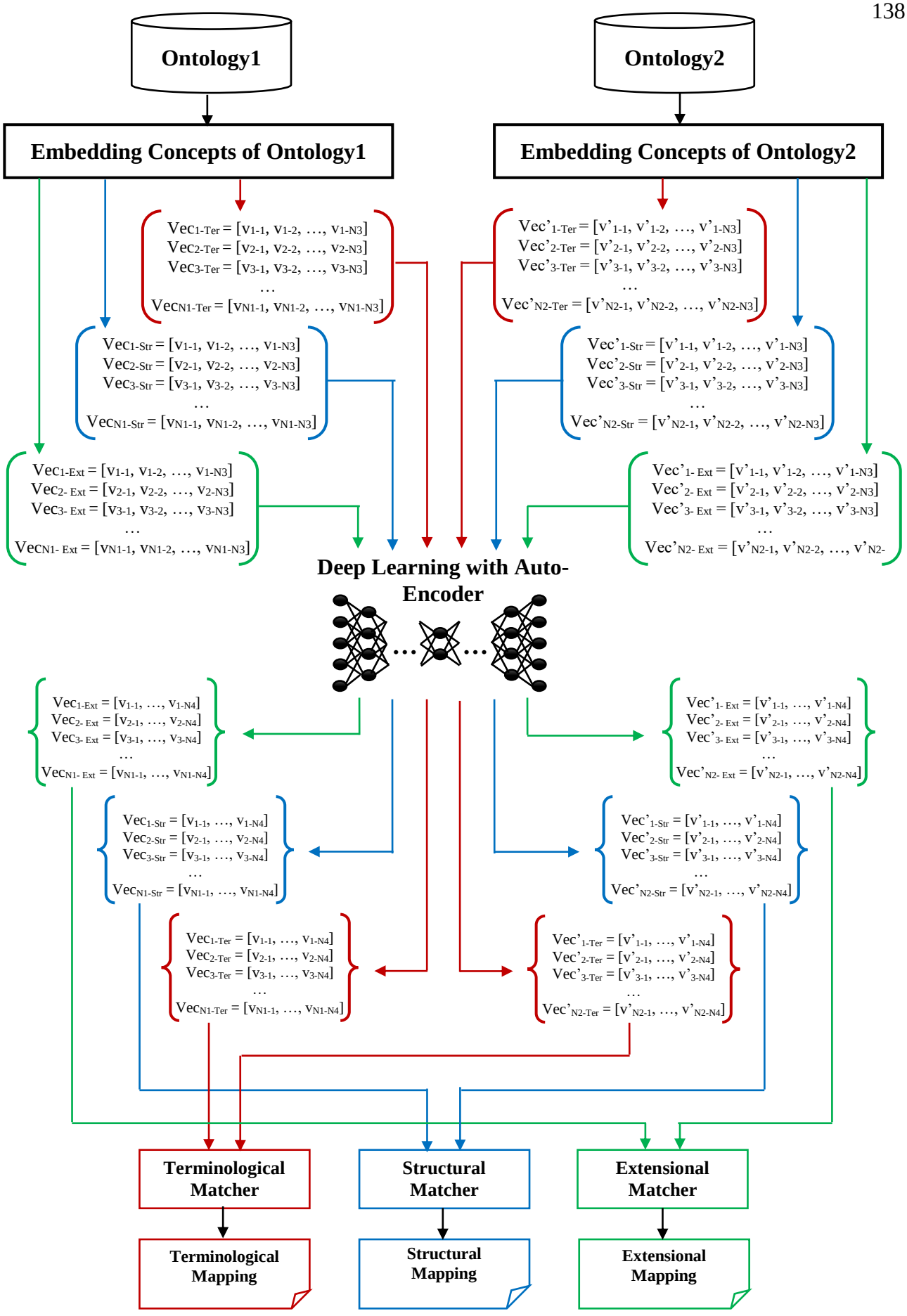

Figure 6.3. Creating Semantic Embeddings and Deep Matching of Concepts.

In order to represent a given concept from *Ontology1* and *Ontology2* in such a vector space, we compute similarities between that concept and all concepts of the reference ontology. N1, N2 and N3 denote supposedly the size of *Ontology1*, *Ontology2* and the reference ontology respectively. Each input concept is then represented by an N3-dimensionnal vector, where each dimension relies on a concept from the reference ontology.

As we interest in covering the three aspects of the ontological concept, we represent each concept from *Ontology1* and *Ontology2* by three different vectors. That produces three embedding sets for each input ontology:

- ***Terminological embedding:*** is based on computing the terminological similarity using the formula (5.1) proposed in chapter 5, combining a *context-based* similarity using an *external knowledge resource*, and a *syntactical* similarity using *Jaro measure*.
- ***Structural embedding:*** is based on measuring the similarity between concepts basing on their related concepts. For that, we use the formula (5.2) presented in chapter 5.
- ***Extensional embedding:*** is based on computing the similarity between instances of concepts using the *Jaccard similarity*.

Once the six embedding sets are created, they undergo an unsupervised deep learning procedure in order to provide *less dimensional* and *more accurate* representations of concepts. Auto-encoders, with their mirrored *Encoder-Decoder* architecture, seem to be very appropriate for such tasks. We train an auto-encoder model to compress down the resulted vectors, and represent them into a lower dimension. The learned representations keep the most important features of concepts. The auto-encoder parameters are fixed by trial-and-error.

6.3.3. Ontology1-Ontology2 Matching

This phase (Figure 6.3) consists on generating alignment between Ontology1 and Ontology2. That requires computing similarities between their concepts. Since these concepts are represented by three various vector representations, the matching process consists on measuring the embedding similarity between these vectors. For that, the cosine similarity measure is used by three different matchers: a ***terminological matcher***, a ***structural matcher*** and an ***extensional matcher***, in order to compute the similarity between the two terminological embeddings, structural embeddings and extensional embeddings respectively.

As results of this step, the three different matchers generate three different mappings, which undergo subsequently a filtering procedure using a defined threshold. That aims for extracting the final mappings by keeping only the most significant correspondences of high similarity scores.

6.3.4. Neural Mapping Reuse

This phase is illustrated by Figure 6.4. It is performed in three main steps.

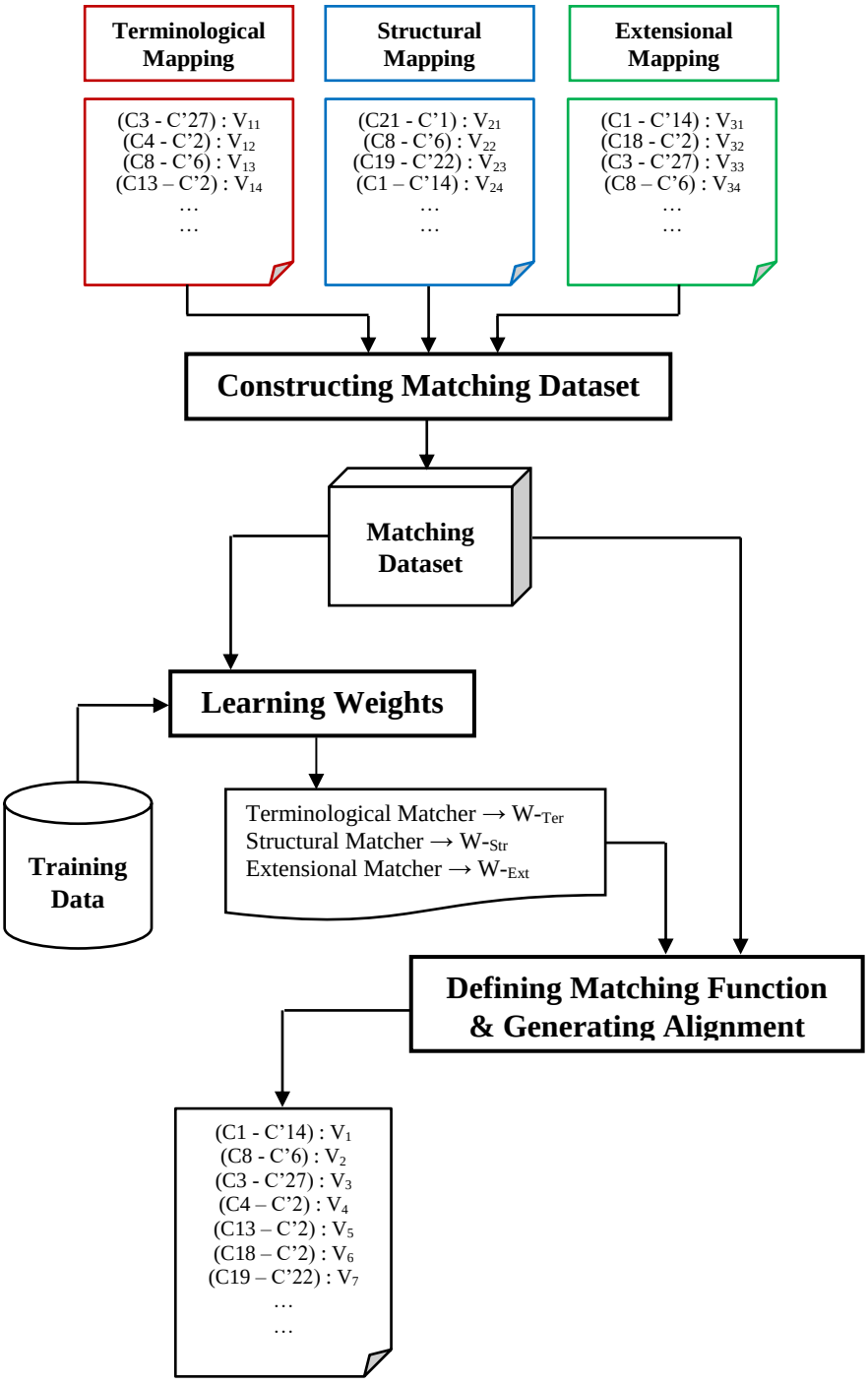

Figure 6.4. Neural Mapping Reuse.

Step 1 - *Constructing Matching Dataset*

The first step consists on constructing the matching dataset from the three mappings previously generated between *Ontology1* and *Ontology2*. Each mapping has been created by a different matcher using its own specific matching technique.

The method of their combination has a big impact on the final alignment results. It determines the size and components of the matching dataset. As we aim at both maximizing the number of relevant alignments and minimizing the number of un-related ones, we worked on three completeness levels so as to conclude the best strategy for producing the dataset:

- ***Intersection:*** consists of keeping only common alignments pairs between chosen mappings.
- ***Majority:*** consists of taking alignments pairs generated by the majority of the matching tools.
- ***Union:*** consists of taking alignments pairs for which at least a chosen matching system has given a value.

Step 2 – *Training Neural Network*

The second step consists of training a neural network in an unsupervised way, so as to learn the matching function that leads to generate correspondences between input ontologies.

We aim by this procedure at fixing, for each matcher, a weight which reflects its importance. Thus, we train a simple neural network of 3 inputs and one output for each pair of concepts from the training dataset. Inputs correspond to the different matchers, and the output, which is obtained from OAEI, represents the pretended similarity score between the two concepts. The initial weights are first initialised to 1. Then, they are updated for each sample of the training dataset, applying the back-propagation learning method, and using Sigmoïd as activation function. The other neural network parameters (like the learning rate) are fixed by trial-and-test.

Step 3 – *Matching Ontologies*

After that the final weights are adjusted for the three different matchers, this third step concerns matching input ontologies. It defines the matching function that allows generating semantic correspondences between *Ontology1* and *Ontology2*, using the matchers weights

as well as their initial mappings. The output similarity between each pair of concepts from input ontologies is computed using formula (4.3) proposed in chapter 4.

6.3.5. Post-Matching

Finally, we prune and filter the generated correspondences so as to get the final mapping of SemBigOM. For that purpose, we define an alignment threshold T in order to keep only significant correspondences of which the value exceeds T. For instance, a pair of input concepts with a zero-similarity value is not considered as a valid correspondence. T is fixed by trial-and-error.

Our aim behind this phase is to improve the precision of the matching process. We seek for eliminating irrelevant correspondences of low similarities, and keeping only the most appropriate ones.

6.4. Evaluation Strategy

In this section, we present the experimental strategy that we proceed for evaluating the ontology matching solutions that we propose in this study for dealing with the large-scale ontology matching issue.

6.4.1. Experimental Settings

For evaluating the proposed methodology, we use the standard evaluation measures: precision, recall and F-measure, against the reference alignments. They are previously defined and described in (Sect.1.3.5.3). These metrics are the most widely used criteria for such evaluation. They are based on the comparison of the resulted alignment *A* against a reference alignment *R*. Precision and recall are inversely proportional. For that, their harmonic mean, F-measure, is also commonly used.

For SemBigOM embedding tasks, we use **BioWordVec [199]** as an external resource for measuring the required similarity. In addition, the Anatomical Entity Ontology: **AEO**[21] is used as reference ontology. It is an OWL ontology of anatomical structures highly adopted for facilitating interoperability across existing anatomy ontologies.

As for deep learning tasks, the structure of the trained auto-encoder model is fixed by trial and test. Thus, the size of the embedded representations, input and output, to the auto-

[21] http://www.obofoundry.org/ontology/aeo.html

encoder model is 250 for each ontological concept. It is basically impacted by the size of the reference ontology. The size of the compressed representation of the concept is 50.

Concerning the construction of the matching dataset of the neural refining phase (Sect.6.3.4), we worked on three completeness levels and compared their results so as to conclude the best way for constructing the dataset: ***intersection***, ***majority*** and ***union***. In the following, we refer by ***SemBigOM-I***, ***SemBigOM-M*** and ***SemBigOM-U*** to our semantic ontology matching methodology by *intersection*, *majority* and *union* strategy of dataset construction respectively.

In order to efficiently control the neural network while training and testing, we adopt a cross-validation procedure. It consists of partitioning the training data into P sets of equal size. The algorithm is run P times. For each time, the corresponding partition is used for testing, where the rest of the dataset is used for learning. The global validation result is the average of the individual validation results of the independent partitions. In SemBigOM cross-validation procedure, the number of cross-validation partitions, is fixed by trial and test.

For the deep matching tasks, we use **Google Colab**[22], a product from Google Research allowing to write and executing arbitrary python codes through the browser. It supports various popular machine learning libraries, mathematical equations, external datasets and free Cloud services with free GPU and TPU resources. For the neural refining tasks, the evaluation was carried out on a Windows8 (64-bit) desktop with an Intel-Core i5-3210M CPU @ 2.50GHz allocating 4.00 GB of RAM.

Ontology Alignment Evaluation Initiative (OAEI)[23] is an international initiative which aims for evaluating ontology matching systems using diverse types of test ontologies. It is the most authenticated initiative in this scope. It offers various test sets aiming to compare the different participant systems on the same basis in order to identify their advantages and limits. Since 2004, OAEI yearly organizes new campaigns and sections, and introduces new challenges for evaluation.

In this study, we evaluate our proposed solutions according to various OAEI Campaigns. Particularly, we evaluate SemBigOM according to the Anatomy track[24] of the

[22] http://research.google.com/colaboratory/
[23] http://oaei.ontologymatching.org/
[24] http://oaei.ontologymatching.org/2022/anatomy/index.html

most recent OAEI campaign (OAEI'2022). We look for using challenges of different domains, types and especially of different and huge sizes. Therefore, seven OAEI tracks are selected. Four of them comprise multiple test cases that we consider as sub-tracks, since they are complete and independent in their datasets and results. All in all, we set up our experiments of this work on twelve test cases from OAEI'2015, OAEI'2017, OAEI'2018, OAEI'2019, OAEI'2020 and OAEI'2022 campaigns. We describe below the experimental design of each test case.

1) OAEI'2018-CONFERENCE Track

The goal of this track[25] **(Conference)** is to find alignments within a collection of 16 ontologies describing the domain of organising conferences. These ontologies, developed within OntoFarm project, are suitable for ontology matching task because of their heterogeneous character of origin. Tests are performed on a suite of 21 matching tasks of the pairwise combination of seven moderately expressive ontologies describing the same domain. Participant systems results had been evaluated based on crisp reference alignment, its uncertain version and logical reasoning evaluation based on violations of conservative principle. These ontologies which are shortly described in Table.6.1.

Ontology name	# of classes	Ontology name	# of classes	Ontology name	# of classes	Ontology name	# of classes
Ekaw	74	Micro	32	ConfTool	38	Paperdyne	47
Sofsem	60	Confious	57	Crs	14	Edas	104
Sigkdd	49	Pcs	23	Cmt	36	MyReview	39
Iasted	140	OpenConf	62	Cocus	55	Linklings	37

Table 6.1. Overview of OAEI Conference test ontologies.

2) OAEI'2018/2020/2022-ANATOMY Track

The anatomy track[26/27/28] **(Anatomy)** comprises a single real-world test case about matching two fragments of biomedical ontologies describing the human anatomy with 3304 classes and the anatomy of the mouse with 2744 classes. Ontologies are large, carefully designed

[25] http://oaei.ontologymatching.org/2018/conference/index.html
[26] http://oaei.ontologymatching.org/2018/anatomy/index.html
[27] http://oaei.ontologymatching.org/2020/anatomy/index.html
[28] http://oaei.ontologymatching.org/2022/anatomy/index.html

and described in technical terms. The evaluation is based on a manually curated reference alignment.

3) OAEI'2018-DISEASE AND PHENOTYPE Track

This track[29] **(Phenotype)** is organized and sponsored by The Pistoia Alliance Ontologies Mapping project team, basing on a real use case that requires finding alignments between disease and phenotype ontologies. The track comprises two tasks:

3.1) HP-MP

It consists on matching the Human Phenotype (HP) Ontology (31034 classes) and the Mammalian Phenotype (MP) Ontology (30273 entities) ontologies, against 696 reference alignments.

3.2) DOID-ORDO

It consists on matching the Human Disease Ontology (DOID) (38240 classes) and the Orphanet and Rare Diseases Ontology (ORDO) (13504 entities) ontologies, against 1237 reference alignments.

4) OAEI'2018-LARGE BIOMEDICAL ONTOLOGIES Track

The large biomedical ontologies track[30] **(Large-BioMed)** consists on finding alignments between the Foundational Model of Anatomy (FMA), the Systematized Nomenclature Of MEDicine-Clinical Terms (SNOMED-CT), and the National Cancer Institute Thesaurus (NCI). These ontologies are semantically rich and contain 78989, 306591 and 66724 classes, respectively. The track comprises three matching problems:

4.1) FMA-NCI

This challenge comprises 3024 equivalence correspondences

4.2) FMA-SNOMED

This challenge comprises 9008 equivalence correspondences

4.3) SNOMED-NCI

This challenge comprises 18844 equivalence correspondences

[29] http://sws.ifi.uio.no/oaei/phenotype/
[30] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/

5) OAEI'2018-BIODIVERSITY AND ECOLOGY Track

The goal of this track[31] **(BioDiv)** is to find pairwise alignments between four ontologies being used in various projects and particularly useful for biodiversity and ecology research. They are semantically rich and very overlapping. The reference alignments are produced using established matching systems to produce an automated consensus alignment, and then manually validating the unique results produced by each system, and finally adding manually generated correspondences. The track features two challenges:

5.1) FLOPO-PTO

This challenge consists on finding alignments between the Flora Phenotype Ontology (FLOPO) (24199 classes) and the Plant Trait Ontology (PTO) (1504 classes).

5.2) ENVO-SWEET

This challenge consists on finding alignments between the Environment Ontology (ENVO) (6909 classes) and the Semantic Web for Earth and Environment Technology Ontology (SWEET) (4543 classes).

6) OAEI'2017-PROCESS MODEL MATCHING Track

This track[32] **(PM)** is a spinoff from the Process Model Matching Contest. It is concerned with the task of matching process models, originally represented in BPML. These models have been converted to an ontological representation. The resulting matching task is a special case of an interesting instance matching problem. The track comprises two test cases:

6.1) UNIVERSITY ADMISSION

The dataset of this task consists of nine process models that describe the process of university admission for different German universities. The BPMN representation of the process models was converted to a set of assertions (ABox) using the vocabulary defined in the BPMN 2.0 ontology (TBox).

6.2) BIRTH REGISTRATION

The dataset of this task consists of process models that describe the process of registering a new-born child in different countries and related administrative tasks. These process models were originally available as Petrinets as *.pnml. These datasets have also been converted into ontologies, more precisely into ABoxes.

[31] http://oaei.ontologymatching.org/2018/biodiv/index.html
[32] http://web.informatik.uni-mannheim.de/oaei/pm17/

7) OAEI'2015-ONTOLOGY ALIGNMENT FOR QUERY ANSWERING Track

The goal of the OA4QA track[33] **(OA4QA)** is to measure the ability of generating alignments to answer a set of queries in an ontology-based data access scenario where several ontologies exist. The dataset is based on the Conference track, and extended with synthetic Aboxes extracted from the DBLP dataset. The reference answer set used is the publicly available reference alignment of the Conference track and a manually repaired version of it from conservativity and consistency violations.

6.4.2. Experimental Results

In this sub-section, we present the experimental results of SemBigOM evaluation. For better evaluating our research methodology, we perform our tests against the most recent OAEI systems as well as against our previously proposed solutions. Therefore, we conducted our experiments in two main sketches: *evaluation against OAEI systems* and *evaluation against NeuralOM and DeepOM*. Furthermore, as we really interest in both matching quality and matching complexity, we analyse, for each sketch, the standard evaluation measures: Precision, Recall, F-measure as well as the matching runtime required by SemBigOM compared to all competing matching systems.

6.4.2.1. First Sketch - Evaluation against OAEI'2022 Systems

In this section, we present the results of our experimental procedure of SemBigOM with the results[34] of the ontology matching systems participating at the anatomy track of the 2022 OAEI campaign (Anatomy'22 track).

The results of the performed evaluation are summarized in Table 6.2. The best scores are marked in bold for each evaluation measure. SemBigOM is compared to the following OAEI systems: Matcha [204], SEBMatcher [205], LogMapBio [206], LogMap [206], AMD [207], ALIN [208], LogMapLite [206], ATMatcher [209], LSMatch [210], ALIOn [211]. StringEquiv is a baseline of OAEI which produces the alignment between input ontologies basing on exact string matching of labels of concepts. These systems are classified decreasingly in Table 6.2 according to their F-measure scores.

[33] http://oaei.ontologymatching.org/2015/oa4qa/index.html
[34] http://oaei.ontologymatching.org/2022/results/anatomy/index.html

Ontology Matching System		Standard Evaluation Measures			Runtime(s)
		Precision	Recall	F-Measure	
SemBigOM	***Intersection***	0.980	**0.996**	**0.987**	66.276
	Majority	0.954	0.995	0.974	66.323
	Union	0.954	0.889	0.919	66.307
OAEI Systems	Matcha	0.951	0.930	0.941	37
	SEBMatcher	0.945	0.874	0.908	35602
	LogMapBio	0.873	0.919	0.895	1183
	LogMap	0.917	0.848	0.881	9
	AMD	0.953	0.817	0.880	160
	ALIN	0.984	0.752	0.852	374
	LogMapLite	0.962	0.728	0.828	**3**
	ATMatcher	0.978	0.669	0.794	156
	LSMatch	0.952	0.634	0.761	20
	ALIOn	0.364	0.460	0.407	26134
	StringEquiv	**0.997**	0.622	0.766	-

Table 6.2. Evaluation results of SemBigOM against OAEI'2022 systems for Anatomy track.

As stated above, we evaluate the proposed ontology matching solution at two stages, aiming at both maximizing the matching quality and minimizing the large-scale matching complexity.

6.4.2.1.1. Matching Quality Evaluation

Figure 6.5 outlines the evaluation results in terms of the three standard evaluation measures defined in the precedent sub-section. It summarises the comparison of SemBigOM results with those of all participant matching systems for OAEI'2022 Anatomy track.

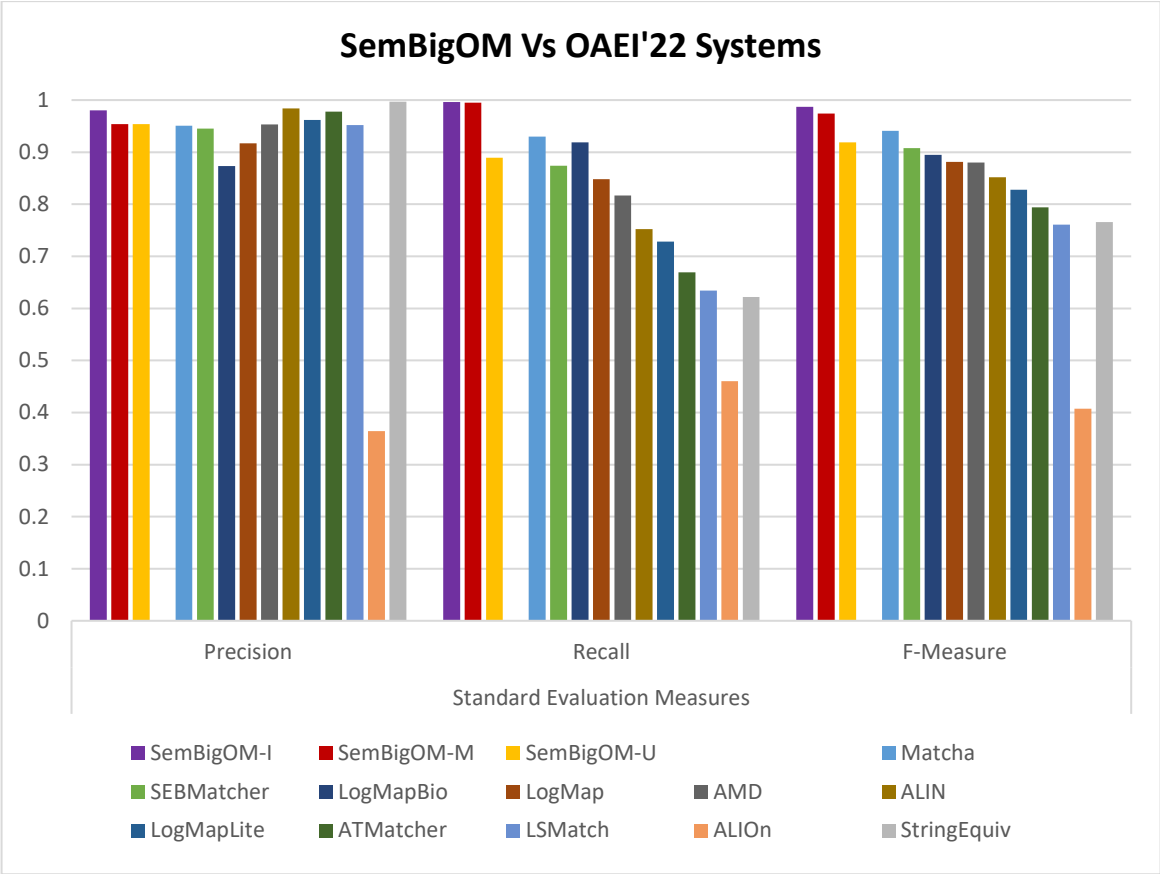

Figure 6.5. Matching quality evaluation results of SemBigOM against OAEI'22 systems for Anatomy track.

From Figure 6.5, we can observe that:

- SemBigOM achieves excellent scores of Precision by its three variants. It gives values of 0.980, 0.954 and 0.954 by SemBigOM-I, SemBigOM-M and SemBigOM-U respectively. They are so closed to the score of StringEquiv which is equal to 0.997, and competitive to ALIN, AMD, LogMapLite, ATMatcher and LSMatch with scores equal to 0.984, 0.953, 0.962, 0.978 and 0.952 respectively.
- As for Recall, SemBigOM gives very significant values of 0.996, 0.995 and 0.889 by its version of intersection, majority and union respectively. They are more elevated than StringEquiv value which is equal to 0.622. The results of SemBigOM-I and SemBigOM-M exceed those of all matching systems. SemBigOM-U has been exceeded only by Matcha and LogMap with scores of 0.930 and 0.919 respectively.
- As expressed by F-measure which presents a balance between the two previous measures, SemBigOM-I and SemBigOM-M achieved by far the best results of 0.987 and 0.974 in comparison with the other systems. SemBigOM-U has an excellent

score as well. It is equal to 0.919 and slightly exceeded only by Matcha with a score of 0.941. The results of the proposed methodology are strongly higher than StringEquiv result which is equal to 0.766.

- The proposed solution achieves the best scores. SemBigOM-I gets the best performance, slightly better than SemBigOM-M, followed by SemBigOM-U.

6.4.2.1.2. Matching Complexity Evaluation

Figure 6.6 outlines the evaluation results in terms of the matching runtime. It summarises the comparison of SemBigOM results with those of all participant matching systems for OAEI'2022 Anatomy track.

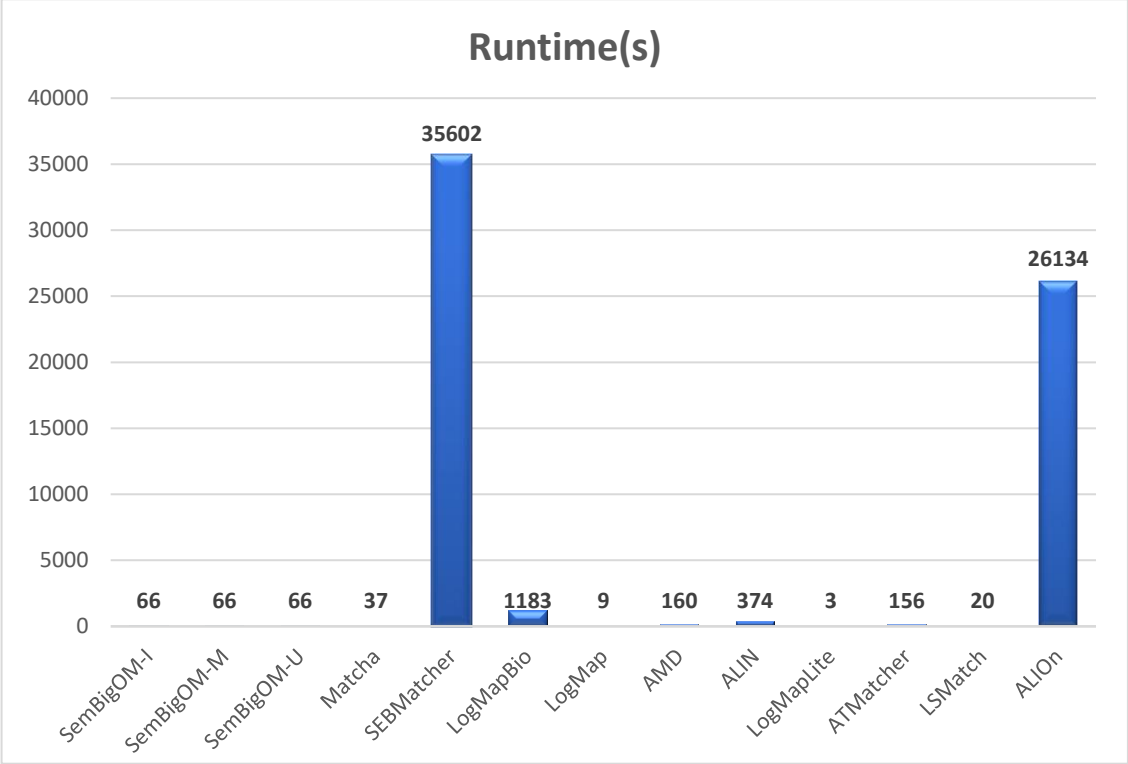

Figure 6.6. Matching complexity evaluation results of SemBigOM against OAEI'22 systems for Anatomy track.

It is clearly seen from Figure 6.6 that, SemBigOM performs the ontology matching process at short notice comparing with the other matching systems. Only four systems; Matcha, LogMap, LogMapLite and LSMatch. which has the shortest runtime, can achieve the matching task before SemBigOM. On the other hand, it is faster than AMD, ALIN, and ATMatcher. Moreover, SemBigOM runtime is roughly smaller than SEBMatcher, LogMapBio and ALIOn which exceed 1000s.

6.4.2.2. Second Sketch - Evaluation against NeuralOM and DeepOM

In this section, we present the results of the performed experimental procedure of SemBigOM with the results of the ontology matching solutions that we propose in this study: NeuralOM and DeepOM. The results of this evaluation, which is performed according to the OAEI anatomy track, are summarized in Table 6.3. The best scores are marked in bold for each evaluation measure.

Ontology Matching System		**Standard Evaluation Measures**			**Runtime(s)**
		Precision	**Recall**	**F-Measure**	
NeuralOM	***Intersection***	0.981	**1.000**	**0.990**	808.219
	Majority	0.951	**1.000**	0.974	808.250
	Union	0.919	0.930	0.923	808.266
DeepOM		**0.994**	0.665	0.797	149
SemBigOM	***Intersection***	0.980	0.996	0.987	**66.276**
	Majority	0.954	0.995	0.974	**66.323**
	Union	0.954	0.889	0.919	**66.307**

Table 6.3. Evaluation results of SemBigOM against NeuralOM and DeepOM for Anatomy track.

6.4.2.2.1. Matching Quality Evaluation

Figure 6.7 illustrates the evaluation results in terms of the three standard evaluation measures previously defined. It summarises the comparison of SemBigOM results with the results of NeuralOM and DeepOM for OAEI Anatomy track.

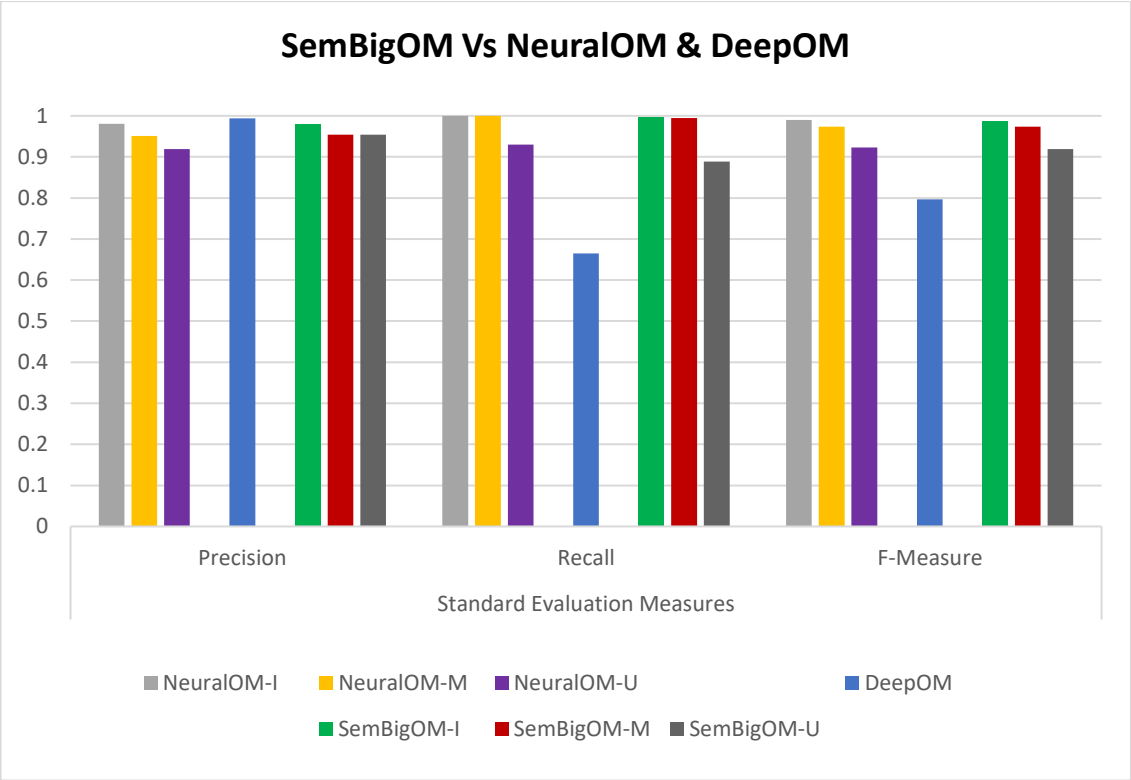

Figure 6.7. Matching quality evaluation results of SemBigOM against NeuralOM and DeepOM for Anatomy track.

It is observed from Figure 6.7 that:

- The best Precision performance belongs to DeepOM with a very high score of 0.994. NeuralOM-I and SemBigOM-I get the next best values equal to 0.98, followed by SemBigOM-M, SemBigOM-U and NeuralOM-M with close results around 0.95. The last score is related to NeuralOM-U which presents an excellent value as well (0.919).
- NeuralOM-I and NeuralOM-M achieve complete scores of Recall (1.0). SemBigOM-I and SemBigOM-U results are not considerably lower with results exceeding 0.99. Then, NeuralOM-U and SemBigOM-U achieve excellent recall values around 0.9. The next score is 0.665 and given by DeepOM.
- As expressed by F-measure which reflects the real quality of matching, NeuralOM-I and SemBigOM-I get the best performance with results of 0.99. The following best score is achieved by NeuralOM-M and SemBigOM-M of 0.974. NeuralOM-U and SemBigOM-U present excellent results as well exceeding 0.91. Lastly, DeepOM gives a very significant score of about 0.8.

- DeepOM results are disparate. It has the highest score of precision and the lowest score of recall. Results of NeuralOM and SemBigOM are more balanced. Their scores are so close and competitive with each other. Both of them present a decreasing order from the intersection to the majority to the union version. However, on average, NeuralOM results are slightly better.

6.4.2.2.2. Matching Complexity Evaluation

Figure 6.8 outlines the evaluation results in terms of the matching runtime required for matching. It summarises the comparison of SemBigOM results with those of NeuralOM and DeepOM for OAEI Anatomy track.

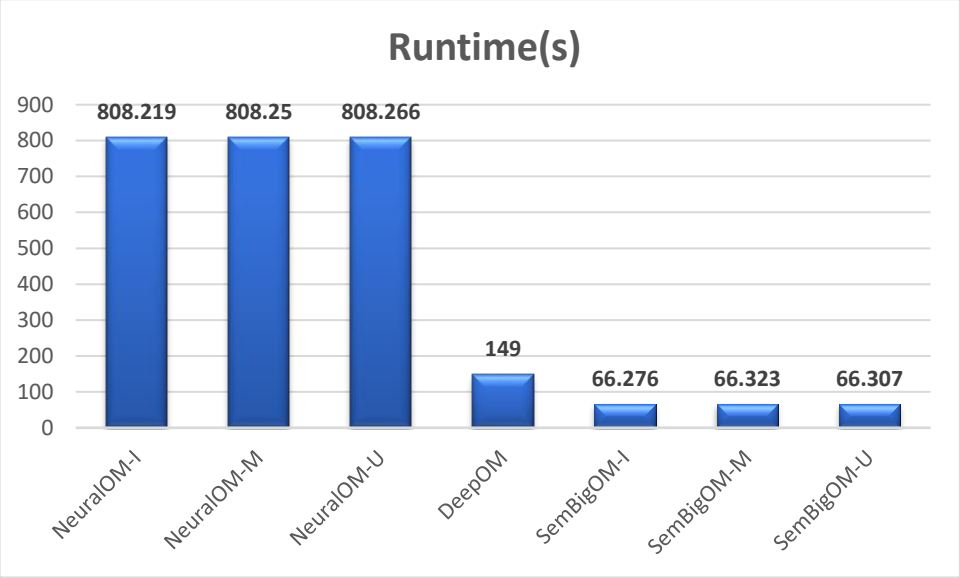

Figure 6.8. Matching complexity evaluation results of SemBigOM against NeuralOM and DeepOM for Anatomy track.

As shown in Figure 6.8, the runtime required by the three versions of NeuralOM (NeuralOM-I, NeuralOM-M and NeuralOM-U) to achieve the matching process is higher than the runtime required by DeepOM. The matching runtime of SemBigOM (SemBigOM-I, SemBigOM-M and SemBigOM-U) is much lower. We should notice that NeuralOM runtime is basically impacted by the runtime required by the input systems (808 is the runtime required by LogMapLite which has the maximal runtime value among input systems).

6.4.3. Synopsis

As we seek for increasing the matching quality and decreasing the matching complexity, we analyse F-measure results (the balanced harmonic average of Precision and Recall) with

respect to the matching runtime. An analytical look at the preceding sub-sections reveals the unfold conclusions:

- The results of the proposed methodology are strongly higher than StringEquiv result. The preliminary results of SemBigOM are very promising. It has achieved F-measure values exceeding 0.9 by its three variants. This is an excellent score for the ontology matching task especially at the large-scale.
- Comparing SemBigOM with the OAEI matching systems participant to the OAEI-Anatomy track of the 2022 campaign shows that, it outperforms them in terms of F-measure. besides the fact that, SemBigOM-U has an excellent score, SemBigOM-I and SemBigOM-M achieved by far the best results and outperforms the other matching systems. Moreover, SemBigOM performs the ontology matching process at short notice comparing with the competing systems.
- It should be noted that, we have tested SemBigOM according to the most recent campaign of OAEI. The Anatomy track is functional yearly since 2004. The competing ontology matching systems have been developed and improved over years through several participations at this initiative. SemBigOM outperforms these systems. Therefore, it is effective to perform the matching process among heterogeneous ontologies.
- The main conclusion revealed from comparing SemBigOM with NeuralOM is that SemBigOM is totally independent of other ontology matching systems. Their matching results are so close and competitive. They present a decreasing order from their intersection to the majority to the union version. The matching runtime required by SemBigOM is much lower than NeuralOM.
- Comparing SemBigOM with DeepOM shows that, the matching results of SemBigOM are much higher than those of DeepOM. In addition, the runtime required by DeepOM to achieve the matching process is higher than the runtime required by SemBigOM. Thus, SemBigOM outperforms DeepOM in terms of both matching quality and complexity.

The conducted evaluation of SemBigOM demonstrates an excellent matching performance at lowest cost. It has perfectly achieved the two objectives of large-scale ontology matching, which are improving the matching quality and reducing its complexity. Moreover, SemBigOM has successfully outperformed the existing ontology matching

systems, and addressed their drawbacks as well as the limits of NeuralOM and DeepOM. The high performance achieved by SemBigOM is due to the fact that it has excellently tackled the challenges which have motivated this work, and has been developed with a very high level of accuracy.

6.5. Conclusion

In this chapter, we have presented SemBigOM, an ontology matching solution that we propose for dealing with the heterogeneity problem at the large-scale. We describe the proposed methodology and outline its contributions. And, we present the experimental procedure that we have performed on the Anatomy track from the 2022 campaign of the OAEI Initiative.

The results of evaluating SemBigOM on large OAEI ontologies and its comparison with the different participant ontology matching systems, show that, it has proven its efficiency to effectively match large-scale ontologies against the OAEI matching systems as well as our previously proposed solutions.

Conclusion and Open Issues

Main Contributions

Ontology matching is an effective process to establish interoperability between heterogeneous ontologies. The ontology matching field is maturing with enormous number of matching techniques. However, dealing with large ontologies still remains a key challenge. Much more work is required at this scale due to the fact that, ontology matching systems suffer from difficulties related to memory consumption and processing time at the large scale. The existing matching techniques which deal with this problem are based on ontology partitioning which is also challenging. However, partitioning ontologies also suffers from interesting challenges. This has a direct impact on the efficiency of the ontology matching process.

Deep learning algorithms have motivated numerous researchers in many fields solving different and complex problems which require powerful computational tools. However, they have limited use in ontology matching, particularly in large-scale ontology matching.

In this study, we propose three different solutions to the large-scale ontology matching problem:

1. **NeuralOM**, a mapping reuse-based large-scale ontology matching approach that we develop basing on artificial neural networks. As we aim not just to generate alignments between heterogeneous ontologies but to ideally match them, we developed it on a very high level of accuracy.

   In order to properly evaluate the effectiveness of NeuralOM, we conducted much precise experiments on twelve test cases of different domains as well as of different dataset sizes from OAEI initiative. The results of these experiments show that the proposed approach has proven its efficiency in front of all OAEI matching systems in terms of the five evaluation measures adopted, with very excellent scores for all matching tasks and with negligible matching time even for very large ontologies. That permits to, significantly increase the performance of the ontology matching

task, and perfectly tackle the large-scale ontology matching challenges which have motivated this work.

2. **DeepOM**, a large-scale ontology matching system based on deep learning techniques to deal with the large-scale heterogeneity problem without partitioning. The key novelty of DeepOM is to use a reference ontology to create semantic embeddings for ontological concepts, which are used to train an auto-encoder in order to learn more accurate and less dimensional representations for concepts.

   The results of its evaluation on large OAEI ontologies, and its comparison with ontology matching systems participant to the same test case, show that, DeepOM outperforms the ontology matching baseline with high ability to tackle the large-scale problem. Learning concepts' embeddings using auto-encoder is effective for matching large-scale ontologies. All the matching factors of DeepOM are positive towards improving the ontology matching quality.

3. **SemBigOM**, the global methodology of this research that combines NeuralOM and DeepOM in order to overcome their limits and achieve excellent matching results independently. The core idea of SemBigOM is that, it exploits DeepOM to generate initial mappings that NeuralOM requires as input to be refined and reused so as to output the final matching results. It seeks for achieving the large-scale ontology matching process at a perfect and independent fashion.

   The results of evaluating SemBigOM on large OAEI ontologies against the different participant ontology matching systems as well as NeuralOM and DeepOM, demonstrate its high ability to tackle the large-scale ontology matching problems. It has successfully outperformed and addressed the limits of the existing work. SemBigOM has proven its high efficiency to perfectly achieve all objectives which have motivated this research.

Opens issues

Despite the significant contributions of this research to the field of ontology matching, several limitations were encountered during the development and evaluation of the proposed solutions. NeuralOM presents some technical limits related to the input data. The selected matching tools, which are not all available, are of different inputs and especially of different outputs. Choice criteria and number of chosen systems are additional challenges. As for

DeepOM and SemBigOM, the choice of the reference ontology is such a delicate and careful task which requires a domain expert. Additionally, the isolated concepts of the reference ontology are also challenging and affect the matching performance leading to zero-filled vectors of concepts. Furthermore, despite efforts to automate the matching process, there may still be a need for human intervention to elect the external resources, which can introduce delays and additional costs in real-world applications.

Future research efforts should focus on addressing these limitations by exploring more efficient algorithms, improving data quality, refining evaluation methodologies, and extending the scope of ontology matching systems to encompass a wider range of domains and ontological structures. Although the experimental results of the proposed solutions are very encouraging, we aim, as future work:

- At adopting the proposed solutions to accurately match large scale ontologies of specific fields. We also aim for using our matching methodology for semantic analysis and several objectives in real-application fields of ontology matching, such as, sentiment analysis, recommendation systems, documents classification, text processing in social media, …etc.
- As for DeepOM and SemBigOM, we aim at filtering concepts of the reference ontology, and removing its isolated and useless concepts before generating the new representations for input ontologies. We also plan to pass the representations of concepts of the input ontologies from one-index to two-indices arrays in order to have more precise representations.
- In addition, we plan to deal with particular matching tasks, such as cross ontology matching, basing on other different and more complicated test cases with larger and more huge ontologies with many instances, especially for DeepOM and SemBigOM.
- Besides, we intend to evaluate, not just the mapping of our ontology matching approaches, but also the similarity values of each generated alignment between ontologies using different and specific evaluation measures dedicated for such purposes.
- Moreover, we plan for participating at the ontology alignment evaluation initiative in several tracks, in order to assess strengths and weaknesses of our work, increase communication among other developers and to help improving the research on ontology matching, particularly on large-scale ontology matching.

Bibliography

1. Gruber, T.R., *A translation approach to portable ontology specifications.* Knowledge acquisition, 1993. **5**(2): p. 199-220.
2. Zamazal, O., *A Survey of Ontology Benchmarks for Semantic Web Ontology Tools.* International Journal on Semantic Web and Information Systems (IJSWIS), 2020. **16**(1): p. 47-68.
3. Ougouti, N.S., H. Belbachir, and Y. Amghar, *Semantic Mediation in MedPeer: An Ontology-Based Heterogeneous Data Sources Integration System.* International Journal of Information Technology and Web Engineering (IJITWE), 2017. **12**(1): p. 1-18.
4. Rinaldi, A.M., C. Russo, and K. Madani, *A semantic matching strategy for very large knowledge bases integration.* International Journal of Information Technology and Web Engineering (IJITWE), 2020. **15**(2): p. 1-29.
5. Tran, D.-T., D.-H. Ngo, and P.-T. Do. *An information content based partitioning method for the anatomical ontology matching task*. in *Proceedings of the Third Symposium on Information and Communication Technology*. 2012.
6. Laadhar, A., et al. *Partitioning and local matching learning of large biomedical ontologies*. in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019.
7. Laadhar, A., et al. *Partitioning and matching tuning of large biomedical ontologies*. in *13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (OM 2018)*. 2018.
8. Jiménez-Ruiz, E., et al. *We divide, you conquer: from large-scale ontology alignment to manageable subtasks with a lexical index and neural embeddings*. in *CEUR Workshop Proceedings*. 2018.
9. Balachandran, S., V. Ranganathan, and D. Vetriveeran, *Aligning large biomedical ontologies for semantic interoperability using graph partitioning.* International Journal of Biomedical Engineering and Technology, 2019. **31**(2): p. 137-160.
10. Portisch, J. and H. Paulheim, *ALOD2Vec matcher*. OM@ ISWC, 2018. **2288**: p. 132-137.
11. Chang, F., G. Chen, and S. Zhang. *FCAMap-KG results for OAEI 2019*. in *OM@ ISWC*. 2019.
12. Hertling, S. and H. Paulheim, *DOME results for OAEI 2018.* OM@ ISWC, 2018. **2288**: p. 144-151.
13. Monych, M., et al. *DESKMatcher*. in *CEUR Workshop Proceedings*. 2020. RWTH.
14. Roussille, P., et al. *Holontology: results of the 2018 OAEI evaluation campaign*. 2018. CEUR-WS: Workshop proceedings.

15. Schreiber, A.T., et al., *Knowledge engineering and management: the CommonKADS methodology*2000: MIT press.

16. Euzenat, J. and P. Shvaiko, *Ontology Matching*2013: Springer Berlin Heidelberg.

17. Guizzardi, G., *Ontological foundations for structural conceptual models.* 2005.

18. Strawson, P.F. and R. Bubner, *Semantik und Ontologie.* Neue Hefte für Philosophie, 1975.

19. Neches, R., et al., *Enabling technology for knowledge sharing.* AI magazine, 1991. **12**(3): p. 36-36.

20. Borst, W., *Construction of engineer ontologies.* Ph thesis, University of Twenty, Enschede, 1997.

21. Studer, R., V.R. Benjamins, and D. Fensel, *Knowledge engineering: principles and methods.* Data & Knowledge Engineering, 1998. **25**(1-2): p. 161-197.

22. Guarino, N. and P. Giaretta, *Ontologies and knowledge bases.* Towards very large knowledge bases, 1995: p. 1-2.

23. Schreiber, G., B. Wielinga, and W. Jansweijer. *The KACTUS view on the 'O'word*. in *IJCAI workshop on basic ontological issues in knowledge sharing*. 1995. Citeseer.

24. Swartout, B., et al. *Toward distributed use of large-scale ontologies*. in *Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*. 1996.

25. Sowa, J.F., *Knowledge representation: logical, philosophical and computational foundations*1999: Brooks/Cole Publishing Co.

26. Udrea, O., L. Getoor, and R.J. Miller. *Leveraging data and structure in ontology integration*. in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007.

27. Cheatham, M. and C. Pesquita, *Semantic data integration*, in *Handbook of big data technologies*2017, Springer. p. 263-305.

28. Zhang, L.-Y., J.-D. Ren, and X.-W. Li, *OIM-SM: A method for ontology integration based on semantic mapping.* Journal of Intelligent & Fuzzy Systems, 2017. **32**(3): p. 1983-1995.

29. Charlet, J., B. Bachimont, and R. Troncy, *Ontologies pour le web sémantique*. Revue I3, numéro Hors Série «Web sémantique, 2004: p. 43-63.

30. Ghawi, R., *Ontology-based cooperation of information systems: contributions to database-to-ontology mapping and XML-to-ontology mapping*, 2010, Dijon.

31. Staab, S. and R. Studer, *Handbook on ontologies*2004: Springer Science & Business Media.

32. Corcho, O., A. Gómez-Pérez, and M. Fernández-López, *Ontological engineering.* With examples from the areas of Knowledge Management, e-Commerce and the Semantic Web (Advanced Information and Knowledge Processing), 2004.

33. Lenat, D. and R. Guha, *Building large knowledge-based systems: Representation and inference in the CYC project.* Artificial Intelligence, 1993. **61**(1): p. 4152.

34. Genesereth, M.R. and R.E. Fikes, *Knowledge interchange format-version 3.0: reference manual.* 1992.

35. Gruber, T., *A mechanism to support portable ontologies.* Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

36. Farquhar, A., R. Fikes, and J. Rice, *The ontolingua server: A tool for collaborative ontology construction.* International journal of human-computer studies, 1997. **46**(6): p. 707-727.

37. MacGregor, R.M., *Inside the LOOM description classifier.* ACM Sigart Bulletin, 1991. **2**(3): p. 88-92.

38. Motta, E., *Reusable components for knowledge modelling*1998: Open University (United Kingdom).

39. Kifer, M., G. Lausen, and J. Wu, *Logical foundations of object-oriented and frame-based languages.* Journal of the ACM (JACM), 1995. **42**(4): p. 741-843.

40. Heflin, J., J.A. Hendler, and S. Luke, *SHOE: A blueprint for the semantic web.* Spinning the Semantic Web, 2003: p. 29-63.

41. Bray, T., et al., *Extensible Markup Language (XML) 1.0 . W3C Recommendation 26 November 2008.* Available at h ttp://www. w3. org/TR/REC-xml, 2008.

42. Karp, P.D., V.K. Chaudhri, and J. Thomere, *XOL: An XML-based ontology exchange language*, 1999, July.

43. Klyne, G., *Resource description framework (RDF): Concepts and abstract syntax.* http://www. w3. org/TR/2004/REC-rdf-concepts-20040210/, 2004.

44. McGuinness, D.L., et al., *DAML+ OIL: an ontology language for the Semantic Web.* IEEE Intelligent Systems, 2002. **17**(5): p. 72-80.

45. McGuinness, D.L. and F. Van Harmelen, *OWL web ontology language overview.* W3C recommendation, 2004. **10**(10): p. 2004.

46. Benerecetti, M., P. Bouquet, and C. Ghidini. *On the dimensions of context dependence: partiality, approximation, and perspective*. in *International and Interdisciplinary Conference on Modeling and Using Context*. 2001. Springer.

47. Shvaiko, P. and J. Euzenat, *Ontology matching: state of the art and future challenges.* IEEE Transactions on knowledge and data engineering, 2013. **25**(1): p. 158-176.

48. Rahm, E., *A survey of approaches to automatic schema matching.* VLDB Journal, 2001. **10**(4).

49. Choi, N., I.-Y. Song, and H. Han, *A survey on ontology mapping.* ACM SIGMOD Record, 2006. **35**(3): p. 34-41.

50. Otero-Cerdeira, L., F.J. Rodríguez-Martínez, and A. Gómez-Rodríguez, *Ontology matching: A literature review*. Expert systems with applications, 2015. **42**(2): p. 949-971.

51. Thiéblin, E., et al., *Survey on complex ontology matching.* Semantic Web, 2020. **11**(4): p. 689-727.

52. Laadhar, A., et al. *OAEI 2018 results of POMap++*. in *13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference (OM@ ISWC 2018)*. 2018.

53. Laadhar, A., et al. *POMap++ results for OAEI 2019: fully automated machine learning approach for ontology matching*. in *14th International Workshop on Ontology Matching co-located with the International Semantic Web Conference (OM@ ISWC 2019)*. 2019.

54. Zou, S., et al., *Lily Results for OAEI 2021*. 2021.

55. Jiménez-Ruiz, E., *LogMap family participation in the OAEI 2021*, 2021. p. 175-177.

56. Zhang, Y., et al. *RiMOM results for OAEI 2016*. in *OM@ ISWC*. 2016.

57. Djeddi, W.E., S.B. Yahia, and M.T. Khadir, *XMap: results for OAEI 2018*, 2018. p. 210-215.

58. Hu, W., et al., *Objectcoref & falcon-ao: results for oaei 2010.* Ontology Matching, 2010. **2010**.

59. Faria, D., et al., *AML and AMLC results for OAEI 2021*. OM@ ISWC, 2021. **2019**.

60. Chua, W.W.K. and J.-J. Kim, *Eff2Match results for OAEI 2010*, 2010.

61. Chen, G. and S. Zhang. *FCAMapX results for OAEI 2018*. in *OM@ ISWC*. 2018.

62. Kengue, J.F.D., J. Euzenat, and P. Valtchev. *OLA in the OAEI 2007 evaluation contest*. in *Proceedings of ISWC+ ASWC Workshop on Ontology Matching*. 2007. Citeseer.

63. David, J., *AROMA results for OAEI 2011*. Ontology Matching, 2011. **122**.

64. Massmann, S., D. Engmann, and E. Rahm, *COMA++: Results for the Ontology Alignment Contest OAEI 2006.* Ontology Matching, 2006. **225**.

65. Seddiqui, M.H. and M. Aono. *Anchor-flood: results for OAEI 2009*. in *Proceedings of the ISWC 2009 Workshop on ontology matching*. 2009. Citeseer.

66. Gulić, M., B. Vrdoljak, and M. Banek, *CroMatcher-Results for OAEI 2016.* Ontology Matching, 2016: p. 153.

67. Hamdi, F., et al., *TaxoMap alignment and refinement modules: Results for OAEI 2010.* Ontology Matching, 2010: p. 212.

68. Jean-Mary, Y.R., E.P. Shironoshita, and M.R. Kabuka, *Asmov: Results for oaei 2010.* Ontology Matching, 2010. **126**: p. 2010.

69. Khiat, A. and M. Benaissa. *InsMT+ results for OAEI 2015 instance matching*. in *OM*. 2015.

70. Lambrix, P., H. Tan, and Q. Liu. *SAMBO and SAMBOdtf results for the ontology alignment evaluation initiative 2008*. in *Proceedings of the Third International Workshop on Ontology Matching*. 2008.

71. Laadhar, A., *Local matching learning of large scale biomedical ontologies*, 2019, Université de Toulouse, Université Toulouse III-Paul Sabatier.

72. Huber, J., et al., *Codi: Combinatorial optimization for data integration–results for oaei 2011*. Ontology Matching, 2011. **134**.

73. Annane, A., et al., *Yam-bio–results for oaei 2017*. 2017.

74. Groß, A., et al. *GOMMA results for OAEI 2012*. in *OM*. 2012.

75. Garcia-Castro, R., et al., *Specification of a methodology, general criteria, and benchmark suites for benchmarking ontology tools.* Deliverable D2, 2004. **1**.

76. Berners-Lee, T., *J., Hendler, and O. Lassila,"The semantic web".* Scientific american, 2001. **284**(5): p. 35-43.

77. Berners-Lee, T., *Linked Data http://www. w3. org/DesignIssues.* LinkedData. html, 2006.

78. Hamdi, F., et al., *Alignment-based partitioning of large-scale ontologies*, in *Advances in knowledge discovery and management*2010, Springer. p. 251-269.

79. Ochieng, P. and S. Kyanda, *Large-scale ontology matching: State-of-the-art analysis.* ACM Computing Surveys (CSUR), 2018. **51**(4): p. 1-35.

80. Karlapalem, K. and Q. Li, *A framework for class partitioning in object-oriented databases.* Distributed and Parallel Databases, 2000. **8**(3): p. 333-366.

81. Babalou, S., M.J. Kargar, and S.H. Davarpana. *A Comprehensive Review Of The Ontology Matching Systems By A Focus On Large Ontologies*. in *International Congress of Chemical and Process Engineering. Prague, Czech Republic*. 2014.

82. Babalou, S., M.J. Kargar, and S.H. Davarpanah. *Large-scale ontology matching: a review of the literature*. in *2016 Second International Conference on Web Research (ICWR)*. 2016. IEEE.

83. Grau, B.C., et al., *Automatic partitioning of OWL ontologies using e-connections.* Description Logics, 2005. **147**.

84. Kutz, O., et al., *E-connections of abstract description systems.* Artificial Intelligence, 2004. **156**(1): p. 1-73.

85. Garcia, A.C., et al. *Applying Graph Partitioning Techniques to Modularize Large Ontologies*. in *ONTOBRAS-MOST*. 2012. Citeseer.

86. Grau, B.C., et al. *Just the right amount: extracting modules from ontologies*. in *Proceedings of the 16th international conference on World Wide Web*. 2007.

87. Jiménez-Ruiz, E., et al. *Safe and economic re-use of ontologies: A logic-based methodology and tool support*. in *European Semantic Web Conference*. 2008. Springer.

88. Li, N. and E. Motta. *Evaluations of user-driven ontology summarization*. in *International Conference on Knowledge Engineering and Knowledge Management*. 2010. Springer.

89. Peroni, S., E. Motta, and M. d'Aquin. *Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures*. in *Asian Semantic Web Conference*. 2008. Springer.

90. Li, N., E. Motta, and M. d'Aquin, *Ontology summarization: an analysis and an evaluation.* 2010.

91. Zhang, X., G. Cheng, and Y. Qu. *Ontology summarization based on rdf sentence graph*. in *Proceedings of the 16th international conference on World Wide Web*. 2007.

92. Rahm, E., *Towards large-scale schema and ontology matching*, in *Schema matching and mapping*2011, Springer. p. 3-27.

93. Algergawy, A., S. Massmann, and E. Rahm. *A clustering-based approach for large-scale ontology matching*. in *East European Conference on Advances in Databases and Information Systems*. 2011. Springer.

94. Yuruk, N., et al. *AHSCAN: Agglomerative hierarchical structural clustering algorithm for networks*. in *2009 International Conference on Advances in Social Network Analysis and Mining*. 2009. IEEE.

95. Ahmed, S.S., M. Malki, and S.M. Benslimane, *Ontology partitioning: Clustering based approach.* International Journal of Information Technology and Computer Science, 2015. **7**(6): p. 1-11.

96. Resnik, P., *Using information content to evaluate semantic similarity in a taxonomy.* arXiv preprint cmp-lg/9511007, 1995.

97. Hu, W., Y. Zhao, and Y. Qu. *Partition-based block matching of large class hierarchies*. in *Asian Semantic Web Conference*. 2006. Springer.

98. Guha, S., R. Rastogi, and K. Shim, *ROCK: A robust clustering algorithm for categorical attributes.* Information Systems, 2000. **25**(5): p. 345-366.

99. Dragisic, Z., et al. *Results of the ontology alignment evaluation initiative 2014*. in *9th ISWC workshop on ontology matching (OM)*. 2014. No commercial editor.

100. Gross, A., et al. *On matching large life science ontologies in parallel*. in *International Conference on Data Integration in the Life Sciences*. 2010. Springer.

101. Ara, T.B., et al., *A parallel approach for matching large-scale ontologies.* Journal of Information and Data Management, 2015. **6**(1): p. 18-18.

102. Rahm, E. and P.A. Bernstein, *A survey of approaches to automatic schema matching.* the VLDB Journal, 2001. **10**(4): p. 334-350.

103. Müllner, D., *Modern hierarchical, agglomerative clustering algorithms.* arXiv preprint arXiv:1109.2378, 2011.

104. Wang, Z., et al. *Matching large scale ontology effectively*. in *Asian Semantic Web Conference*. 2006. Springer.

105. Lambrix, P. and Q. Liu. *Using partial reference alignments to align ontologies*. in *European Semantic Web Conference*. 2009. Springer.

106. Lambrix, P. and R. Kaliyaperumal, *A session-based ontology alignment approach enabling user involvement 1.* Semantic Web, 2017. **8**(2): p. 225-251.

107. Li, Y., et al., *Matching large scale ontologies based on filter and verification.* Mathematical Problems in Engineering, 2020. **2020**.

108. Kachroudi, M., S. Zghal, and S. Ben Yahia, *Ontopart: at the cross-roads of ontology partitioning and scalable ontology alignment systems.* International Journal of Metadata, Semantics and Ontologies, 2013. **8**(3): p. 215-225.

109. Do, H.-H. and E. Rahm. *COMA—a system for flexible combination of schema matching approaches*. in *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. 2002. Elsevier.

110. Aumueller, D., et al. *Schema and ontology matching with COMA++*. in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005.

111. Do, H.-H. and E. Rahm, *Matching large schemas: Approaches and evaluation.* Information Systems, 2007. **32**(6): p. 857-885.

112. Zhang, H., W. Hu, and Y.-z. Qu, *VDoc+: a virtual document based approach for matching large ontologies using MapReduce.* Journal of Zhejiang University SCIENCE C, 2012. **13**(4): p. 257-267.

113. Qu, Y., W. Hu, and G. Cheng. *Constructing virtual documents for ontology matching*. in *Proceedings of the 15th international conference on World Wide Web*. 2006.

114. Santos, E., et al., *Ontology alignment repair through modularization and confidence-based heuristics.* PloS one, 2015. **10**(12): p. e0144807.

115. Hu, W., Y. Qu, and G. Cheng, *Matching large ontologies: A divide-and-conquer approach.* Data & Knowledge Engineering, 2008. **67**(1): p. 140-160.

116. Madhavan, J., et al. *Corpus-based schema matching*. in *21st International Conference on Data Engineering (ICDE'05)*. 2005. IEEE.

117. Algergawy, A., et al. *Seecont: A new seeding-based clustering approach for ontology matching*. in *East European Conference on Advances in Databases and Information Systems*. 2015. Springer.

118. Debora, N., *Deep Learning for Feature Representation in Natural Language Processing*, 2017, University of Milan-Bicocca.

119. Stiles, J. and T.L. Jernigan, *The basics of brain development.* Neuropsychology review, 2010. **20**(4): p. 327-348.

120. Churchland, P. and T. Sejnowski, *The computational brain MIT Press.* Cambridge, Massachusetts, 1992.

121. Bottou, L., *Stochastic gradient descent tricks*, in *Neural networks: Tricks of the trade*2012, Springer. p. 421-436.

122. Polyak, B.T., *Some methods of speeding up the convergence of iteration methods.* Ussr computational mathematics and mathematical physics, 1964. **4**(5): p. 1-17.

123. Sutskever, I., et al. *On the importance of initialization and momentum in deep learning*. in *International conference on machine learning*. 2013. PMLR.

124. Nesterov, Y.E. *A method for solving the convex programming problem with convergence rate O (1/k^ 2)*. in *Dokl. akad. nauk Sssr*. 1983.

125. Duchi, J., E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.* Journal of machine learning research, 2011. **12**(7).

126. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.

127. Tieleman, T. and G. Hinton, *Lecture 6.5-rmsprop, coursera: Neural networks for machine learning.* University of Toronto, Technical Report, 2012. **6**.

128. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

129. Zeiler, M.D., *Adadelta: an adaptive learning rate method.* arXiv preprint arXiv:1212.5701, 2012.

130. Crammer, K. and Y. Singer, *On the algorithmic implementation of multiclass kernel-based vector machines.* Journal of machine learning research, 2001. **2**(Dec): p. 265-292.

131. Bishop, C.M., *Neural networks for pattern recognition*1995: Oxford university press.

132. Tihonov, A.N., *Solution of incorrectly formulated problems and the regularization method.* Soviet Math., 1963. **4**: p. 1035-1038.

133. Tibshirani, R., *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society: Series B (Methodological), 1996. **58**(1): p. 267-288.

134. Zou, H. and T. Hastie, *Regularization and variable selection via the elastic net.* Journal of the royal statistical society: series B (statistical methodology), 2005. **67**(2): p. 301-320.

135. Srivastava, N., et al., *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research, 2014. **15**(1): p. 1929-1958.

136. Hinton, G.E. and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks.* science, 2006. **313**(5786): p. 504-507.

137. Cappuzzo, R., P. Papotti, and S. Thirumuruganathan. *Creating embeddings of heterogeneous relational datasets for data integration tasks*. in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020.

138. Kalinowski, A. and Y. An, *A Survey of Embedding Space Alignment Methods for Language and Knowledge Graphs.* arXiv preprint arXiv:2010.13688, 2020.

139. Funahashi, K.-I., *On the approximate realization of continuous mappings by neural networks.* Neural networks, 1989. **2**(3): p. 183-192.

140. Kuroe, Y., Y. Nakai, and T. Mori. *A learning method of nonlinear mappings by neural networks with considering their derivatives*. in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. 1993. IEEE.

141. Li, W.-S. and C. Clifton. *Semint: A system prototype for semantic integration in heterogeneous databases*. in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. 1995.

142. Li, W.-S. and C. Clifton, *SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks.* Data & Knowledge Engineering, 2000. **33**(1): p. 49-84.

143. Li, W.-S. and C. Clifton. *Using field specifications to determine attribute equivalence in heterogeneous databases*. in *Proceedings RIDE-IMS93: Third International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*. 1993. IEEE.

144. Li, W.-S. and C. Clifton. *Semantic integration in heterogeneous databases using neural networks*. in *Proceedings of the 20th international conference on very large data bases*. 1994.

145. Li, W.-S., C. Clifton, and S.-Y. Liu, *Database integration using neural networks: implementation and experiences.* Knowledge and Information Systems, 2000. **2**(1): p. 73-96.

146. Ehrig, M. and Y. Sure. *Ontology mapping–an integrated approach*. in *European Semantic Web Symposium*. 2004. Springer.

147. Ehrig, M., S. Staab, and Y. Sure. *Bootstrapping ontology alignment methods with APFEL*. in *International semantic Web conference*. 2005. Springer.

148. Chortaras, A., G. Stamou, and A. Stafylopatis. *Learning ontology alignments using recursive neural networks*. in *International conference on artificial neural networks*. 2005. Springer.

149. Hariri, B.B., H. Abolhassani, and H. Sayyadi. *A neural-networks-based approach for ontology alignment*. in *SCIS & ISIS SCIS & ISIS 2006*. 2006. Japan Society for Fuzzy Theory and Intelligent Informatics.

150. Stegmayer, G., et al. *ANN-agent for distributed knowledge source discovery*. in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. 2007. Springer.

151. Huang, J., et al. *Ontology matching using an artificial neural network to learn weights*. in *IJCAI workshop on semantic Web for collaborative knowledge acquisition*. 2007.

152. Huang, J., et al., *Use artificial neural network to align biological ontologies.* BMC genomics, 2008. **9**(2): p. 1-12.

153. Ichise, R. *Machine learning approach for ontology mapping using multiple concept similarity measures*. in *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. 2008. IEEE.

154. Curino, C., G. Orsi, and L. Tanca. *X-som: A flexible ontology mapper*. in *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*. 2007. IEEE.

155. Curino, C., G. Orsi, and L. Tanca, *X− SOM Results for OAEI 2007.* 2007.

156. Merlin, P., et al. *X-SOM and L-SOM: a nested approach for missing value imputation*. in *ESANN*. 2009. Citeseer.

157. Merlin, P., et al., *X-SOM and L-SOM: a double classification approach for missing value imputation.* Neurocomputing, 2010. **73**(7-9): p. 1103-1108.

158. Peng, Y., P. Munro, and M. Mao. *Learning to map ontologies with neural network*. in *Proceedings of the 4th International Conference on Ontology Matching-Volume 551*. 2009.

159. Peng, Y., P. Munro, and M. Mao, *Ontology Mapping Neural Network: An Approach to Learning and Inferring Correspondences among Ontologies.* 2010.

160. Mao, M., Y. Peng, and M. Spring. *A profile propagation and information retrieval based ontology mapping approach*. in *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*. 2007. IEEE.

161. Mao, M., Y. Peng, and M. Spring, *An adaptive ontology mapping approach with neural network based constraint satisfaction.* Web Semantics: Science, Services and Agents on the World Wide Web, 2010. **8**(1): p. 14-25.

162. Mao, M. and Y. Peng. *PRIOR System: Results for OAEI 2006*. in *Ontology matching*. 2006.

163. Mao, M. and Y. Peng. *The PRIOR+: Results for OAEI Campaign 2007*. in *OM*. 2007.

164. Mao, M., *Ontology mapping: An information retrieval and interactive activation network based approach.* The Semantic Web, 2007: p. 931-935.

165. Mao, M., Y. Peng, and M. Spring. *Integrating the IAC neural network in ontology mapping*. in *Proceedings of the 17th international conference on World Wide Web*. 2008.

166. Mao, M., Y. Peng, and M. Spring. *Neural Network based Constraint Satisfaction in Ontology Mapping*. in *AAAI*. 2008.

167. Mao, M., Y. Peng, and M. Spring. *A Harmony based Adaptive Ontology Mapping Approach*. in *SWWS*. 2008.

168. Esposito, F., N. Fanizzi, and C. d'Amato. *Recovering uncertain mappings through structural validation and aggregation with the MoTo system*. in *Proceedings of the 2010 ACM Symposium on Applied Computing*. 2010.

169. Fanizzi, N., C. d'Amato, and F. Esposito, *Composite ontology matching with uncertain mappings recovery.* ACM SIGAPP Applied Computing Review, 2011. **11**(2): p. 17-29.

170. Rubiolo, M., et al., *Knowledge discovery through ontology matching: An approach based on an Artificial Neural Network model.* Information Sciences, 2012. **194**: p. 107-119.

171. Gracia, J. and E. Mena. *Ontology matching with CIDER: Evaluation report for the OAEI 2008*. in *Proceedings of the 3rd International Conference on Ontology Matching-Volume 431*. 2008. CEUR-WS. org.

172. del Río, J.G., J. Bernad, and E. Mena, *Ontology Matching with CIDER: evaluation report for OAEI 2011*. 2011.

173. Gracia, J. and K. Asooja, *Monolingual and cross-lingual ontology matching with CIDER-CL: evaluation report for OAEI 2013.* OM, 2013. **1111**: p. 109-116.

174. Shenoy, K.M., K. Shet, and U.D. Acharya. *NN based ontology mapping*. in *International Conference on Advances in Information Technology and Mobile Communication*. 2012. Springer.

175. Djeddi, W.E. and M.T. Khadir. *XMAP: a novel structural approach for alignment of OWL-full ontologies*. in *2010 International Conference on Machine and Web Intelligence*. 2010. IEEE.

176. Djeddi, W.E. and T. Khadir. *A Dynamic Multistrategy Ontology Alignment Framework Based on Semantic Relationship using WordNet*. in *CIIA*. 2011. Citeseer.

177. Khadir, M., A. Djeddai, and W. Djeddi. *XMap++: A novel semantic approach for alignment of OWL-Full ontologies based on semantic relationship using WordNet*. in *International Symposium on Innovations in Information and Communications Technology*. 2011. IEEE.

178. Djeddi, W.E. and M.T. Khadir, *Introducing artificial neural network in ontologies alignment process.* Control and Cybernetics, 2012. **41**(4).

179. Djeddi, W.E. and M.T. Khadir, *Introducing artificial neural network in ontologies alignement process*, in *New Trends in Databases and Information Systems*2013, Springer. p. 175-186.

180. Djeddi, W.E. and M.T. Khadir, *Ontology alignment using artificial neural network for large-scale ontologies.* International Journal of Metadata, Semantics and Ontologies 16, 2013. **8**(1): p. 75-92.

181. Djeddi, W.E. and M.T. Khadir. *XMapGen and XMapSiG results for OAEI 2013*. in *OM*. 2013.

182. Djeddi, W.E. and M.T. Khadir. *A novel approach using context-based measure for matching large scale ontologies*. in *International Conference on Data Warehousing and Knowledge Discovery*. 2014. Springer.

183. Djeddi, W.E. and M.T. Khadir. *Xmap++: results for Oaei 2014*. in *Proceedings of the 9th International Conference on Ontology Matching-Volume 1317*. 2014. CEUR-WS. org.

184. Djeddi, W.E., M.T. Khadir, and S.B. Yahia. *XMap: results for OAEI 2015*. in *OM*. 2015.

185. Djeddi, W.E., M.T. Khadir, and S.B. Yahia, *XMap results for OAEI 2016*, 2016. p. 222-226.

186. Djeddi, W.E., M.T. Khadir, and S.B. Yahia, *XMap results for OAEI 2017*, 2017. p. 196-200.

187. Viana, T., et al. *Ontology Alignment with Weightless Neural Networks*. in *International Conference on Artificial Neural Networks*. 2017. Springer.

188. Xiang, C., et al. *Ersom: A structural ontology matching approach using automatically learned entity representation*. in *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015.

189. Qiu, L., et al., *Knowledge entity learning and representation for ontology matching based on deep neural networks.* Cluster Computing, 2017. **20**(2): p. 969-977.

190. Nkisi-Orji, I., et al. *Ontology alignment based on word embedding and random forest classification*. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2018. Springer.

191. Chandrashekar, M., R. Nagulapati, and Y. Lee. *Ontology mapping framework with feature extraction and semantic embeddings*. in *2018 IEEE International Conference on Healthcare Informatics Workshop (ICHI-W)*. 2018. IEEE.

192. Liu, J., Y. Tang, and X. Xu. *HISDOM: A Hybrid Ontology Mapping System based on Convolutional Neural Network and Dynamic Weight*. in *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*. 2019.

193. Dhouib, M.T., C.F. Zucker, and A.G. Tettamanzi. *An ontology alignment approach combining word embedding and the radius measure*. in *International Conference on Semantic Systems*. 2019. Springer, Cham.

194. Zhang, Y., et al., *Ontology matching with word embeddings*, in *Chinese computational linguistics and natural language processing based on naturally annotated big data*2014, Springer. p. 34-45.

195. Ali Khoudja, M., M. Fareh, and H. Bouarfa. *Ontology matching using neural networks: survey and analysis*. in *2018 International Conference on Applied Smart Systems (ICASS)*. 2018. IEEE.

196. Ali Khoudja, M., M. Fareh, and H. Bouarfa. *A new supervised learning based ontology matching approach using neural networks*. in *International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*. 2018. Springer.

197. Ali Khoudja, M., M. Fareh, and H. Bouarfa. *Ontology matching using neural networks: Evaluation for OAEI tracks*. in *International Symposium on Modelling and Implementation of Complex Systems*. 2020. Springer.

198. Ali Khoudja, M., M. Fareh, and H. Bouarfa, *Deep Embedding Learning With Auto-Encoder for Large-Scale Ontology Matching.* International Journal on Semantic Web and Information Systems (IJSWIS), 2022. **18**(1): p. 1-18.

199. Zhang, Y., et al., *BioWordVec, improving biomedical word embeddings with subword information and MeSH.* Scientific data, 2019. **6**(1): p. 1-9.

200. Hertling, S. and H. Paulheim. *ATBox results for OAEI 2020*. in *CEUR Workshop Proceedings*. 2020. RWTH.

201. Chakraborty, J., et al. *OntoConnect: results for OAEI 2020*. in *OM@ ISWC*. 2020.

202. Portisch, J., M. Hladik, and H. Paulheim. *ALOD2Vec Matcher results for OAEI 2020*. in *CEUR Workshop Proceedings*. 2020. RWTH.

203. Lütke, A., *AnyGraphMatcher Submission to the OAEI Knowledge Graph Challenge 2019.* OM@ ISWC, 2019. **2536**: p. 86-93.

204. Faria, D., et al., *Matcha and Matcha-DL results for OAEI 2022.* 2022.

205. Gosselin, F. and A. Zouaq, *SEBMatcher Results for OAEI 2022.* 2022.

206. Jiménez-Ruiz, E. *LogMap Family Participation in the OAEI 2022*. in *CEUR Workshop Proceedings*. 2022.

207. Wang, Z., *AMD Results for OAEI 2022.* 2022.

208. Silva, J.d., et al., *ALIN Results for OAEI 2022.* 2022: p. 129-136.

209. Hertling, S. and H. Paulheim, *ATBox Results for OAEI 2022.* 2022: p. 153-157.

210. Sharma, A., A. Patel, and S. Jain, *LSMatch and LSMatch-Multilingual Results for OAEI.* 2022.

211. Alghamdi, S.M., F. Zhapa-Camacho, and R. Hoehndorf, *A-LIOn-Alignment Learning through Inconsistency negatives of the aligned Ontologies.* 2022.