TU WIEN Informatics

Aligning ontologies describing computer science for patents and scientific papers

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Hannes Marcher, BSc
Matrikelnummer 11776841

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury

Wien, 26. Jänner 2023

Hannes Marcher	Allan Hanbury

TU WIEN Informatics

Aligning ontologies describing computer science for patents and scientific papers

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Hannes Marcher, BSc
Registration Number 11776841

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury

Vienna, 26th January, 2023

Hannes Marcher	Allan Hanbury

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Hannes Marcher, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 26. Jänner 2023

Hannes Marcher

Danksagung

An dieser Stelle ist es nun an der Zeit, mich bei einigen Personen zu bedanken.

Besonders bedanke ich mich bei Univ. Prof. Dr. Allan Hanbury für die Unterstützung sowie die Bereitstellung dieses spannenden Themas. Ich bedanke mich für das Feedback sowie die Freiheiten, welche mir während der Ausarbeitung dieser Diplomarbeit eingeräumt worden sind. Ohne ihn wäre diese Diplomarbeit niemals zustande gekommen.

Ich möchte hier auch ganz bewusst meiner Familie, insbesondere meinen Eltern, danken. Danke für eure umfangreiche Unterstützung während der letzten Jahre. Ihr wart mir immer eine große Stütze, wobei ohne euch mein Studium in dieser Form so nicht möglich gewesen wäre.

Acknowledgements

At this point it is time to thank some people.

I want to especially thank Allan Hanbury for the support and the provision of this exciting topic. I am thankful for the constructive feedback as well as the freedoms that were granted to me during the elaboration of this thesis. Without him, this work would never have been accomplished.

I would also like to consciously thank my family here, especially my parents. Thank you for your extensive support during the last years. You have been a great support to me. Without you, my studies would not have been possible in this form.

Kurzfassung

Ontologien stellen den Grundstein des Semantic Web dar und ermöglichen die explizite Spezifikation von Konzeptualisierungen. An der Erstellung und Wartung von Ontologien sind eine Vielzahl unterschiedlicher Organisationen beteiligt. Da jede Organisation einen anderen Blickwinkel hat, sind selbst Ontologien über ähnliche Bereiche recht heterogen. Dadurch entsteht eine semantische Lücke zwischen verschiedenen Ontologien, die durch die Erstellung sogenannter *Ontology Alignments* geschlossen werden kann. In der Regel werden solche Alignments zwischen Ontologien mithilfe automatisierter *Ontology Alignment Systeme* generiert.

Das Hauptziel dieser Masterarbeit ist zweigeteilt. Zum einen werden in dieser Arbeit *Ontology Alignment Systeme* aus der Literatur auf ihre Anwendbarkeit zur Erstellung eines Alignments zwischen der Patentontologie CPC und den beiden Informatikontologien CCS und CSO geprüft. Zum anderen wurden zwei Alignments erstellt, ein Alignment zwischen CPC und CCS und ein weiteres Alignment zwischen CPC und CSO unter der Verwendung zweier geeigneter *Ontology Alignment Systeme*.

In einem ersten Schritt führt diese Arbeit eine Literaturrecherche durch, um vielversprechende Systeme zu ermitteln. Das Ergebnis besteht aus einer Reihe an *Ontology Alignment Systemen*, welche alle Voraussetzungen erfüllen, um Alignments zwischen den oben genannten Ontologien zu erstellen.

In einem weiteren Schritt wird die Leistung jedes Ansatzes auf einem manuell erstellten Referenzdatensatz sowie auf dem Anatomie-Anwendungsfall der Ontology Alignment Evaluation Initiative ermittelt. Die Leistung wird hierbei in Form von Precision, Recall sowie F1-Score gemessen.

Final werden dann die endgültigen Alignments, welche mit den vielversprechendsten Ansätzen erzeugt wurden, vorgestellt. Das Ergebnis dieses Schrittes besteht aus einer Bewertung der beiden Alignments unter Verwendung von approximierten Precision, Recall und F1-Score Maßen.

Die Hauptergebnisse dieser Arbeit sind: (a) Die State-of-the-Art Systeme LogMap und AML sind am besten geeignet, um Alignments zwischen CPC und CCS bzw. CPC und CSO zu erstellen. (b) Die generierten Alignments stellen einen nützlichen ersten Schritt dar. Sie reichen jedoch nicht aus, um die semantische Lücke zwischen diesen Ontologien zu schließen.

Abstract

Ontologies are the core of the Semantic Web and enable the explicit specification of conceptualizations. A large variety of different communities are involved in the generation and maintenance of ontologies. Because each community has a different perspective, even ontologies about similar domains are quite heterogeneous. This imposes a semantic gap between different ontologies which can be addressed through the creation of *ontology alignments*. Typically, alignments between ontologies are generated through *ontology alignment systems* - automated tools to generate such alignments.

The major aim of this master thesis is two folded. First, this work compares ontology alignment systems from the literature towards their applicability to align the patent ontology CPC with the two computer science ontologies CCS and CSO. Second, two alignments were created, one between CPC and CCS and one between CPC and CSO using two suitable ontology alignment systems.

As the initial step, this thesis conducts a literature review to elicit promising ontology alignment systems. The outcome is a set of ontology alignment systems that fit the characteristics of the use case of this thesis.

Next, this thesis presents the performance of each approach on a manually created ground truth as well as on the Anatomy test case of the Ontology Alignment Evaluation Initiative. The outcome comprises a comparison using precision, recall, and f1-score.

Third, the characteristics of the final alignments that have been generated using the most promising approaches, are presented. The result of this step consists of an evaluation of the two alignments, i.e. one alignment between CPC and CCS and one alignment between CPC and CSO, using estimated precision, recall, and f1-score measures.

The major results of this thesis are: (a) The State-of-the-Art matchers LogMap and AML are most suited to align CPC with CCS and CPC with CSO, respectively. (b) The alignments represent a useful first building block in aligning the patent ontology CPC with the computer science ontologies CCS and CSO, but they are not sufficient to bridge the semantic gap between these ontologies.

Contents

CHAPTER 1

Introduction

The Semantic Web is an established infrastructure to share knowledge on the web [1]. *Ontologies* serve as the core for the Semantic Web [1] and enable the explicit specification of conceptualizations [2]. Different communities design and maintain ontologies. Because each community has a different perspective, it is common that even ontologies about similar domains are represented quite differently [3]. It might, for instance, happen that independent communities use the same words to describe different concepts, as well as utilize different words to refer to the same concept [4]. This imposes a semantic gap between different ontologies. This semantic gap might be tackled through discovering mappings between ontologies that contain overlapping representations of the same domain [3]. A set of these mappings is often referred to as an *ontology alignment*. Manually finding such relations is tedious, error-prone, and not feasible when considering the scale of modern ontologies [5]. Therefore, automated tools are crucial in finding alignments between ontologies. These tools are often referred to as *ontology alignment systems* or *ontology matching systems*. Typically, such systems map concepts from a source ontology to concepts from a target ontology, thus bridging the semantic gap between two ontologies.

This thesis investigates ontology alignment in the context of three taxonomies that are all to some extent related to computer science. The first ontology represents a patent classification scheme, the second and third ontologies represent taxonomies to categorize computer science publications. The overall goal is to find suitable ontology alignment systems to align the patent ontology with both computer science research ontologies.

This chapter introduces the reader to the problem domain and the research aim. Section 1.1 justifies this work through providing the concrete problem description. In the subsequent section the major goals are outlined. Section 1.3 then defines the research questions and proposes a research roadmap to reach the envisioned goals. The fourth section gives a very brief overview of the open source repository that underlies this work. The chapter then concludes in Section 1.5 by outlining the structure of this thesis.

1.1 Motivation & Problem Description

Scientific papers are often labeled according to terms taken from ontology classes. In the field of computer science, two prominent ontologies are the ACM Computing Classification System (CCS) [1] and the Computer Science Ontology (CSO) [2]. While the CCS ontology consists of only roughly 2K research topics, the CSO is much larger, i.e. it contains about 26K research topics and 226K semantic relationships [6]. On the other hand, patents are the practical application of technology [7] and also often labeled according to ontologies. A very prominent ontology in this field is the Cooperative Patent Classification (CPC) [3]. The CPC consists of nine sections, that are further split into classes, subclasses, groups, and subgroups. The whole CPC ontology consists of roughly 250K classes [8].

Knowing which ontology classes of CPC correspond to which classes from CCS and CSO, i.e. between patent categories and computer science research areas, is very desired when it comes to various analysis tasks. For example, Kotti et al. [7] investigated to what extent scientific work in the field of software engineering impacts industry. Knowing which ontology class in CPC corresponds to which ontology class(es) in CCS and CSO would facilitate similar analysis approaches. However, these three ontologies have quite different structures and deviate in various characteristics, such as in their granularity, their categorization schemes, their number of classes, and so on. Due to this heterogeneity and the huge sizes of these ontologies, it is a non-trivial task to find corresponding ontology classes. This is where *Challenge 1* of this thesis arises, i.e. it is not clear which ontological concepts form CPC correspond to which ontological concepts in CCS and CSO.

One way to bridge this gap would be to search manually for correspondences. For example, assume one would like to know which class(es) in CPC correspond(s) to the concept *model driven development* of CSO, then one might conduct a tedious and time-consuming manual search. While this approach is definitely doable for a few classes, it becomes infeasible when the number of classes increases. Additionally, such a manual procedure is tedious and error-prone [5].

Ontology Alignment (OA) (or equivalently, ontology matching) techniques aim at finding semantically related entities from two ontologies [9] and are, therefore, a natural fit to this problem domain. OA systems play an important role in knowledge engineering, and represent a key technique for ontology integration and quality assurance [10, 11]. In the literature one can find a large variety of OA approaches whereby, in general, there is no single best approach. For example, during the Ontology Alignment Evaluation Initiative (OAEI) [4] (a yearly held competition aiming at evaluating OA systems on various test cases) campaign of 2021 no OA approach was the best in all instances [12]. Thus, it is ad-hoc not clear which OA approach is appropriate to align CPC with CCS and CPC with CSO - *Challenge 2*.

[1] https://dl.acm.org/ccs
[2] https://cso.kmi.open.ac.uk/home
[3] https://www.cooperativepatentclassification.org/home
[4] http://oaei.ontologymatching.org/

1.2 Aim of the Work

According to the identified challenges from the previous section, the major outcome of this thesis is two-folded.

Comparison of Ontology Alignment Approaches. The first major outcome of this thesis comprises a comparison between various OA approaches that expresses the applicability of the related approaches to the problem domain of this thesis, thus addressing *Challenge 2* identified in the previous section. To identify candidate OA systems, a structured literature review has been conducted. As comparison metrics *precision*, *recall*, and *f1-score* have been chosen. Further, to gauge problem-specific performance figures, the comparison has been performed on two manually created ground truths to test the applicability of the approaches on CPC-CCS resp. CPC-CSO. In addition, to grasp scalability and to have another, problem-independent, performance measure, the approaches were also evaluated on a publicly available dataset from the literature.

Alignment between CPC-CCS as well as between CPC-CSO. The second major outcome consists of two full alignments, i.e. one alignment between CPC and CCS, as well as one alignment between CPC and CSO - addressing *Challenge 1* depicted in Section 1.1. The respective alignments were generated using the best performing approaches from the previous outcome. Due to the huge solution space and the large size of the final alignments (i.e. number of correspondences that are contained within an alignment), it is not possible to compute the exact *precision*, *recall*, and *f1-score* values. Instead, the performance measures were approximated through procedures known from literature.

1.3 Research Issues

To obtain the outlined contributions from the previous section, this section formulates three research issues that were elaborated in this thesis. Figure 1.1 depicts the research roadmap that is further detailed in the remainder of this section. The squares in the figure represent steps, the solid arrows indicate the order in which the steps were elaborated, and the dashed arrows indicate at which step the research questions were answered.

1.3.1 RI-1: Identification of Ontology Alignment Systems

As Section 1.1 already mentioned, there exists a large pool of different OA systems. The performance of these systems is potentially very different depending on the use case (see for example the performances of OAEI participants such as in the competition of 2021 [12]). Due to this consideration it is a-priori not clear which OA system is suited to align the ontologies of this thesis. Additionally, many of the systems from the literature are not openly available, therefore evaluating an approach might not be as straight-forward as it initially may appear. This is why a structured approach is necessary to elicit promising

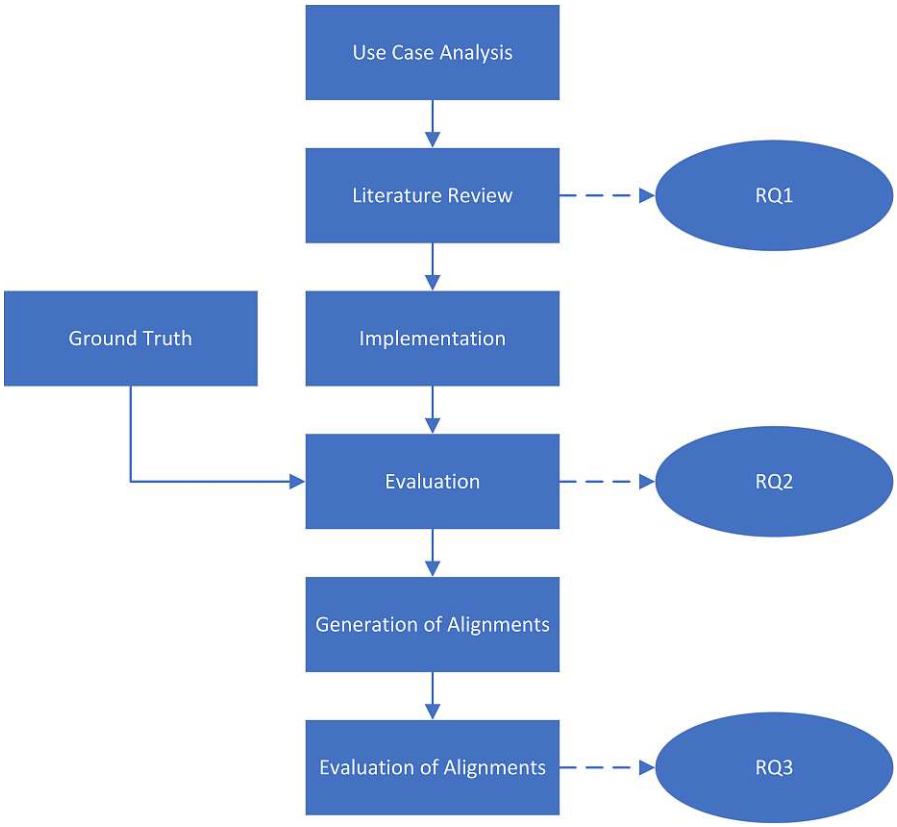

Figure 1.1: Basic research roadmap for this thesis

OA systems from the literature that are applicable to a particular problem domain. To address this issue, the question that needs to be asked is:

RQ1: What are candidate Ontology Alignment (OA) approaches from the literature that are applicable to this problem domain?

To answer *RQ1* two steps were defined: (a) Use Case Analysis and (b) Literature Survey.

Use Case Analysis. During the use case analysis the characteristics of each ontology have been investigated. An example for such characteristics might be the structure of the individual ontologies. For instance, if CPC would have a significantly different structure than CSO, then certain types of OA systems might be excluded. Based on these characteristics requirements for OA systems were formulated.

Literature Survey. To identify OA systems that fulfill the requirements elicited during the use case analysis and are, therefore, feasible to align CPC-CCS and CPC-CSO, a systematic review was conducted. Biolchini et al. [13] proposed a review protocol template for the software engineering domain. This protocol is based on guidelines proposed by Kitchenham [14] and on the protocol example found in [15]. In this work

an adapted/simplified version of [13] was used. The outcome of the literature survey consists of a set of OA systems that are applicable to the problem domain of this thesis.

1.3.2 RI-2: Comparison of Ontology Alignment Systems

The outcome of *RQ1* is a set of OA approaches. When initially identifying these approaches, they had unknown power with regard to this problem domain. Therefore, it is necessary to conduct an evaluation that is problem-specific, thus giving rise to the next research question:

RQ2: What is the performance of the identified Ontology Alignment (OA) approaches on this problem domain?

The elaboration of this research question was divided into three steps: (a) Ground Truths, (b) Implementation, and (c) Evaluation.

Ground Truths. The first step in elaborating *RQ2* consisted in creating two ground truths that are specific to this problem domain - one for CPC-CCS and one for CPC-CSO. To create a high quality ground truth, it was necessary to specify a particular domain, such that the size of the ontologies becomes reasonable for manual inspection. As domain Software Engineering has been chosen. Afterwards, a subset of each ontology was constructed that represents that particular domain. Having obtained a subset for each ontology, i.e. for CPC, CCS, and CSO, it was possible to check which ontological concept of the CPC subset corresponds to which concept from the CCS/CSO subset. Thus, arriving at two reference alignments, one for CPC-CCS subset and one for CPC-CSO subset. These reference alignments in conjunction with the related subsets form the ground truths.

Besides these manually created ground truths, the OA approaches were also compared against each other on the *Anatomy* test case of the OAEI. As the underlying ontologies of the *Anatomy* test case are much larger than the CPC/CCS/CSO subsets, the performance and runtime on this dataset was used to measure the scalability and to have another, problem-independent, performance indication for each approach.

Implementation. During the second step in elaborating *RQ2*, the approaches identified during the literature survey have been implemented. Additionally to the implementation tasks, several of these OA systems have been parameterized to be able to compare the performance of different configurations of the same system.

Evaluation. The implementations from the previous step have then been used during this step to compare the various OA approaches. For that sake, the approaches have been evaluated based on *precision*, *recall*, and *f1-score* - similar to the OAEI [12]. As underlying datasets for the evaluation, the manually created ground truths and the *Anatomy* test case from the OAEI have been used. The outcome of this step consists of performance and runtime indications for each OA approach. Based on these performance

measures, it is then argued which OA systems were eventually employed to generate the final two alignments, i.e. between CPC and CCS and between CPC and CSO.

1.3.3 RI-3: Aligning CCS with CPC and CSO with CPC

To the best of our knowledge, there has been no alignment generated between CPC and CCS, nor between CPC and CSO in any previous work. This research question aims at generating and assessing the quality for an auto-generated CPC-CCS alignment as well as for an auto-generated CPC-CSO alignment. To this extent, the following research question was formulated:

RQ3: What is the performance of the best performing approach on the ground truth when applying it to generate a full alignment between CPC and CCS/CSO?

To address the above research question, two steps have been formulated: (a) Generation of Alignments, and (b) Evaluation of Alignments.

Generation of Alignments. The initial step in elaborating *RQ3* was to apply the approach with the best performance on the CPC-CCS ground truth to align CPC with CCS. Analogously, a full alignment between CPC and CSO was generated. The outcome of this step comprises two alignments, one for CPC-CCS and one for CPC-CSO.

Evaluation of Alignments. Besides the ground truths, there are no gold standards available on which the resulting alignments might have been evaluated. Therefore, in order to evaluate the quality of the alignments, a similar approach was considered as in [16]. The key idea of this evaluation approach is to approximate *precision*, *recall*, and *f1-score* through manually assessing the correctness of randomly selected correspondences of an alignment. The outcome of this step are quality indications for the two final alignments.

1.4 Code Base

To make the descriptions within this thesis as transparent and reproducible as possible, the major documents and all the code base can be found under at the related GitHub repository[5].

In a nutshell it contains the results of the literature review, the ground truths with justifications, the implementation of each OA system, the result of the evaluation on the ground truths, the full alignments between CPC and CCS as well as between CPC and CSO, and the evaluation results of the full alignments. Moreover, a Dockerfile exists for each approach that allows the execution of each approach in a convenient way.

[5] https://github.com/hannesmarcher/msc_thesis

1.5 Structure of the Work

While this chapter introduced the problem domain, the aim, as well as the research approach for this thesis, the remainder of this thesis is structure as follows:

Chapter 2 clarifies the fundamental concepts that are necessary to fully comprehend the subsequent chapters. It defines in a formal way what an *ontology* is, what *ontology alignments* are, and what *ontology alignment systems* are.

Afterwards, Chapter 3 describes the use case (i.e. CPC, CCS, and CSO ontologies) in conjunction with the conducted preprocessing steps as well as eliciting several observations that were relevant for the literature survey.

Chapter 4 presents the procedure of the structured literature review as well as the outcome, i.e. a set of OA systems that are capable of aligning CPC with CCS and CPC with CSO. This set of OA systems is described in detail and represents the major outcome of this chapter. Thus, it answers the first research question *RQ1*.

Chapter 5 delivers all the performances of the elicited OA systems on the ground truths as well as on the third-party dataset *Anatomy*. The obtained results form the answer of the second research question *RQ2*. Based on these results, the chapter concludes with the identification of two suitable OA system, i.e. one for CPC-CCS and one for CPC-CSO.

Chapter 6 evaluates the final alignments and lists various characteristics of the respective alignments. For that sake, the chapter approximates *precision*, *recall*, and *f1-score* for the two final alignments. The results of that chapter address the third research question *RQ3*.

Finally, Chapter 7 concludes the thesis and presents possible directions for future work.

CHAPTER 2

Preliminaries

Chapter 1 introduced the reader to the problem domain, the research issues, and how the research issues are tackled in this work. Before continuing with Chapter 3, which describes the underlying ontologies as well as the conducted use case analysis, we provide a summary of foundational concepts as well as define the terminology that is used in the remainder of this thesis.

Section 2.1 defines what an ontology is. Next, Section 2.2 defines formally what an Ontology Alignment (OA) is, as well as justifying the need for having OAs. Section 2.3 provides the reader with a high level overview about OA systems in conjunction with a classification scheme.

2.1 Ontologies

An ontology may be formally defined as [9, 17]:

$$O = (C, DP, OP, I) \tag{2.1}$$

where:

- C represents a set of classes
- DP stands for data properties that are assigned to classes
- OP is a set of object properties that describe the relation between the classes
- I is a set of instances

All ontologies in this thesis are expressed through Web Ontology Language (OWL) that in turn is built upon the Resource Description Framework (RDF). In a nutshell, RDF is essentially a data model used to describe directed graphs through triples [18]. Each triple consists of a *subject* (node), a *predicate* (edge), and an *object* (node) [18]. Subjects and

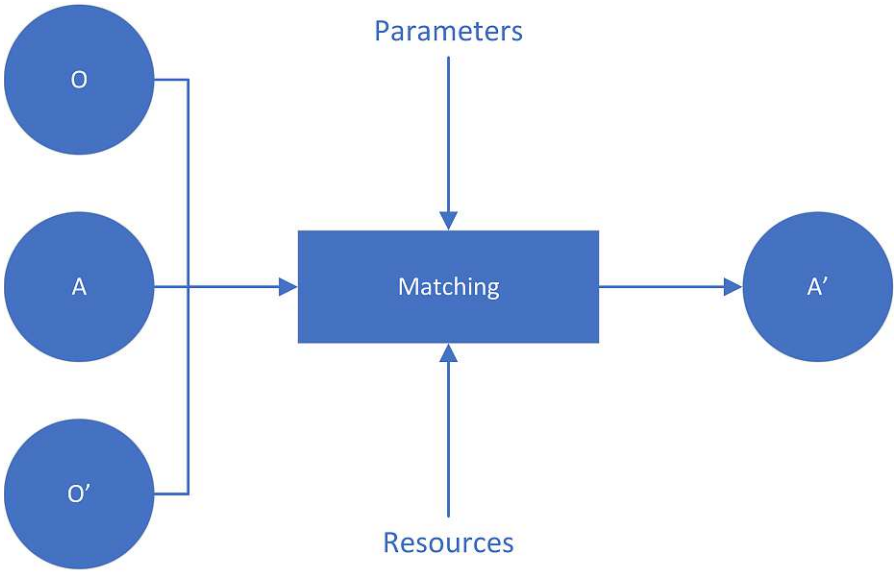

Figure 2.1: The matching process [9]

predicates are typically represented through URIs, while an object may either consist of an URI or a literal [18]. OWL is a superset of RDF that provides many additional capabilities, such as cardinality constraints or disjointness relations [19]. Further, OWL has been widely used in many domains [20, 21, 22].

2.2 Ontology Alignments

The semantic web is a heavily distributed and open system, therefore heterogeneity is inevitably the case [9]. The aim of OA, i.e. matching of ontologies, is to reduce this heterogeneity [9].

There exist many formalization for the OA process and its result [9]. In [9], the matching process is seen as a function f that returns an alignment A' from two input ontologies O and O' (defined in definition 2.1), an initial (possibly empty) alignment A, a set of parameters P, and a set of oracles and resources R:

$$A' = f(O, O', A, P, R) \tag{2.2}$$

Figure 2.1 illustrates definition 2.2.

The outcome of the matching process is an Ontology Alignment (OA) between the two ontologies O and O'. An OA may formally be defined as a set of 4-tuples (correspondence) [23]:

$$< e, e', r, n > \tag{2.3}$$

where

- $e \in O$ and $e' \in O'$ represent the two matched entities that may be classes, instances, properties, etc.
- r is the relationship that holds between e and e', typically either *equivalence* ($=$), *subsumption* ($\subset$), or *disjointness* ($\perp$)
- n indicates the confidence for the mapping, i.e. a floating point number between 0 and 1

Such a 4-tuple is referred to in this thesis as a *correspondence*, a *mapping*, or a *match*. A collection of such 4-tuples is referred to as an *alignment*. Further, this thesis is restricted to alignments between classes and equivalence mappings, i.e. e representing always a class, and r is always equal to $=$. Also, the term *concept* is here used interchangeably with the term *class*.

Another important distinction that arises from definition 2.3, is the distinction between one-to-one alignments and many-to-many alignments. As the name already suggests, in a one-to-one alignment each ontological concept from O (source ontology) as well as from O' (target ontology) appears in at most one correspondence. In a many-to-many alignment, on the contrary, each entity $e \in O \cup O'$ may appear in multiple correspondences. In a similar way, one can also define one-to-many and many-to-one alignments.

2.3 Ontology Alignment Systems

To generate OAs as defined in definition 2.3, a branch of research has been formed that is about the design of approaches that are capable of generating alignments between ontologies. Note, this work uses the terms *matcher*, *OA approach*, *OA technique*, *OA solution*, *OA system* interchangeably. The aim of this section is to provides a high level overview about different types of OA systems.

Rahm and Bernstein [24] proposed a classification scheme for OA approaches that is widely used [25]. OA systems are distinguished along the following dimensions: schema-level matching vs. instance-level matching and element-level matching vs. structure-level matching [24, 9, 26, 25].

Schema-level OA systems use exclusively schema information about an ontology [24, 25], i.e. C, DP, and OP from definition 2.1. Instance-level OA systems, on the other hand, employ information about the instances [24], i.e. I from definition 2.1. Note this thesis is limited to the application of schema-level OA systems. Moreover, it is limited to the application of schema-level matchers in the context of classes that generated equivalence mappings. Thus, for any e, e' and r (definition 2.3) it always holds that $e, e' \in C$ and r is equal to $=$.

Schema-level OA systems as well as instance-level OA systems may further be divided into element-level and structure-level matching approaches [24, 9, 27, 25]. Element-level matching is about matching entities while not considering information about other entities within the same ontology [24, 9, 25]. On the other hand, as the name suggests, in

structure-level matching the OA systems map combinations of entities from one ontology onto a combination of entities from a second ontology, thus using structural information [24, 9, 25].

Element-level matchers can further be sub-divided into string-based and language-based approaches. String-based approaches match entities based on the similarity of their respective labels/descriptions, while language-based approaches typically employ natural language processing techniques, such as the usage of word embeddings. [9]

Very often OA systems employ a combination of element-level matching and structure-level matching. For example, the State-of-the-Art (SOTA) matcher LogMap matches entities based on their lexical information, but also considers the ontologies' structures [28].

A further term that frequently appears in the literature about OAs is *meta-matching* resp. *meta-matcher*. Martinez-Gil and Aldana-Montes [29] defined meta-matching as *it is the technique of selecting the appropriate algorithms, weights and thresholds in ontology matching scenarios in order to obtain a satisfactory alignment between ontologies.* Thus, informally speaking: it describes the set of approaches that automatically configure weights, thresholds and select appropriate matchers as well as aggregation procedures for the respective use case.

CHAPTER 3

Use Case

The use case of this thesis consists of three ontologies, i.e. CPC, CCS, and CSO, whereby the former one shall be aligned with the latter two. While these three ontologies were already roughly outlined in Chapter 1, this chapter introduces the characteristics of the three ontologies in more detail in conjunction with the conducted preprocessing steps. The chapter concludes with observations originating from the characteristics of each ontology. Identifying these challenges is the first building block in answering the first research question:

RQ1: What are candidate Ontology Alignment (OA) approaches from the literature that are applicable to this problem domain?

The structure of this chapter is as follows: The upcoming Sections 3.1, 3.2, and 3.3 describe the properties and conducted preprocessing steps for CCS, CSO, and CPC, respectively. Section 3.4 describes then the observations that emerged from a closer look at the ontologies.

3.1 ACM Computing Classification System

The 2012 ACM Computing Classification System (CCS) is a poly-hierarchical ontology that serves as a classification system for the computing field [30]. The 2012 version of CCS replaces the version from 1998 that represented the de-facto standard classification system for the computing field [30, 31]. The 2012 version of CCS can be inspected through a visual web tool [1] or it may also be downloaded as a Simple Knowledge Organization System (SKOS) instance for local usage [2].

Before continuing with this section, here a brief introduction to SKOS: SKOS is an application of RDF that enables the management and representation of many knowledge

[1] https://dl.acm.org/ccs
[2] https://dl.acm.org/pb-assets/dl_ccs/acm_ccs20121626988337597.xml

organization systems, such as thesauri, classification systems, and taxonomies. The SKOS data model is formally defined as an OWL ontology, whereby SKOS data can be expressed as RDF triples. [32]

The SKOS instance of CCS utilizes the following subset of SKOS-specific semantic relationships: *skos:broader*, *skos:hasTopConcept*, *skos:narrower*, *skos:prefLabel*, *skos:topConceptOf*. Each of the CCS concepts is represented as an instance of the class *skos:Concept*. Moreover, the whole CCS scheme is an instance of *skos:ConceptScheme*. In total there are 2113 instances of type *skos:Concept*, thus 2113 classification entries.

Preprocessing. As mentioned in Chapter 2, this work is restricted to OA systems that generate schema-level alignments. Therefore, it is necessary to transform this instance-level representation of CCS into a schema-level representation without losing information with respect to the capabilities of the OA systems. To achieve this goal the following transformation rules have been employed:

- *skos:broader* results in *rdfs:subClassOf*
- *skos:hasTopConcept* can be ignored since it serves only to conveniently find the entry points of the hierarchies
- *skos:narrower* can be ignored since it is the same as *skos:broader*, except in the other direction
- *skos:prefLabel* results in *rdfs:label*
- *skos:topConceptOf* can be ignored as it is the same as *skos:hasTopConceptOf*, except in the other direction
- *skos:Concept* results in *owl:Class*
- *skos:ConceptScheme* can be ignored

3.2 Computer Science Ontology

In contrast to CCS, the Computer Science Ontology (CSO) is an automatically generated ontology using the *Klink-2* algorithm [33] on the Rexplore dataset [34] [6]. Due to its auto-generated nature it is feasible to construct a large-scale ontology covering a large variety of research areas that are mainly about, but not restricted to, computer science [6]. Further, when new data is available, updating the ontology is easy as the *Klink-2* algorithm can simply be re-executed [6].

The data model of CSO is an extension of the BIBO ontology [3] that in turn builds upon SKOS [6]. Note that during this thesis version 3.3 of CSO is used. As semantic relationships in version 3.3 we have: *cso:contributesTo*, *cso:preferentialEquivalent*, *cso:relatedEquivalent*, *cso:superTopicOf*, *schema:relatedLink*, *rdf:type*, and *owl:sameAs*.

[3] http://purl.org/ontology/bibo/

[6, 35]. Further, in version 3.3 of CSO there exist 23790 individuals, i.e. research areas, that, in turn, are of type *cso:Topic* which represents the only class in the ontology.

As it is the case with CCS, also the CSO ontology can be utilized from a web portal [4], but it can also be downloaded for local usage [5].

Preprocessing. To transform the instance-level ontology to a schema-level representation, the following mapping procedures have been performed:

- *cso:contributesTo* indicates that the research outputs of one topic contributes to another [35]. This relation can be ignored since OA systems typically cannot handle such relationships.
- *cso:relatedEquivalent* is added as additional *rdfs:label* since it is referred to as *alternative label of* on the web portal [35].
- *cso:preferentialEquivalent* is used to indicate the main label for a cluster of concepts that are linked with each other through *cso:relatedEquivalent* [35]. This relation is treated equivalently to *cso:relatedEquivalent*.
- *cso:superTopicOf* results in *rdfs:subClassOf*.
- *schema:relatedLink* points to external sources, namely either to a Wikipedia article or to a Microsoft Academic article [35]. This relationship is ignored.
- *owl:sameAs* points to entries from other knowledge graphs that refer the same concept [35]. If it points to a Dbpedia/Wikidata/Yago entry, then an additional *rdfs:label* is generated through using the label provided by the respective ontological entry. If it points to Freebase, it is ignored as the corresponding links are not valid in the majority of the cases.
- Each subject that appears in a triple with *rdfs:type* as the predicate and *cso:Topic* as the object, is represented as *owl:Class*.

Note that there exist loose subjects, i.e. subjects that are not of type *cso:Topic* and are not integrated into the hierarchy, i.e. have neither children, nor parents. These loose subjects have been ignored during preprocessing.

After conducting all the above listed preprocessing steps, there were 14290 research topics left. This number is much smaller than the original 23790 research topics since all the external sources, such as Dbpedia, Wikidata, etc., are not included as nodes anymore.

3.3 Cooperative Patent Classification System

Cooperative Patent Classification (CPC) is an ontology that is developed jointly by the European Patent Office (EPO) and the United States Patent and Trademark Office

[4] https://cso.kmi.open.ac.uk/home
[5] https://cso.kmi.open.ac.uk/downloads

(USPTO) with the aim to obtain a common and internationally compatible classification system for patents [36]. CPC consists of nine sections:

- A: Human necessities,
- B: Performing operations; transporting
- C: Chemistry; metallurgy
- D: Textiles; paper
- E: Fixed constructions
- F: Mechanical engineering; lighting; heating; weapons; blasting engines or pumps
- G: Physics
- H: Electricity
- I: General tagging of new technological developments; general tagging of cross-sectional technologies spanning over several sections of the International Patent Classification (IPC); technical subjects covered by former USPC cross-reference art collections and digests.

These nine sections are in turn divided into classes, subclasses, groups, and subgroups. Overall there exist 250 thousand classification entries. [8]

The whole CPC ontology is provided in *N-TRIPLES* [6]. *N-TRIPLES* is a line-based, plain text format for encoding RDF graphs [37]. The *N-TRIPLES* CPC file contains 18 different relationship types and each concept represents either a CPC entry or an IPC entry. Out of these 18 relationships, the next paragraph mentions only those relationships that are relevant for the sake of this thesis.

Preprocessing. The following list outlines the steps taken to transform all the triples into an OWL ontology that has the same structure as the preprocessed versions of CCS and CSO:

- The first step is to remove all tuples containing predicates not equal to either: *cpc:title*, *cpc:guidanceHeading*, or *skos:broader*.
- The second step is to remove all triples that point to IPC concepts. This operation does not remove any CPC related information as all the remaining $< subject, predicate, object >$ triples that contain IPC concepts have IPC concepts, both, as their subject as well as their object. Thus, after this operation the remaining triples comprise only CPC items.
- Next, to create a schema-level ontology the following procedure is employed:
  - Each subject where its URI indicates that the respective subject represents a classification entry (e.g. *http://data.epo.org/linked-data/def/cpc/G03G9-0815*) is mapped to *owl:Class*.

[6] https://data.epo.org/linked-data/download/

  - *cpc:title* results in an *rdfs:label* annotation in the related class.
  - *cpc:guidanceHeading* results in an additional *rdfs:label* annotation in the related class since this relation indicates additional information.
  - *skos:broader* results in *owl:subClassOf* in the related class.
- Further necessary preprocessing steps:
  - Several labels contain reserved XML characters, thus escaping these characters is necessary
  - If a label contains simple brackets, then everything between these brackets (including the bracket itself) is removed, since inside brackets we have only information about other CPC concepts, e.g. entry *G06F 8/314* has as label *Parallel programming languages (G06F 8/313 takes precedence)*
  - All labels of sub-groups and below are enclosed with curly braces. These curly braces are removed

Due to the fact that the resulting ontology is still extremely large with many branches of completely unrelated topics (when compared to the domain of computer science), the following sections were removed: A, B, C, D, E, F, and H. It may be assumed that these sections have the least in common with computer science. Thus, the only remaining sections are G and I. The major benefit of this size reduction is two-folded: (a) without size reduction aligning CPC with CCS and CSO would probably not be feasible on a standard laptop (more on the hardware specifications follow in Chapter 5), (b) to evaluate the final alignment (more information regarding the final evaluation follows in Chapter 6) between CPC-CCS and CPC-CSO a manual assessment is performed. However, such a manual assessment would not be possible if the reviewer (in this case, the author of this thesis) is unfamiliar with the domain.

After conducting all the above described preprocessing steps, there were 53849 classification entries left in the ontology.

3.4 Observations

The previous sections describe the meta information for each ontology in conjunction with the conducted preprocessing steps. After preprocessing, each ontology is represented in the same format, i.e. as a schema-level ontology where each concept is of type *owl:Class* and has one or more *rdfs:label* properties assigned to it, additionally as relationships between the classes we have only *owl:subClassOf*.

After having an in-depth look at the content of each ontology, a few observations came up that might affect the performance of a matcher (depending on the functionality of the respective matcher):

- *Obs1*: CPC has a deviating structure when compared to CCS or CSO. For example, the CPC concept *Object-oriented languages* (*G06F 8/315*) has as hierarchy

Arrangements for software engineering → *Creation or generation of source code* → *Programming languages or programming paradigms* → *Object-oriented languages*, while the corresponding CCS concept resides in the following hierarchy *Software and its engineering* → *Software notations and tools* → *General programming languages* → *Language types* → *Object oriented languages*.

- *Obs2*: CPC is especially large.
- Observations regarding CPC naming scheme:
  - *Obs3*: The classification entries have rather long labels that typically need additional context to make sense of, e.g. the classification entry *G16Z* has as its label *information and communication technology [ict] specially adapted for specific application fields, not otherwise provided for*.
  - *Obs4*: CPC often has a rather unconventional naming scheme, especially when inspected through the lenses of computer science. For example, *Creation or generation of source code* (*G06F 8/30*) refers to all conceptual steps of converting an abstract representation of a software system into program code, while from a computer science perspective the part *generation of source code* merely refers to automatically generating source code.
  - *Obs5*: Several classification entries in CPC have labels that would require very complex reasoning to make sense of. For example, the CPC classification entry *G06F 1/00* has as its title *details not covered by groups [cpc: g06f3/00] - [cpc: g06f13/00] and [cpc: g06f21/00]*
- *Obs6*: It is quite common that there are concepts in CPC that have a very similar/exact label to another concept in CCS or CSO, but with significantly deviating hierarchies (being in completely different knowledge branches), thus not representing the same underlying notion. For example, when comparing the hierarchies of the CPC concept *Type checking* (*G06F8-437*) and *typechecking* of CSO, then it immediately becomes clear that they can hardly be considered as equivalent. For CPC we have: *Arrangements for software engineering* → *Transformation of program code* → *Compilation* → *Checking; Contextual analysis* → *Semantic checking* → *Type checking*. For CSO, on the other hand, we have: *computer science* → *object oriented programming* → *type systems* → *typechecking*. I.e.: The CPC classification entry is about type checking in the context of compilers, while the CSO concept only refers to type checking in the field of object oriented programming.
- *Obs7*: When comparing concepts from CPC to concepts from CCS/CSO, it becomes evident that if a concept from CPC has a correspondence in CCS/CSO, then it is very often the case that there exist multiple candidate matching partners.

Based on these observations, there arise a few implications. *Obs1* indicates that element-level matchers should be preferred over structure-level matchers. *Obs2* underlines the importance of runtime efficiency on this use case. *Obs3*, *Obs4*, *Obs5*, and *Obs6* indicate that employing elements of structure-level matching might bring advantages. *Obs3*

indicates that language-based matchers might be very beneficial, due to their capabilities of making sense out of sentences and their semantic reasoning capabilities. Based on *Obs7*, it might be the case that many-to-many matchers outperform one-to-one matchers on this particular use case. Of course, these implications partly conflict each other, i.e. there exists no matcher that fulfills all of the above outlined features. However, we can require the final set of OA systems to contain:

- *Requirement 1*: At least one purely element-level matcher.
- *Requirement 2*: At least one matcher that employs concepts of structure-level matching.
- *Requirement 3*: At least one language-based matcher.
- *Requirement 4*: At least one matcher capable of generating many-to-many alignments.
- *Requirement 5*: At least one matcher that is very time efficient, otherwise it may turn out that no matcher is capable of aligning CPC-CCS and CPC-CSO due limited computational resources.

In this way, it is possible to compare whether having one of those features brings advantages when aligning CPC-CCS and CPC-CSO.

CHAPTER 4

Ontology Alignment Approaches

The previous chapter gave a detailed overview about the individual ontologies, the related preprocessing steps, as well as identifying desired features regarding the OA systems. This chapter describes the steps that have been taken to arrive at OA systems that may be considered appropriate for the problem domain of this thesis, as well as the OA systems itself. Thus, it answers the first research question:

RQ1: What are candidate Ontology Alignment (OA) approaches from the literature that are applicable to this problem domain?

This chapter starts, in Section 4.1, by describing the structured literature review that was performed to find appropriate OA systems. The outcome of the literature survey is a set of OA systems that are then described in more detail in Section 4.2. It was ensured that these OA systems fulfill the requirements identified in the previous chapter. The chapter concludes with Section 4.3 which delivers a brief summary and discusses limitations. The identified OA systems were then implemented and the results of the experiments on the ground truths are then discussed in Chapter 5.

4.1 Literature Survey

As Section 1.3 states, the conducted literature survey follows a simplified version of the review protocol template proposed in [13] that, in turn, is based on the guidelines identified in [14].

4.1.1 Literature Review Overview

Biolchini et al. [13] described that from an operational point of view systematic reviews comprise five stages. These five stages are outlined next (for more detailed insights, see [13]):

Figure 4.1: Overview of literature survey - steps based on descriptions in [13]

The first stage is related to the problem formulation and comprises the construction of criteria by which relevant and irrelevant literature can be distinguished from each other, i.e. criteria by which literature is selected or rejected. The second stage is about data collection. Here, the major point is to define which sources shall be used, i.e. how literature can be found that corresponds to the criteria found during the previous stage. For example, it includes determining the search engines and search queries. The third stage is related to data evaluation. During this stage quality criteria are applied to filter the literature found during the proceeding stage. During the fourth stage, the found literature is then inspected and inferences are drawn from the collected data. The fifth and final stage is about conclusions and decisions on which literature shall be included in the systematic review report (a report comprising all the found literature).

These five stages can further be summarized into three steps. The stages one, two and three (partially) can be grouped into the *Review Planning* phase, the stages three (partially) and four into the *Review Execution* phase, and the stage five into the *Result Analysis* phase. Figure 4.1 illustrates the process. For convenience and simplicity there are no indications for iterations in Figure 4.1, however, note that many stages indeed involve iterations and a possible fallback to previous stages is also common.

The next subsections detail the individual steps that have been conducted during the elaboration of this thesis with regard to the three phases illustrated in Figure 4.1.

4.1.2 Review Planning

As explained previously, the purpose of the *Review Planning* phase is to determine the aim, the data collection strategy, as well as the definition of selection criteria.

Aim & Selection Criteria The aim of this survey was to find literature about recently developed OA approaches. To find a reasonable set of of publications, requirements were formulated that are listed below. Note: The letters inside the simple brackets at the end of each requirement indicate how difficult it is to determine the respective requirement. (S) means that it can already be satisfied during the search process - essentially all search engines allow to query for certain dates. (A) mean that the respective requirement can typically be decided after having read the title + abstract. In some cases, to determine (A), it is necessary to have an in-depth look at the related publication.

- The related publication is from 2018 or newer (S)
- The publication proposes a new OA system that is not tailored to a specific domain (except if the domain is very similar to the domain of this thesis). Examples of unrelated domains often tackled by domain-specific OA systems include biomedicine or sensor ontologies. (A)
- The proposed OA system is an unsupervised approach - necessary since no training data exists (A)
- The publication is not about the optimization of existing alignments. Instead, it presents an OA system that is applicable independently from any pre-existing alignment. (A)
- The proposed approach is fully automatic, e.g. no interactive approaches (approaches that require the involvement of a human) are allowed. (A)
- The related publication reports performance measures of the system on a publicly available dataset, such as on an OAEI track. (A)
- The reported performance is compared against the performance of a SOTA matcher to ensure comparability between the individual approaches. (A)
- The system is able to match ontologies at schema-level, e.g. no instance-level matchers are allowed. (A)
- The approach must deliver equivalent correspondences, e.g. no subsumption correspondences. (A)

Sources As search engine to find scientific literature *Google Scholar* [1] was used. Table 4.1 shows the search string that was used to query *Google Scholar*. Note that *allintitle* was used to ensures that only publications are retrieved that contain the string *ontology alignment* or *ontology matching*. This restriction was indispensable since, otherwise, several tens of thousands of publications would have been found that are, in their majority,

[1] https://scholar.google.com

Figure 4.2: Overview of literature filtering process

not suitable for this thesis. Naturally, this imposes a limitation on this literature survey since some publications describing suitable OA approaches might have been missed.

Engine	Search Query	Publication Years	#Results
Google Scholar	allintitle: ontology alignment OR matching	2018 - 2022	332

Table 4.1: Search string

4.1.3 Review Execution & Results Analysis

Review Execution. Querying *Google Scholar* using the search string described in Table 4.1 delivered 332 publications. The corresponding meta-data of each publication was then downloaded as a bib file. The publications were then evaluated based on the criteria defined in the previous subsection. Figure 4.2 depicts the process.

After reading the titles of all 332 publications found on *Google Scholar*, 252 publications

were rejected because of the title. 139 publications were rejected because the title did not suggest a new OA system (e.g. publications were about surveys, datasets for OA evaluation, optimization approaches, repair strategies, ...). Further, 80 were rejected because the related OA systems were tailored to specific domains. 16 publications were about interactive matching techniques. 13 titles indicated supervised approaches. Three publications were about instance-level alignment. One publication was not in English.

Next, the abstracts of the 80 remaining publications were skimmed. Doing so, seven were rejected because the authors described a supervised approach, 26 publications were rejected because they were not related (e.g. interactive matching approaches, etc.), 23 were rejected because of evaluation procedures/outcomes (e.g. the approach was evaluated on a private dataset or the performance on a public dataset showed that the approach is not competitive), further two publications were not available for download. The remaining number of publications was then 21.

Out of the remaining 21 publications, there were multiple papers describing the same approaches - leading to 17 distinct approaches.

After having a more detailed look at the publications, it became clear that six publications relied in some way or the other on training data/a partial reference alignment - thus were supervised. Another publication had to be rejected since the authors compared their system against non-SOTA systems implying that the results were not considered expressive enough.

Result Analysis. Table 4.2 shows the remaining 10 approaches with the related publications. Note the names LXLHMeta, LJLEvolutionary, LPGrasshopper, and LSSJ-Fireworks are not the official names since there are no names for the respective OA systems in the corresponding publications. Instead, these names are composed of the authors' initial letters followed by a peculiarity of the respective approach. As it is not feasible to implement all these approaches, the last column in the table indicates whether the related approach has been further considered for implementation and, eventually, for the experiments on the ground truths. Reasons for inclusion/exclusion are provided in the upcoming paragraph.

One of the major outcomes of Chapter 3 was a list of requirements towards the final set of OA systems. These requirements have been named as *Requirement 1* - *Requirement 5*. According to *Requirement 1* at least one purely element-level matcher must be contained, examples hereof are: LXLHMeta and AML. *Requirement 2* imposes that at least one matcher is contained that employs concepts from structure-level matching, examples hereof are SANOM and LogMap-ML. *Requirement 3* demands that at least one language-based matcher is contained. Examples hereof would be BERTMap or OntoConnect. *Requirement 4* is about the containment of a many-to-many matcher. Hereof one can name LogMap-ML. When considering the last column in Table 4.2, it becomes clear that the first four requirements are satisfied by the set of approaches that were considered for implementation (i.e. "Yes" in the last column). *Requirement 5* (time efficiency), on the

Publication	Name	Impl.
[16]	LogMap-ML	Yes
[38]	BERTMap	Yes
[39]	OntoConnect	Yes
[17]	SANOM	Yes
[40]	PoMap++	No
[41]	DeepAlignment	No
[42]	LXLHMeta	Yes
[43]	LJLEvolutionary	Yes
[44]	LPGrasshopper	No
[45]	LSSJFireworks	No

Table 4.2: Outcome of literature review

other hand, is more difficult to determine at this stage since most of the publications do not outline efficiency values. *Requirement 5* can be seen as a requirement towards approach selection after conducting the experiments in Chapter 5.

In particular, the approaches PoMap++, DeepAlignment, LPGrasshopper, and LSSJFireworks have been rejected. PoMap++ and DeepAlignment have been rejected because there are already sufficient machine learning centered solutions present, e.g. LogMap-ML, BERTMap, etc. LXLHMeta, LJLEvolutionary, LPGrasshopper and LSSJFireworks are all about meta-matching. To not heavily overweight meta-matching in the final sample of systems, two have been rejected.

The result of this literature review is summarized as a bib file and can be found in the repository of this thesis. The field "group" of each bib entry indicates at which point the related publication has been excluded or if the publication has been accepted. A convenient way to inspect the bib file is through JabRef[2].

The goal of this literature survey was to explore and select recent approaches to further examine their applicability regarding the problem domain of this thesis. Besides these novel approaches, the SOTA systems LogMap [28] and AML [11] were also used. In addition a baseline matcher was implemented as well.

4.2 Identified Ontology Alignment Systems

The remainder of this chapter explains all the considered OA approaches in detail.

4.2.1 StringEquiv (Baseline)

Similar to the OAEI [12], the matcher StringEquiv is also used here as baseline matcher. The functionality consists of simply comparing each concept's lower-cased labels with the

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

lower-cased labels of other concepts. If a label of a concept from the source ontology is identical to a label of a concept in the target ontology, then the respective concepts form a correspondence in the resulting alignment. Note this implies that, in case a concept has multiple labels, only one of the labels must match.

4.2.2 LogMap

Jiménez-Ruiz and Cuenca Grau [28] proposed LogMap in 2011 that was, to the best of the LogMap-authors' knowledge, the first approach able to handle large instances (tens or even hundreds of thousand classes) and to spot logical inconsistencies that may occur after aligning two ontologies. When considering the results of the OAEI conference of 2021 [12], one can see that LogMap still shows SOTA performance on various tracks. Also several recent OA approaches use concepts from LogMap, e.g. [38] and [16]. LogMap is therefore an essential building block for recent OA research and, as a consequence, very influential. The LogMap OA system can be roughly divided into 4 parts that are sketched in Figure 4.3.

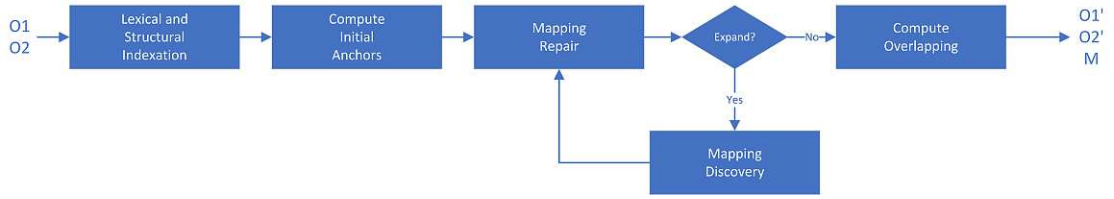

Figure 4.3: LogMap in a nutshell [28]

The first part can be seen as a preparation phase. During this phase an inverted lexical index is created that will be used during the next step to efficiently compute a set of anchor mappings. LogMap uses sets consisting of the tokens for each label and their variations, that are obtained by external sources such as the UMLS lexicon [46] or Wordnet [47], as the key for the inverted index. In addition to this lexical indexation, a structural index is built as well. This structural index makes use of additional relationships within an ontology, e.g. it exploits information about disjoint classes.

As next part a set of anchor mappings are created through simply intersecting the inverted indexes of the input ontologies, i.e. the outcome of this step consists of an initial set of high confidence mappings.

The core of LogMap is then an iterative process, i.e. it alternates between: *mapping repair* and *mapping discovery*. During mapping repair logical inconsistencies are identified and repaired using a greedy diagnostics algorithm. The process of mapping discovery is about extending the alignment. For each anchor semantically related classes are identified. This step follows the principle of locality, e.g. if the two classes C_1 and C_2 form a correct correspondence, then the semantically related classes of C_1 are likely to be related to the semantically related classes of C_2. The newly found classes form the new set of anchors.

The algorithm continues with this iterative process until, during mapping discovery, no new context is expanded.

As the final step, LogMap computes a fragment of each ontology that corresponds to the *overlapping* between both ontologies. Note that this output is typically only relevant for curators (experts that manually maintain ontologies) and can, therefore, be neglected for the sake of this thesis.

Figure 4.3 illustrates and summarizes the individual steps, i.e. the approach starts with the creation of lexical and structural indices, followed by the computation of initial anchors, next there is an iterative process of mapping repair and mapping discovery, additionally, as the last step, LogMap computes the *overlapping* between both ontologies.

4.2.3 AML

When considering the performance of the OAEI conference of 2021 [12], another traditional approach shows remarkable performance - namely AgreementMakerLight (AML) (proposed by Faria et al. [11] in 2013). AML builds upon the pre-existing framework AgreementMaker [48]. At the time when AML was published, AgreementMaker was one of the leading OA systems [11]. The major drawback of AgreementMaker was that the system was never designed to match ontologies with more than a few thousand concepts [11]. Due to this limitation, AML embeds several novelties and improvements when compared to the original AgreementMaker system.

Note that the term "framework" is here chosen consciously as the system is not about a single OA system, instead it employs multiple OA systems. More precisely, AgreementMaker resp. AML are frameworks that combine multiple matchers to obtain a final high quality alignment.

The core framework of AML includes two major components: the ontology loading module and the ontology matching module. The ontology loading module is sketched in Figure 4.4, it is concerned with loading the ontology files and parsing them into ontology objects. Figure 4.5 shows the ontology matching module that uses the generated ontology objects and generates the alignment by combining one or more matching algorithms and one or more selection steps.

As is visible in Figures 4.4 and 4.5, there are three data structures, i.e. the Lexicon, the RelationshipMap, and the Alignment. The former two data structures represent the ontology objects and are employed during the matching process. The latter stores the generated alignment produced by one or more matchers.

As described in a previous paragraph, AML essentially emerged from the application of various improvements with respect to the original AgreementMaker system. The improvements can be split into three categories: (a) improvements towards the data structures, (b) improvements in the ontology loading module, and (c) improvements in the ontology matching module.

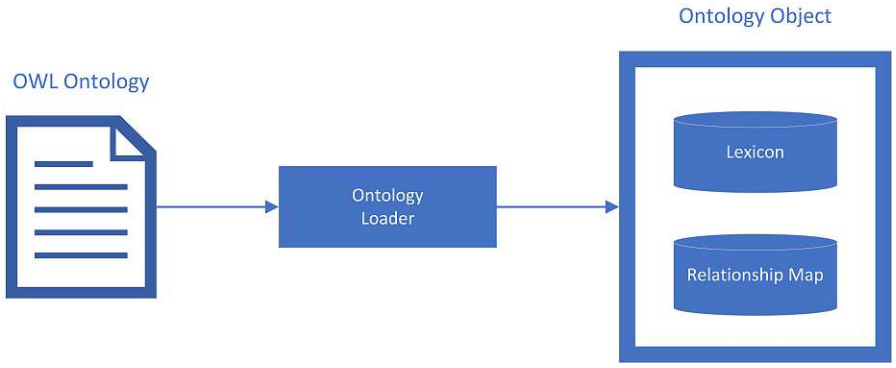

Figure 4.4: The AML ontology loading module [11]

The data structures have been improved through exclusively representing them as internal structures, whereas, in the case of AgreementMaker, they were tied to the ontology loading API Jena2. This makes AML more flexible. Additionally, all the data structures were all optimized for a more economical memory usage.

The ontology loading module is improved through restricting the loading to necessary parts of ontologies resp. omitting all unnecessary parts.

Regarding the ontology matching module: As in AgreementMaker there are three components, i.e. Matchers, Selectors, and Alignment - see Figure 4.5. In contrast to AgreementMaker, AML divides the matchers into primary and secondary matchers. Primary matchers rely on HashMap cross-searches and are therefore very efficient. Secondary matchers, on the other hand, are matchers that make non-literal comparisons between the concepts and, thus, require that each concept of the first ontology is compared against every concept of the second ontology. Due to the fact that secondary matchers cannot match large ontologies efficiently, it is stipulated that secondary matchers only match concepts in the vicinity of previously aligned classes. While AgreementMaker combines the output (=alignment) of each matcher through a linear combination of the weights associated with each correspondence in a given alignment, this is, due to the improved data structures, not possible in AML. AML simply joins the individual alignments and only keeps the correspondences with highest weight in case of duplicates. Selectors are responsible to trim alignments. More precisely, selectors exclude correspondences below a given threshold and constructs a one-to-one mapping from a many-to-many mapping. In AgreementMaker this is conducted through finding the maximum weighted bipartite mapping, however, in AML this step is done through a heuristic. The implemented matchers of AML are derived from existing matchers in AgreementMaker.

4.2.4 SANOM

Mohammadi et al. [17] proposed the OA system SANOM that utilizes Simulated Annealing (SA) as its core strategy. SANOM has participated in the OAEI conference of 2019 [49] where it obtained decent results on the Anatomy test case.

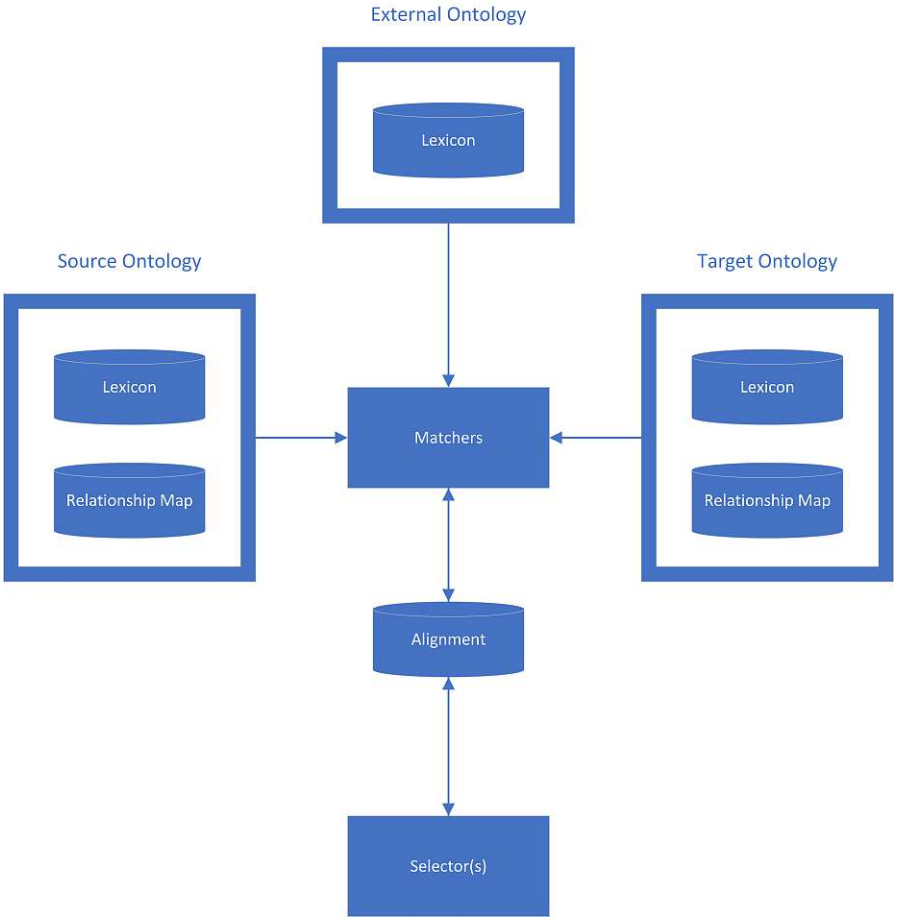

Figure 4.5: The AML ontology matching module [11]

In a nutshell, the idea behind SA is to approximate a global optimum of an objective function f through a probabilistic procedure. The probability is computed through a function that is dependent on an artificial temperature. The artificial temperature is high at the beginning and decreases after each iteration. The probability determines the likelihood for exploration, i.e. accepting worse solutions in the search space. Due to the fact that the temperature is decreased after each iteration, early iterations tend to be more exploratory, while later iterations tend to put more emphasis on intensification. The method is based on the concepts introduced in [50].

Essentially, to be able to apply SA to the OA problem, an objective function as well as a state representation need to be defined.

Mohammadi et al. [17] defined the objective function (fitness function) as $f(c) = f_{string}(c) + f_{structural}(c)$ whereby c represents a correspondence between two concepts, $f_{string}(c)$ is a lexical similarity of the correspondence c, $f_{structural}(c)$ represents a structural similarity of the correspondence c. In this work weights and a second structural

similarity measure are added. Thus, the objective function becomes:

$$f(c) = w_1\ f_{string}(c) + w_2\ f_{structural}(c) + w_3\ f_{structural_2}(c)$$

In the publication, the authors propose to utilize Soft Term Frequency (TF)-Inverse Document Frequency (IDF) as its string similarity function $f_{string}(c)$. TF-IDF is a very popular strategy in the field of information retrieval [51, 17]. It measures the similarity between a word w and a document d within a corpus of documents D. The measure is high for a particular document if the word has a high frequency within that document and is, at the same time, contained only in a few $d' \in D$ for $d' \neq d$. Mohammadi et al. [17] utilized a derived TF-IDF measure that is applicable to compute the similarity between two documents. This TF-IDF measure is high if both documents share expressive words (i.e. words with high TF-IDF values) and is low otherwise. Soft TF-IDF, on the other hand, is derived from the concepts behind this TF-IDF measure. The difference is that, instead of considering only string-equivalent words in both documents, words are considered as equivalent if some similarity measure between both words is above a given threshold. For that purpose, the authors employ two similarity measures, namely Jaro-Winkler and the WordNet based measure *Wu & Palmer* [52]. Note: In this work, instead of Jaro-Winkler, Levenshtein distance was used since that metric was found to work better on the problem domain of this thesis.

The structural similarity function $f_{structural}(c)$, on the other hand, is computed through $f_{structural}(c) = f(s)$ where c is again a correspondence, i.e. it comprises the concepts c_1 and c_2, for which the similarity shall be computed, and s comprises s_1 and s_2 whereby s_1 is the parent of c_1 and s_2 is the parent of c_2. Thus, if the parents of c_1 and c_2 are determined to be similar, then $f_{structural}(c)$ of the child concepts is high as well. During the re-implementation of this approach, another structural similarity measure has been found to be beneficial, namely using the maximum similarity of direct descendants - $f_{structural_2}$. It is defined analogously, except that it considers the direct descendants instead of the ascendants.

Besides having an objective function, SA additionally requires the notion of neighborhood. To apply the notion of neighborhood to the context of OAs, an alignment is modelled as a state S and a function $n(S)$ is defined to generate successor states. All successor states S' that may be delivered by a particular neighborhood function $n(S)$ represent the neighborhood of S with respect to the neighborhood function n. The authors represent the current state S as $[(e_1, e_{j_1}), ..., (e_i, e_{j_i}), ..., (e_n, e_{j_n})]$ whereby (e_i, e_{j_i}) represents a correspondence. The neighborhood function $n(S)$ for a current state S, on the other hand, is defined through randomly swapping 5% of the entries in S and thus delivering S'.

Having an intrinsic objective function f, a way to represent an alignment as a state, and a function to generate new states within a neighborhood, SA can be employed to generate alignments.

A few implementation specific notes:
The initial state is initialized using a randomized greedy construction heuristic similar as in [17]. To ensure one-to-one alignments, [17] demands that each entity appears at most once in S. However, here this restriction was lifted during the re-implementation, leading to an approach that is capable of generating many-to-many alignments.

4.2.5 LogMap-ML

Chen et al. [16] proposed a machine learning extension that is used in the respective publication along with traditional OA approaches, e.g. LogMap or AML. The authors evaluated their approach on two food ontologies and on an alignment task of the OAEI and stated that, when compared to SOTA systems, the approach recalls many additional mappings and avoids several false positive.

The approach is divided into three stages: (a) utilization of an external OA system that is capable of generating high precision mappings, (b) training of a neural network on the output of the high-precision matcher, and (c) using the trained neural network to generate the final alignment. Figure 4.6 shows the entire procedure for LogMap as the underlying matcher. The three major stages are outlined in the upcoming paragraphs alongside with the steps from Figure 4.6.

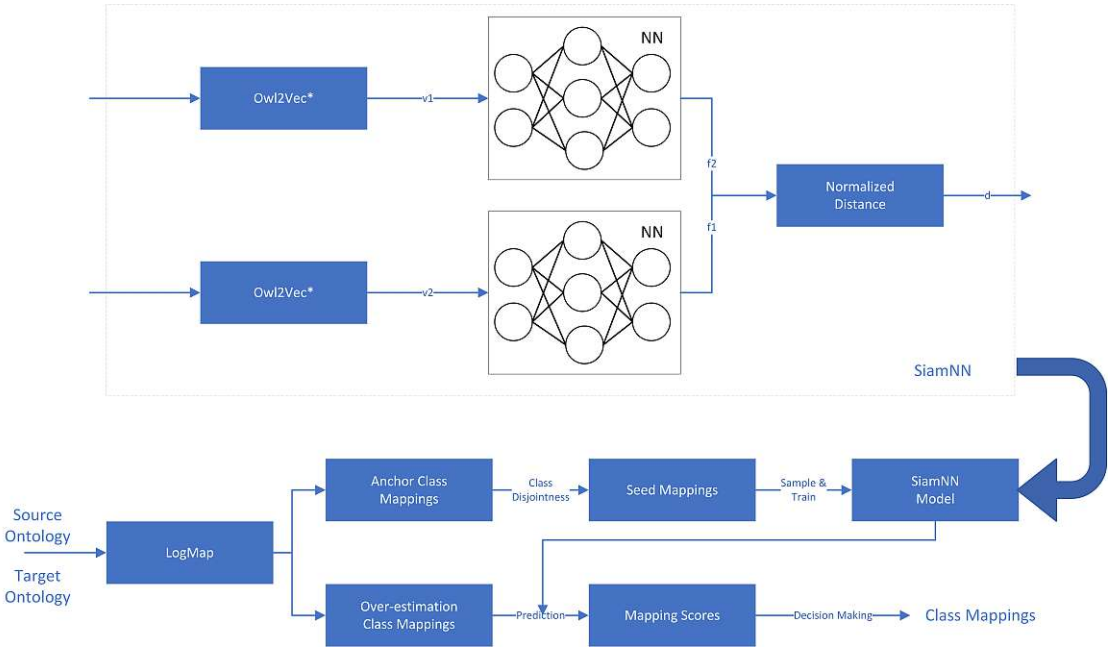

Figure 4.6: LogMap-ML overview [16]

The first stage comprises the steps *LogMap*, *Anchor Class Mappings*, *Seed Mappings*, and *Over-estimation Class Mappings* from Figure 4.6. As a starting point the anchor mappings from LogMap (high precision mappings) are used. These mappings represent

positives samples. Further, statistical methods are used to infer on class disjointness constraints (Note: during the experiments in the publication the authors additionally added disjointness constraints manually - this probably has boosted the performance). On the basis of these disjointness constraints, the current positives samples are filtered. The resulting positive samples form the set of seed mappings. Additionally to this sampling procedure, the LogMap over-estimations (high recall mappings) are generated that are used later during the prediction stage.

The second step (*SiamNN Model* step in Figure 4.6) is about the training of the neural network. In the experiments, the authors concluded that a Siamese Neural Network architecture shows decent performance. To train the model, it is initially necessary to obtain training data, i.e. in addition to the positive samples also negative samples are required. These negative samples are obtained through random sampling. Before the neural network can eventually be trained, it is necessary to transform the mappings into a representation that can be used by a neural network. The authors experimented with the language model Word2Vec [53], as well as with the ontology tailored embedding model Owl2Vec* [20]. The authors concluded that Owl2Vec* embeddings are more beneficial. Hereby, the authors discussed two embedding strategies. Firstly, directly embedding the classes using the embedding model. Secondly, traversing a random path in the class hierarchy until *owl:Thing* (top-level class in each OWL ontology) is reached and to embed the associated path. Both approaches have their advantages and disadvantages. Having positive and negative samples as well as an embedding model, it is finally possible to train the neural network.

During the third stage, the *Mapping Scores* step in Figure 4.6, the high recall mappings that have been generated during stage one (LogMap over-estimations) are used to predict the final output mappings. In theory one might also apply the trained model on the whole ontology, however, this would be computationally very intense and, therefore, in the most cases not feasible.

4.2.6 OntoConnect

Chakraborty et al. [39] presented a OA system that employs techniques from machine learning and is unsupervised in its nature. More precisely, the approach uses unsupervised learning techniques through the usage of a recursive neural network architecture. The approach was evaluated against the *Anatomy* test case of the OAEI. The evaluation yielded satisfactory results.

Figure 4.7 illustrates the workflow of the system. The approach is divided into a learning phase (left side of the figure) and a prediction phase (right side of the figure). Note how the initial three steps are the same in both phases, i.e. the source ontology and target ontology are preprocessed the same way. Therefore, the approach might also be summarized in three stages, i.e. preprocessing stage, training stage, and prediction stage. The remainder of this section, there are three paragraphs, describe the preprocessing steps, the training procedure, as well as the prediction approach.

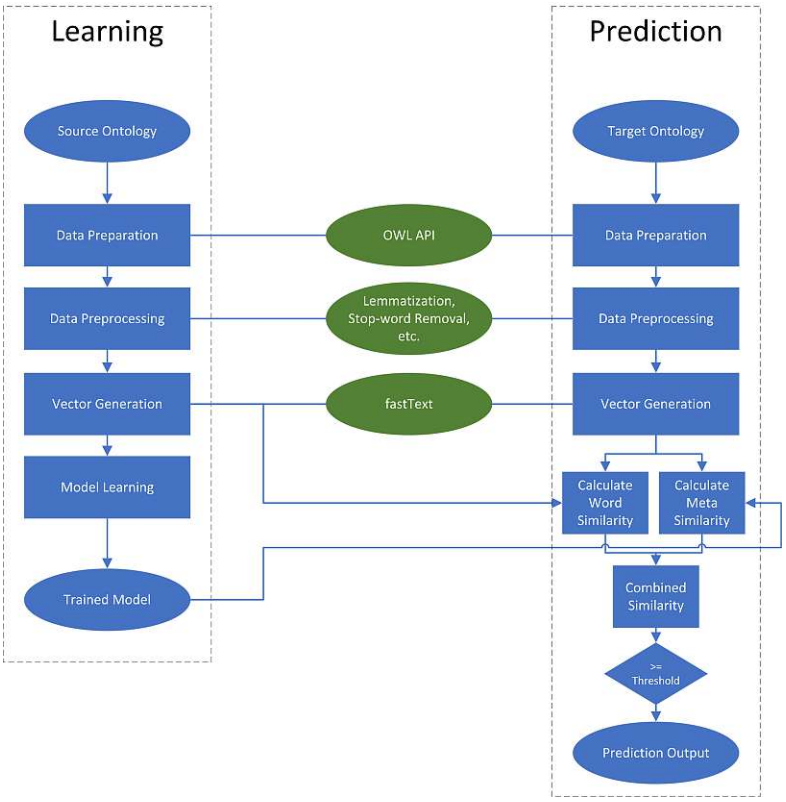

Figure 4.7: Overview of OntoConnect system [39]

As initial step, the source and target ontologies are parsed to extract meta information such as labels, IRIs, and various relationships (e.g. sub-class relations). Afterwards on both ontologies several preprocessing techniques are employed, e.g. stemming, stop word removal, etc. Next, a vector for each concept is generated through the usage of the pre-trained word embedding fastText [54].

In the second stage, the meta-information (encoded as fastText word embedding) for each of the concepts within the source ontology is fed into a recursive neural network architecture - a Long Short-Term Memory (LSTM) architecture. The major advantage of recursive architectures, such as LSTM, is that they are able to process an arbitrary number of input vectors. This is useful since a concept might have an arbitrary size of meta-information. The output of the LSTM architecture is the source class itself. Note that the intuition behind this kind of training is that it is very likely that a true positive target class has similar meta-information when compared to a source class. In these cases, the model will be able to predict the same or, at least, a similar vector when compared to the source class vector.

Having a trained LSTM model, it is time for the third stage, i.e. the prediction stage. The prediction stage starts by computing the cosine similarity between each concept vector of

the source ontology to each concept vector of the target ontology (step *Calculate Word Similarity* in Figure 4.7). Next, each concept vector of the target ontology is fed into the trained model. The output is another vector that approximates the corresponding concept vector of the source ontology. Afterwards, a second similarity score is obtained through computing the cosine similarity between each concept vector of the source ontology and this predicted vector. This step is illustrated as *Calculate Meta Similarity* in Figure 4.7. The final similarity score is calculated through combining both similarity scores using a weighted harmonic mean. Finally, the alignment is generated through rejecting mappings that are below a well-defined threshold.

4.2.7 BERTMap

When describing LogMap-ML [16] and OntoConnect [39], it became clear that many of the recently proposed machine learning based OA systems rely on non-contextualized word embeddings, such as Word2Vec or fastText. The advantage of such embeddings is that they are typically time efficient since no problem-specific fine-tuning is necessary - any pre-trained word embedding can, in principle, be applied to any domain. The problem, on the other hand, with non-contextual word embeddings, such as Word2Vec, is that they do not solve the ambiguity problem, i.e words have different meanings depending on the context. This is why these embeddings are not able to capture deviating word senses in different contexts [25]. In contrast to non-contextual word embeddings, contextual word embeddings provide different vectors for the same words depending on the context. As an example for contextual word embeddings one might mention Bidirectional Encoder Representation from Transformers (BERT) [55].

He et al. [38] explored the capabilities of BERT towards the OA problem. Doing so, the authors proposed BERTMap, i.e. an OA system that uses BERT word embeddings as its core alignment strategy. Figure 4.8 shows the 3-step approach of BERTMap: (a) fine-tuning stage, (b) prediction stage, and (c) extension and repair stage.

The aim of the first stage is to extract the corpora and to fine-tune a pre-trained BERT model. Figure 4.8 distinguishes between intra-ontology corpos, cross-ontology corpus, and complementary corpus. The intra-ontology corpus consists of all synonyms and non-synonyms that can be extracted from a single ontology. To this extent, synonyms are derived from labels of the same class, i.e. if a class has multiple *rdfs:label* annotations, then the resulting set of labels are considered as a synonym set. Non-synonyms, on the other hand, are further distinguished in soft non-synonyms, i.e. labels from two random classes, and hard non-synonyms, i.e. labels from logically disjoint classes. Further, if there is a sample of annotated mappings (i.e. a partial reference alignment), then one is able to construct a cross-ontology corpus and, thus, support a semi-supervised setting. If external ontologies are available, then the construction of a complementary corpus is feasible as well. The latter two, i.e. complementary corpus and cross-ontology corpus, are neglected in the further descriptions since the problem domain of this thesis does not support the construction of such corpora. Having extracted the corpora, i.e. having a set

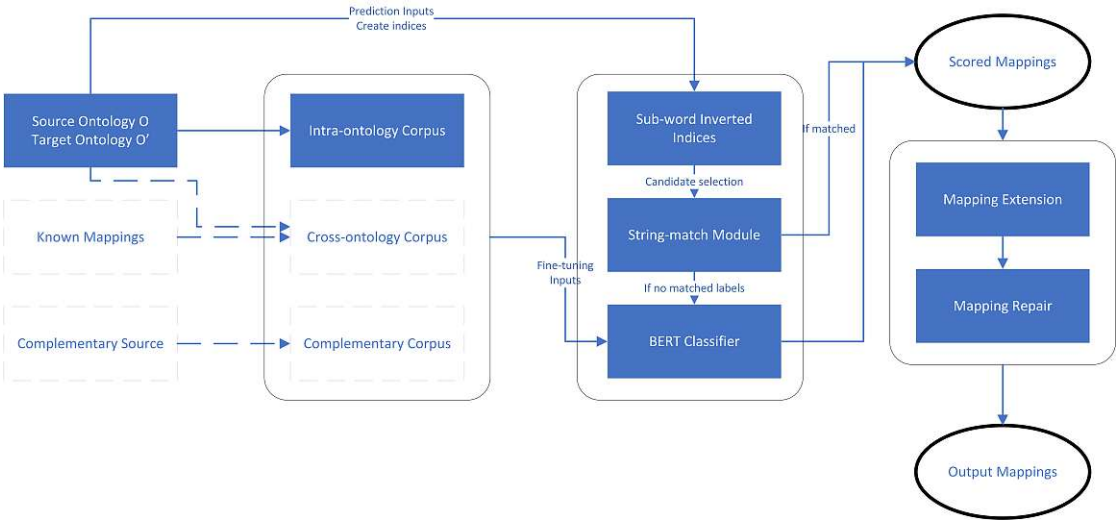

Figure 4.8: Overview of BERTMap system [38]

of synonyms and non-synonyms that are problem-specific, a pre-trained BERT model can be fine-tuned.

The second stage performs the prediction/generation of an initial alignment. As initial step, a sub-word inverted index is built (see Figure 4.8). Each entry of this index is a sub-word and its value is a list of classes that have at least one label containing this sub-word. This index is then employed to search for candidate target classes for each source class. The number of candidate classes is controlled via a parameter. These candidate classes are then ranked by the value of IDF and further fed into the string matching module (see Figure 4.8). The string matching module, in turn, returns two classes if they have one label in common. It is assumed that if two classes have a label in common, then the resulting classes are equivalent. In this way computation time is saved since the involved classes are not fed into the next step - the BERT classification (see Figure 4.8). All the remaining classes are fed into the fine-tune BERT model and are included as correspondences if the obtained similarity score is above a given threshold.

During the third stage, a similar approach is taken as in LogMap - an iterative extension and repair procedure. The extension step utilizes the principle of locality that states that if two classes match, then their respective parents and children are also likely to be related. Note the this extension procedure is only performed for mappings above a given threshold - for high confidence mappings. During the prediction and extension stages, mappings might have been introduced that lead to logical inconsistencies. These mappings are then removed during the mapping repair step. The mapping repair step is outsourced to LogMap.

4.2.8 LXLHMeta

Lu et al. [42] proposed a ontology meta matching technique that combines two broad categories of similarity measures. The evaluation on a test case of the OAEI showed the effectiveness of the approach.

This meta matching technique utilizes three similarity measures. Two of these three similarity measures are applied to the IDs and labels of each concept (typically referred to by *rdfs:label* annotations), while the last one is applied to the comments (typically referred to by *rdfs:comment* annotations).

The first similarity measure is the Wu & Palmer measure [52] - a WordNet based measure. The next similarity measure is the N-gram measure. And the last similarity measure is the cosine similarity. Wu & Palmer and N-gram are applied to the IDs and labels, while the cosine similarity is applied to the comments. The cosine similarity measure is a similarity measure for vectors. Therefore when computing the cosine similarity between two sentences s_1 and s_2, both sentences have to be converted into a vector representation. This is achieved through representing the string s_i as an array of its words, thus creating $D_i = (s_{i_1}, ..., s_{i_n})$, whereby s_{i_j} for $1 \leq j \leq n$ is a word. Based on $C = D_1 \cup D_2$ it is now possible to derive $V_i = (v_{i_1}, ..., v_{i_n})$ with v_{i_j} $1 \leq j \leq n$ being equal to 1 if $C_j \in D_i$ and 0 otherwise. In this way V_1 and V_2 can be constructed out of the strings s_1 and s_2. The cosine distance is then defined as:

$$Cos(V_1, V_2) = \frac{V_1 \cdot V_2}{||V_1|| \times ||V_2||}$$

Having defined the underlying similarity measures, the first step is then to calculate the similarity sets N and W. N and W represent sets of correspondences. The measures Wu & Palmer as well as N-gram are computed between each concept of the source ontology and each concept of the target ontology. Wu & Palmer as well as the N-gram similarity scores are calculated on the ID of the concepts, as well as on the labels. If the Wu & Palmer measure for two IDs is above a given threshold or the measure of the labels of the same entities is above that threshold, then the related correspondence is assigned to the set W. Analogously the set N is constructed, except with N-gram as its similarity measure.

The next step is to combine W and N, i.e. $U = W \cup N$. The authors now distinguish between four cases regarding these three sets. Due to the fact that none of the ontological concepts (in CPC, CCS, and CSO) have *rdfs:comment* annotations, the four cases boil down to one simple check:

- For $W \setminus N$, $N \setminus W$, and U there is only one entity matching pair in the three. By *entity matching pair* the authors refer to a correspondence. Here, it is assumed that this means: for each entity there is only one correspondence c in the three sets, i.e. the related entity matching pair is either in $W \setminus N$, $N \setminus W$, or U which would imply that $c \in N \cap W$.

- If true for a correspondence: take that correspondence and put it into the set S
- If false for a correspondence: for each correspondence in U that contains one of the involved entities, calculate the average N-gram measure between the IDs and the labels. Assign the resulting correspondence to the set S if the value is above an adapted threshold (higher than during N generation).

The authors further propose to vary the entity matching order as they observe that the set S changes if doing so. However, following the descriptions provided in the paper there is no evidence as why this should be the case since N and W are the same regardless of the order (the authors explicitly state that each concept is compared against every other concept). Consequently, the outcome of the above described check is the same regardless of the entity matching order, thus the set S remains stable as well.

4.2.9 LJLEvolutionary

Lv et al. [43] proposed an Evolutionary Algorithm (EA)-based OA system. In previous works EA-based solutions required a reference alignment in order to evaluate the quality of the genereted solutions. Here, the authors utilize an approximated *f1-score* metric as its objective function, thus leading to an unsupervised approach. The second major contribution regards the prevention of premature convergence through the employment of an adaptive selection pressure. Evaluation on an OAEI test case proved that the approach was competitive with other EA-based solutions.

The basic idea behind EAs comes from mechanism in the context of biological evolution, e.g. mutation, selection, mating. EAs typically involves (a) the generation of a set of solutions (population) and (b) iterative altering the set of solutions through a set of variation operators. The aim of (b) is to improve the fitness of the population, i.e. optimizing the solutions towards an objective function. The type of EA employed in LJLEvolutionary is a Genetic Algorithm (GA). In GA an individual/solution is represented through chromosomes that are typically implemented as an array of values. Using an array representation the three standard altering operators are easy to implement, namely selection, recombination (mating), and mutation. More information on GA can be found in [56].

As it is common for meta-matching techniques, several similarity measures are employed, namely two syntax-based similarity measures (N-gram and SMOA), one linguistic-based measure (Wu & Palmer), and one structure-based similarity measure (SimRank). According to the evaluation on the OAEI benchmark dataset, SimRank did not prove to be beneficial, therefore it was omitted during the re-implementation.

To measure the similarity of two entities/concepts the IDs, labels, and comments of the respective entities/concepts are considered. Thus, for each of the above outlined similarity measures three different similarity matrics originate. One of the major contributions of this work was the introduction of a novel framework for the aggregation of different

similarity matrices stemming from different textual contents. Previous studies typically aggregated the similarity matrices through MAX, MIN, or Averaging. Such integration mechanisms, however, suffer from various shortcomings, e.g. MIN puts too much emphasis on precision, MAX on recall, etc. Figure 4.9 illustrates the framework. For any pair of entities $< E_1, E_2 >$ it is checked if the similarity measure of the respective IDs is above the threshold V_1, if so: retain the value at that position, if not: apply a second filtering method. In the second filtering method, the similarity measures for label and comment are aggregated through the MAX operator. If the resulting value is larger than a second threshold V_2, that particular value is written at the corresponding position in the matrix, if smaller 0 is returned.

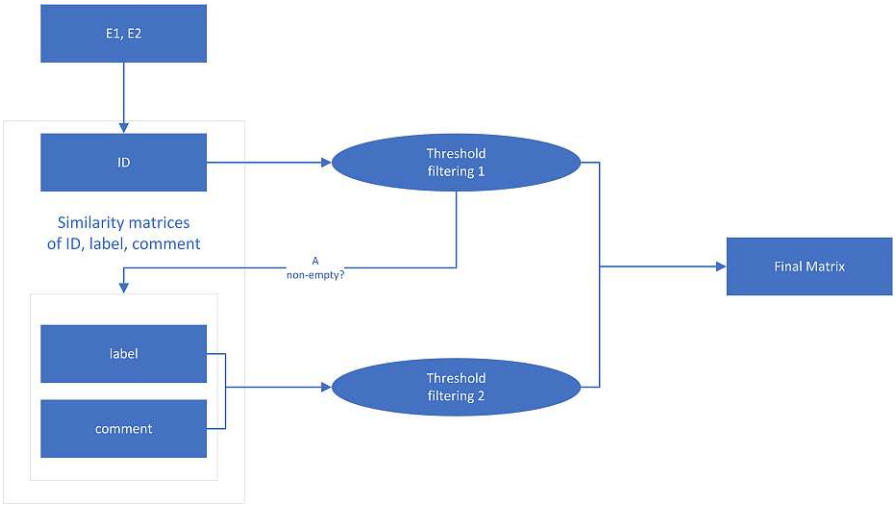

Figure 4.9: Framework to aggregate similarity matrices from different textual contents [43]

After having obtained a way how similarity matrices stemming from different textual contents can be aggregated, a way is required to merge similarity matrices originating from different similarity functions. For that purpose a simple weighted average is used:

$$M_{final} = \sum_{k=1}^{n} (w_k \; M_k)$$

whereby $0 \leq w_k \leq 1$ is the weight of the k'th similarity function with $\sum_{k=1}^{n} w_k = 1$ and M_k is the matrix originating from the k'th similarity function.

The fitness of each individual is then given by a function $f(M_{final}, O_1, O_2)$. f is defined in way such that it approximates the *f1-score* metric, i.e. $f(M_{final}, O_1, O_2) = f - measure_{pseudo}$. To compute $f - measure_{pseudo}$ it is necessary to approximate recall and precision. Precision and recall are approximated with the assumption of one-to-one mappings. *Recall* is approximated through:

$$recall_{pseudo} = \frac{num(M_{final})}{min(|O_1|, |O_2|)}$$

where $num(M_{final})$ represents the number of rows that contain a cell m_{ij} that is the maximum value in row i and column j. Next, *precision* is approximated through:

$$precision_{pseudo} = \frac{conf(M_{final})}{num(M_{final})}$$

where $conf(M_{final})$ is the sum over all m_{ij} that are the maximum value in their respective row and column. Thus, $precision_{pseudo}$ indicates the level of confidence. Having $recall_{pseudo}$ as well as $precision_{pseudo}$ $f - measure_{pseudo}$ can also be computed through the standard *f1-score* formula, i.e.:

$$f - measure_{pseudo} = \frac{2 \times recall_{pseudo} \times precision_{pseudo}}{recall_{pseudo} + precision_{pseudo}}$$

As indicated by the previous paragraph, V_1, V_2, w_1, ...w_n represent the search space that is explored to find high values of $f(M_{final}, O_1, O_2)$. Therefore, the chromosomes of an individual are defined through an array that encodes these values, i.e. $[V_1, V_2, w_1, ..., w_n]$.

A crucial concept in the field of GA is the notion of selection. To prevent early convergence, the generation crossing individuals are selected at random using probabilities that are derived from $f(M_{final}, O_1, O_2)^K$ whereby K is the so-called attenuation factor. The attenuation factor ensures to have more exploration in early iterations and in situations where the fitness of two subsequent generation deviates heavily. More precisely, the attenuation factor is defined as:

$$K = exp(-\frac{E(G_{new}) - E(G_{old})}{\frac{iter}{T}} + \theta)$$

whereby G_{new} represents the new population, G_{old} the old one, $E(X)$ is the total fitness of population X, *iter* represents the current iteration number, and θ and T are constants.

As crossover the very simple one-point crossover operator is employed. For example, assume the two individuals I_1 and I_2 with chromosomes $(V_{1_1}, V_{1_2}, w_{1_1}, w_{1_2})$ and $(V_{2_1}, V_{2_2}, w_{2_1}, w_{2_2})$ are selected for crossover, then the chromosome of the child comprises $(V_{1_1}, V_{1_2}, w_{2_1}, w_{2_2})$.

As mutation operator, the authors chose to flip a random bit of a chromosome with low probability. However, this operator is only applicable when chromosomes are encoded as bit-strings (in this approach the chromosomes are encoded through a vector of floating point numbers - $[V_1, V_2, w_1, ..., w_n]$). Therefore, the corresponding part of the paper seems to be incomplete and an assumption was necessary during re-implementation. In the re-implementation a chromosome is mutated with probability 0.01 (same as during the experiments of the respective paper). If the case of mutation, each value in the chromosome array is altered by a random value $-1 < r < 1$.

4.2.10 Notes on Third-Party Libraries

During the implementations of the previously described OA approaches, several external libraries have been used. Among the libraries we have: Pandas [57], Numpy [58], Gensim [59], Torch [60], Scikit-learn [61], Tensorflow [62], Owlready2 [63], MELT [64], and Alignment API 4.0 [65].

4.3 Summary

This chapter identified suitable ontology alignment approaches from the literature that are capable of aligning CPC-CCS and CPC-CSO. While Section 4.1 described the conducted literature review, the consecutive Section 4.2 detailed all the resulting OA systems in conjunction with the adaptations. We saw that the approaches are quite heterogeneous, i.e. there are machine learning centered solutions as well as traditional approaches. Due to this multifaceted set of OA systems, the first four requirements elicited in Chapter 3 are fulfilled.

CHAPTER 5

Experiments on Ground Truth

Chapter 4 described the conducted literature survey and detailed the resulting OA systems.

This chapter describes the setup for the conducted experiments, the related results, also provides reasons for the selected OA systems to align CPC-CCS and CPC-CSO. Thus, this chapter answers the second research question:

RQ2: What is the performance of the identified Ontology Alignment (OA) approaches on this problem domain?

The chapter starts with Section 5.1 that describes everything related to the reference alignments, i.e. the alignments against which the output of the OA systems is compared. Afterwards, Section 5.2 describes for each OA system its respective configurations, i.e. which parameters have been used on which OA system. The evaluation criteria are then defined in in Section 5.3. To put the measured runtimes of the individual OA systems in the correct context, Section 5.4 provides the core properties of the laptop on which the experiments have been conducted. Then, the subsequent sections describe the obtained results. Section 5.5 provides the outcomes of the CPC-CCS experiments. Analogously, Section 5.6 is about CPC-CCS and Section 5.7 is about the third-party dataset *Anatomy*. The penultimate section argues which approach has been eventually used to align CPC and CCS as well as CPC and CSO. The chapter concludes with a summary.

5.1 Ground Truths / Reference Alignments

As there is no problem specific gold standard to evaluate the OA systems, it is necessary to manually create ground truths.

The first step in generating the ground truths was to select a subset of CPC, CCS, and CSO, respectively. These subsets (sub-ontologies) are referred to as CPC_{subset}, CCS_{subset},

and CSO_{subset}. Afterwards, rules were defined that allow determining correspondences between the different concepts. The resulting correspondences form the reference alignments $GT_{CPC-CCS}$ and $GT_{CPC-CSO}$, for CPC_{subset}-CCS_{subset} and CPC_{subset}-CSO_{subset}, respectively. A tsv-file containing both reference alignments in conjunction with justifications can be found in the repository of this thesis - see Section 1.4.

Limitation. Although best effort was invested in creating ground truths that are as objective as possible through, for example, the definition of clear rules, it would be unrealistic to assume that another person would end with the same reference alignments. This is a very common limitation when evaluating OA systems as the experiments in [66] show. Tordai et al. [66] conducted three experiments in which experts evaluated alignments and talked through their decisions using the think aloud method. As result the researchers recognized that the agreement between the experts was very low, i.e. two different experts might create and evaluate an alignment significantly differently.

5.1.1 Sub-Ontologies

Te generate the sub-ontologies CPC_{subset}, CCS_{subset}, and CSO_{subset}, the knowledge branches of CPC, CCS, and CSO, respectively, have been restricted to the domain Software Engineering. All branches that do not cover Software Engineering were eliminated, thus creating three subsets.

CPC_{subset}. Kotti et al. [7] identified 117 CPC classes that are related to Software Engineering. These classes were used in this work to define the sub-ontology of CPC, namely CPC_{subset}. Note that not all of the identified software engineering classes belong to the same hierarchy. For example, *G06F8/451 Code distribution* references *G06F9/5083 load rebalancing* that, in turn, references *G06F9/5088 involving task migration*. As a consequence, in the generated sub-ontology *Code distribution* is a sibling of *load rebalancing* and an uncle of *involving task migration*, while in the original CPC ontology *Code distribution* is in a completely different branch than *load rebalancing*. This imposes a limitation on the ground truth, however, as there are only a few cases where the original hierarchy is disturbed, it can be assumed that this limitation is not severe.

CCS_{subset}. To retrieve a sub-ontology for CCS simply everything below the concept *Software and its engineering* was used.

CSO_{subset}. For CSO a similar simple approach is chosen since everything below *computer programming*, *software*, and *software engineering* forms the sub-ontology for CSO.

5.1.2 Ground Truth Construction Rules

In order to obtain ground truth that is as objective as possible, the following rules have been considered during the generating of the ground truth:

- *R1:* If the descriptions of two concepts match (in case a description for a concept exists in the first place - note each CPC concept has a description, but only a few CSO concepts and none of the CCS concepts possess a description) - assign similarity score of 1.
  - Example: The description of the CPC concept *G06F8/75 Structural analysis for program understanding* matches with the description of the CSO concept *static analysis*.
- *R2:* If the name is the same/is a synonym and also both ascendants are similar - assign similarity score of 1.
  - Example: *G06F8/36 Software reuse* of CPC and *software re-use* of CSO.
- *R3:* If the names deviate and one cannot determine if the concepts are different, then consider them as a match if ascendants & descendants are similar. In such a case assign a value of 1.
  - Example: *G06F8/70 Software maintenance or management* of CPC and *Software post-development issues* of CCS.
- *R4:* If *R2* can be applied after transforming to noun/verb/adjective, then reduce the similarity score by 0.1.
  - Example: *G06F8/313 Logic programming* of CPC and *logic programs* of CSO.
- *R5:* If ascendants are similar and one concept is 'slightly' broader than the other, then reduce the similarity score by 0.2.
  - Example: *G06F8/24 object-oriented* of CPC and *Object oriented development* of CCS.
- *R6:* If ascendants are similar and both concepts deviate, but the intersection is still considered 'large enough', then reduce the similarity score by 0.3.
  - Example: *G06F8/33 Intelligent editors* of CPC and *Integrated and visual development environments* of CCS.
- Else: reject.

While *R1-R4* can be determined relatively objectively, the rules *R5-R6* are subject to the eye of the beholder. To address this limitation, two variations of the ground truth were created. One ground truth that contains all the resulting correspondences and one ground truth that comprises only correspondences with similarity scores above 0.9. Thus, we arrive at 4 ground truths in total, i.e. $GT_{CPC-CCS}$, $GT_{CPC-CSO}$, $GT_{CPC-CCS;\geq 0.9}$, and $GT_{CPC-CSO;\geq 0.9}$.

Despite best effort in developing the above outlined criteria, there are still several cases in which another person would have chosen differently. For example, *Aspect-oriented programming techniques* of CPC and *aspect-oriented software* of CSO was not included in the respective ground truth, however, another person might have included that correspondence.

5.1.3 Anatomy

To also gauge performance metrics for each OA system on an independent use case, the test case *Anatomy* from the OAEI has been considered as well. The advantage of using another use case is to see how well each OA system scales as the number of classes increases and to observe the generalization capabilities for each OA system.

The *Anatomy* test case comprises two fragments of biomedical ontologies [12]. While one fragment describes the human anatomy, the other fragment describes the mouse anatomy [12]. The former consists of 3304 classes, and the latter consists of 2744 classes [12]. The manually curated reference alignment between both ontologies varies between the years. The version that has been used during the experiments comprises 1516 correspondences.

An important property that may impose performance restrictions onto several approaches, especially BERTMap, is that each class has only one *rdfs:label* annotation. On such ontologies the unsupervised BERTMap setting won't be able to work properly since the intra-ontology corpus (see Section 4.2) is essentially empty, meaning that it is not possible to derive synonyms (with the exception of identity synonyms). As a consequence, the pre-trained BERT embedding cannot be fine-tuned effectively.

5.2 Parameter Space

This section outlines the configurations that have been used during the experiments. In a nutshell, three different LogMap settings, one AML setting, two SANOM settings, one OntoConnect setting, four LogMap-ML settings, two BERTMap settings, one LXLHMeta setting, and one LJLEvolutionary setting were compared against each other.

More precisely, the various settings comprise:

- LogMap:
  - LogMap with default parameters - see the corresponding *paramters.txt* file in LogMap's GitHub repository [1].
  - LogMapLt, i.e. the lightweight variant of LogMap, which skips all reasoning, repair, and semantic indexation steps [67].
  - LogMap$_{Exp}$ (Exp = experimental), i.e. LogMap with modified default parameters. The aim was to make LogMap more 'aggressive', i.e. increasing recall through stemming, as well as reducing the weights *good_isub_anchors*, *good_isub_candidates*, and *good_confidence*, respectively.
- AML was used in its default variant.
- SANOM:
  - SANOM without WordNet.
  - SANOM with WordNet.

[1] https://github.com/ernestojimenezruiz/logmap-matcher/blob/master/parameters.txt

- OntoConnect was used as it is shipped through its respective Docker image [2].
- LogMap-ML:
  - Owl2Vec* as embedding model (with Word2Vec as pre-trained model) + Label embedding.
  - Owl2Vec* as embedding model (with Word2Vec as pre-trained model) + Path embedding.
  - Word2Vec (same pre-trained model as in [16]) + Label embedding.
  - Word2Vec (same pre-trained model as in [16]) + Path embedding.
- BERTMap:
  - 25 as candidate number.
  - 200 as candidate number.
- LXLHMeta was used with the configurations described in its respective publication [42] (exceptions were discussed in Section 4.2).
- LJLEvolutionary also only used in one configuration (see its publication [43] and Section 4.2).

5.3 Alignment Evaluation Criteria

To evaluate the performance of the individual OA systems, *precision*, *recall*, and *f1-score* were measured. In addition to these performance metrics, also the runtime (in seconds) was measured. However, it is important to note that the reported runtimes are by no means representative and, thus, shall not be treated as a benchmark. The runtime for each approach simply stems from one independent run that might have been distributed by various background processes on the execution machine. Therefore, the runtimes simply provide a rough indication about the time efficiency and scalability of the related approaches. The following paragraphs provide the definition for each performance metric.

Precision represents the percentage of true positives to the overall predictions. Thus, in the context of OA *precision* is defined as:

$$precision = \frac{|R \cap A|}{|A|} \tag{5.1}$$

where R is the reference alignment and A is the predicted alignment.

Recall, on the other hand, indicates the percentage of true positives to the overall number of positives in a dataset. In the context of OA it is:

$$recall = \frac{|R \cap A|}{|R|} \tag{5.2}$$

[2] https://hub.docker.com/r/jchakra1/ontosim_imgbatch

The third performance measure, namely the *f1-score*, is a combination of *precision* and *recall*, i.e. it is the harmonic mean between both measures:

$$f1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.3}$$

5.4 Hardware

All the experiments were carried out on a laptop with 16 GB of RAM and 4 CPU cores with a base clock of 2.3 GHz. These specs are comparatively low, especially when compared to the machines employed in the OAEI (see for instance [12]). Further, as visible when inspecting the GitHub repository of this thesis, all the approaches have been executed as Docker containers, which, in turn, might have led to an additional performance overhead.

5.5 Results on CPC-CCS

As already explained in Section 5.1, there exist two versions of the ground truth for CPC-CCS, namely $GT_{CPC-CCS}$ and $GT_{CPC-CCS;\geq 0.9}$ whereby $GT_{CPC-CCS;\geq 0.9} \subseteq GT_{CPC-CCS}$.

Performance on Full Ground Truth. Table 5.1 shows the performance of all the OA systems on $GT_{CPC-CCS}$. All the figures are rounded to two decimals. The naming convention of the approaches follows the pattern name_of_approach$_{configuration}$. The results are sorted by *f1-score*. The last three columns indicate the number of true positives (#TP), number of false positives (#FP), and the size of the respective ground truth (#GT).

LogMap$_{Exp}$ is the approach with highest *f1-score* and a moderate *precision* metric of 0.68. SANOM achieves the same performance metrics with and without WordNet. Thus, WordNet is probably not beneficial for this task. Despite SANOM having the second best *f1-score*, its *precision* metric ranks with only 0.34 very low. Next, LogMap achieved a reasonable *f1-score* and promising *precision* metric of 0.75. The remaining approaches were all either too aggressive (i.e. predicting too many correspondences at the cost of *precision* such as OntoConnect) or were too cautious (i.e. predicting too few mappings at the cost of *recall* such as LXLHMeta).

Further, as it was expected, BERTMap performed very poorly on this dataset since there is no training data available (remember: the fine-tuning procedure in BERTMap in an unsupervised setting requires that the concepts have multiple labels, however, the CCS ontology has only one label per concept). Also interesting to see that LogMap-ML became more effective when OWL2Vec* embeddings were used instead of Word2Vec embeddings. This is congruent with the observations in [16]. Regarding the application of WordNet, it becomes clear that applying WordNet is not beneficial within this problem

domain. In general, it is safe to assume that the machine learning centered approaches suffer from the small sizes of the sub-ontologies. For example, LogMap-ML relies on distant supervision for its training process, however, the underlying data stems from a high precision matcher (i.e., the LogMap anchor mappings) that can, on this use case, only conclude on a few mappings and, therefore, the trained neural network performed relatively poorly when generating the alignment. The performance of LJLEvolutionary is disappointing since it is the only matcher that performed worse than the Baseline.

Approach	Precision	Recall	F1-Score	#TP	#FP	#GT
LogMap_{Exp}	0.68	0.36	0.47	15	7	42
SANOM	0.34	0.48	0.40	20	39	42
$\text{SANOM}_{WordNet}$	0.34	0.48	0.40	20	39	42
LogMap	0.75	0.21	0.33	9	3	42
$\text{LogMap-ML}_{Owl2Vec*\_path}$	0.35	0.21	0.26	9	17	42
AML	0.75	0.14	0.24	6	2	42
OntoConnect	0.16	0.45	0.24	19	98	42
$\text{LogMap-ML}_{Owl2Vec*\_label}$	0.50	0.12	0.19	5	5	42
LXLHMeta	1.00	0.10	0.17	4	0	42
$\text{LogMap-ML}_{Word2Vec\_label}$	0.44	0.10	0.16	4	5	42
$\text{LogMap-ML}_{Word2Vec\_path}$	0.31	0.10	0.15	4	9	42
LogMapLt	1.00	0.07	0.13	3	0	42
BERTMap-US_{200}	1.00	0.05	0.09	2	0	42
BERTMap-US_{25}	1.00	0.05	0.09	2	0	42
Baseline	1.00	0.05	0.09	2	0	42
LJLEvolutionary	0.67	0.05	0.09	2	1	42

Table 5.1: Results on $GT_{CPC-CCS}$ of all models sorted by F1-Score

Performance on High Confidence Ground Truth. The performance on $GT_{CPC-CCS;\geq 0.9}$, described in Table 5.2, is very similar to the performance on $GT_{CPC-CCS}$. The only additional insight, that is revealed in Table 5.2, is that LogMap ranks higher than on the full ground truth. This shows that LogMap is a very effective high-precision matcher.

Alignment Analysis. Figure 5.1 illustrates which systems are similar to each other in terms of prediction output regarding $GT_{CPC-CCS}$. The graphic is essentially a heatmap. On the y-axis of the heatmap we see one entry per model/configuration. The x-axis, on the other hand, has one entry for each $c \in C$ whereby $C = P \cup GT_{CPC-CCS}$ with P being the union over all correspondences that at least one model has predicted. Everything at the left side of the dotted vertical black line belongs to the ground truth. This implies that red values at the left side represent false negatives, while the green ones represent true positives. The right side of the dotted line contains all the wrong correspondences that at least one model/configuration has generated. Red values at that

Approach	Precision	Recall	F1-Score	#TP	#FP	#GT
LogMap$_{Exp}$	0.55	0.48	0.51	12	10	25
LogMap	0.67	0.32	0.43	8	4	25
SANOM	0.24	0.56	0.33	14	45	25
SANOM$_{WordNet}$	0.24	0.56	0.33	14	45	25
LogMap-ML$_{Owl2Vec*_path}$	0.31	0.32	0.31	8	18	25
LogMap-ML$_{Owl2Vec*_label}$	0.50	0.20	0.29	5	5	25
LXLHMeta	1.00	0.16	0.28	4	0	25
AML	0.50	0.16	0.24	4	4	25
LogMap-ML$_{Word2Vec_label}$	0.44	0.16	0.24	4	5	25
LogMapLt	1.00	0.12	0.21	3	0	25
LogMap-ML$_{Word2Vec_path}$	0.31	0.16	0.21	4	9	25
OntoConnect	0.12	0.56	0.20	14	103	25
BERTMap-US$_{200}$	1.00	0.08	0.15	2	0	25
BERTMap-US$_{25}$	1.00	0.08	0.15	2	0	25
Baseline	1.00	0.08	0.15	2	0	25
LJLEvolutionary	0.67	0.08	0.14	2	1	25

Table 5.2: Results on $GT_{CPC-CCS;\geq 0.9}$ of all models sorted by F1-Score

side indicates false positives, while the green ones indicate true negatives. We see that the systems SANOM and OntoConnect were way too aggressive, i.e. they predicted too many correspondences and have, as a consequence, too many false positives. Further, we see that there exist multiple correspondences within the ground truth that no matcher was able to predict. Such correspondences often rely on on structural understanding of the ontologies, such as *compiler construction; parser generation* (CPC) and *Translator writing systems and compiler generators* (CCS). Vice versa, we also see several correspondences within the ground truth that each matcher got correct. Such correspondences often have identical labels in both concepts as well as congruent knowledge branches, such as *incremental compilation* (CPC) and *Incremental compilers* (CCS).

Runtimes. Table 5.3 shows that there exist large discrepancies between the individual approaches in terms of their runtimes. Note that the Owl2Vec* embedding generation is listed here separately as it has to be generated only once per alignment task and can then be re-used by subsequent LogMap-ML runs. Further note: BERTMap-US$_{200}$ and BERTMap-US$_{25}$ are summarized here as BERTMap-US that includes both runtimes. The major insight from the runtimes is that it is very probably only feasible to execute either LogMapLt, SANOM, LogMap$_{Exp}$, AML, LogMap, LXLHMeta, or LJLEvolutionary to generate a full alignment between CPC and CCS. This, however, is not a big limitation as the performance metrics of the other approaches were not as good (see Tables 5.1 and 5.2).

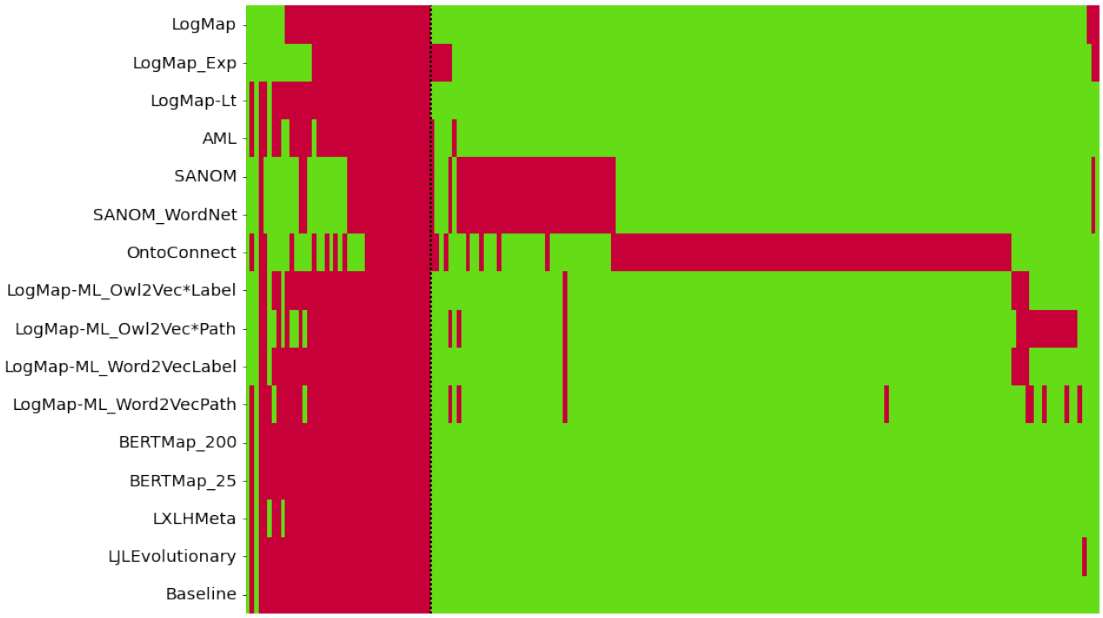

Figure 5.1: Heatmap for each correspondence-model tuple (CPC-CCS); green indicates correctly predicted/rejected; red indicates the opposite

Approach	**Sec**
Baseline	2
LogMapLt	3
SANOM	3
LogMap_{Exp}	5
AML	5
LogMap	6
LXLHMeta	10
LJLEvoluationary	14
$\text{LogMap-ML}_{Word2Vec\_label}$	44
$\text{SANOM}_{WordNet}$	50
$\text{LogMap-ML}_{Word2Vec\_path}$	62
$\text{LogMap-ML}_{Owl2Vec*\_label}$	77
$\text{LogMap-ML}_{Owl2Vec*\_path}$	94
OntoConnect	203
Owl2Vec*	264
BERTMap-US	2967

Table 5.3: Runtime in seconds to align CPC and CCS (subontologies)

5.6 Results on CPC-CSO

Equivalent to the previous section, also this section compares the generated alignments against the two ground truths - $GT_{CPC-CSO}$ and $GT_{CPC-CSO;\geq 0.9}$ whereby $GT_{CPC-CSO;\geq 0.9} \subseteq GT_{CPC-CSO}$.

Performance on Full Ground Truth. Table 5.4 shows the performance of all the OA systems on $GT_{CPC-CSO}$. We see that AML is the best performing approach for this alignment task with a *precision* score of 0.88 and an *f1-score* of 0.33. Besides AML, LogMap$_{Exp}$ is the only approach that performed better than the Baseline in terms of *f1-score*. The relatively good performance of the baseline is caused by many correspondences in $GT_{CPC-CSO}$ where the two involved concepts have identical labels. BERTMap performed much better than during the CPC-CCS ground truth alignment task. This was expected since many CSO concepts have multiple *rdfs:label* annotations and, therefore, a meaningful corpus and, consequently, fruitful fine-tuning of BERT is enabled. Interestingly, as it was also the case on $GT_{CPC-CCS}$ and $GT_{CPC-CCS;\geq 0.9}$, the candidate number of BERTMap did not impact the performance at all.

Approach	Precision	Recall	F1-Score	#TP	#FP	#GT
AML	0.88	0.20	0.33	21	3	103
LogMap$_{Exp}$	0.70	0.18	0.29	19	8	103
LogMapLt	0.94	0.16	0.27	16	1	103
BERTMap-US$_{200}$	0.94	0.16	0.27	16	1	103
BERTMap-US$_{25}$	0.94	0.16	0.27	16	1	103
LXLHMeta	0.94	0.16	0.27	16	1	103
Baseline	0.94	0.16	0.27	16	1	103
SANOM	0.33	0.22	0.27	23	47	103
SANOM$_{WordNet}$	0.33	0.22	0.27	23	47	103
OntoConnect	0.18	0.20	0.19	21	96	103
LogMap	0.90	0.09	0.16	9	1	103
LogMap-ML$_{Owl2Vec*_label}$	0.89	0.08	0.14	8	1	103
LogMap-ML$_{Word2Vec_label}$	0.80	0.08	0.14	8	2	103
LogMap-ML$_{Owl2Vec*_path}$	0.16	0.12	0.13	12	64	103
LJLEvolutionary	0.70	0.07	0.12	7	3	103
LogMap-ML$_{Word2Vec_path}$	0.22	0.04	0.07	4	14	103

Table 5.4: Results on $GT_{CPC-CSO}$ of all models sorted by F1-Score

Performance on High Confidence Ground Truth. The performance on $GT_{CPC-CSO;\geq 0.9}$ is very similar - see Table 5.5. The only difference is that LogMap$_{Exp}$ performs now below the Baseline. Thus, AML is the only approach that outperforms the Baseline.

Approach	Precision	Recall	F1-Score	#TP	#FP	#GT
AML	0.83	0.34	0.48	20	4	59
LogMapLt	0.94	0.27	0.42	16	1	59
BERTMap-US$_{200}$	0.94	0.27	0.42	16	1	59
BERTMap-US$_{25}$	0.94	0.27	0.42	16	1	59
LXLHMeta	0.94	0.27	0.42	16	1	59
Baseline	0.94	0.27	0.42	16	1	59
LogMap$_{Exp}$	0.56	0.25	0.35	15	12	59
LogMap	0.90	0.15	0.26	9	1	59
SANOM	0.23	0.27	0.25	16	54	59
SANOM$_{WordNet}$	0.23	0.27	0.25	16	54	59
LogMap-ML$_{Owl2Vec*_label}$	0.89	0.14	0.24	8	1	59
LogMap-ML$_{Word2Vec_label}$	0.80	0.14	0.23	8	2	59
LJLEvolutionary	0.70	0.12	0.20	7	3	59
OntoConnect	0.13	0.25	0.17	15	102	59
LogMap-ML$_{Owl2Vec*_path}$	0.14	0.19	0.16	11	65	59
LogMap-ML$_{Word2Vec_path}$	0.22	0.07	0.1	4	14	59

Table 5.5: Results on $GT_{CPC-CSO;\geq 0.9}$ of all models sorted by F1-Score

Alignment Analysis. Analogously to the previous section, Figure 5.2 illustrates which systems are similar to each other in terms of prediction output. The insights are closely related to the insights obtained when analyzing the output of the systems on CPC_{subset}-CCS_{subset} - see Section 5.5. SANOM, OntoConnect, and LogMap-ML$_{Owl2Vec*_path}$ were again too aggressive. Further, the approaches BERTMap-US$_{200}$, BERTMap-US$_{25}$, LXLHMeta, and Baseline predicted this time the exact same output.

Runtimes. Regarding the runtime, we see again large discrepancies between the approaches. As before, the only feasible approaches to generate the final alignment between CPC and CSO are LogMapLt, Baseline, LogMap, LogMap$_{Exp}$, AML, SANOM, LXLHMeta, and LJLEvolutionary. Especially, BERTMap was too slow (~6.5 hours).

5.7 Results on Anatomy

This section outlines the performance of the implemented OA systems on the third-party dataset - the *Anatomy* test case of the OAEI. The aim was to obtain a third, problem-independent, performance indication for all the OA systems as well as to test how they perform on a larger alignment task.

Performance Figures. Table 5.7 lists the performance of all systems. Note that this time there are *nan* values for the systems LogMap-ML$_{Owl2Vec*_path}$, LogMap-ML$_{Word2Vec_path}$, BERTMap-US$_{200}$, and BERTMap-US$_{25}$. Executing

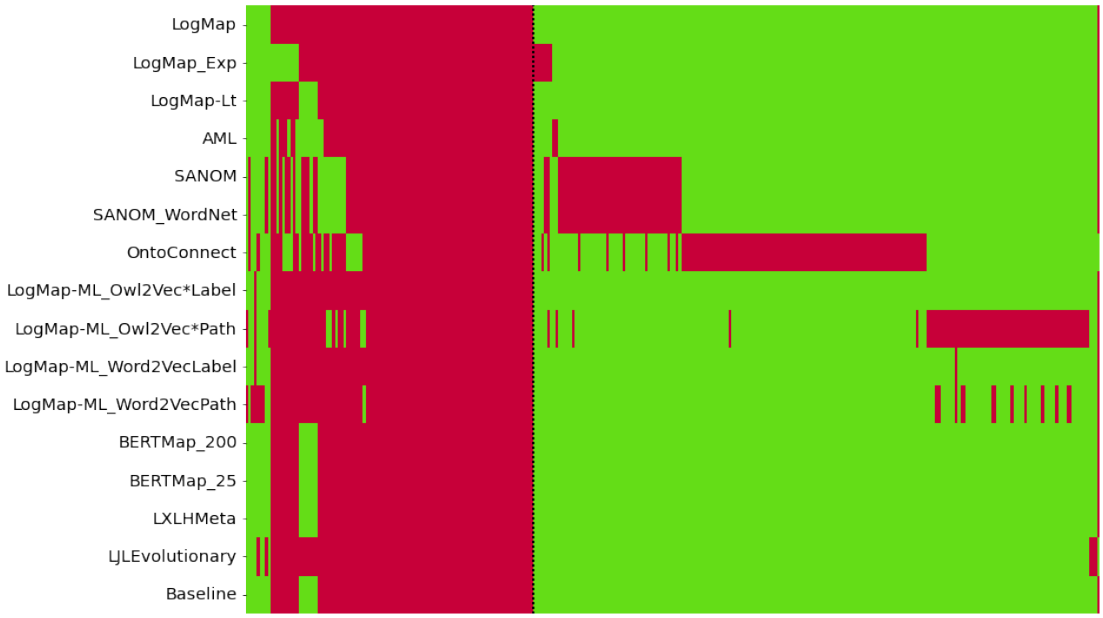

Figure 5.2: Heatmap for each correspondence-model tuple (CPC-CSO); green indicates correctly predicted/rejected; red indicates the opposite

Approach	**Sec**
Baseline	2
LogMapLt	3
LogMap	8
LogMap_{Exp}	8
AML	14
SANOM	17
LXLHMeta	20
LJLEvluationary	24
$\text{LogMap-ML}_{Word2Vec\_label}$	48
$\text{LogMap-ML}_{Owl2Vec*\_label}$	83
$\text{LogMap-ML}_{Word2Vec\_path}$	133
$\text{SANOM}_{WordNet}$	135
$\text{LogMap-ML}_{Owl2Vec*\_path}$	188
OntoConnect	207
Owl2Vec*	561
BERTMap-US	23191

Table 5.6: Runtime in seconds to align CPC and CSO (subontologies)

$\text{LogMap-ML}_{Owl2Vec*\_path}$ and $\text{LogMap-ML}_{Word2Vec\_path}$ led to memory issues as, apparently, the hardware used to align the ontologies was not sufficient. BERTMap-US_{200} and

BERTMap-US$_{25}$, on the other hand, were killed after ~5 hours since the approaches were only half way through.

It is worth noting that the performances of AML, LogMap, and LogMapLt that have been achieved during the OAEI conference of 2021 [12] deviate from those reported here. Considering the fact that those three approaches have not been altered, the differences might originate from slight differences in implementations when compared to the OAEI submissions and the implementations of the publicly available versions. Another reason for worse performance might be in the hardware limitations.

Further, also the approaches SANOM, SANOM$_{WordNet}$, and OntoConnect have deviating performances when compared to the numbers reported in their publications - see [17] [39], respectively. The implementation of SANOM, employed to generate the results reported here, indeed deviates from the description in the related publication [17], therefore it is no surprise that the performance numbers are different. OntoConnect, on the contrary, has not been altered at all. In fact, the provided OntoConnect Docker image [3] has been used. It is therefore unclear why the performance deviates so heavily. Note, in the respective publication [39] the authors reported *f-scores* around 80% depending on the configuration.

However, overall a very similar picture is delivered as in the previous sections, i.e. LogMap and AML show very promising results, while the rest of the approaches are rather limited in their performance.

Runtimes. The runtimes in table 5.8 reveal which approaches are scalable and, therefore, candidates to generate the final alignments. While LogMap and AML were still quite fast, the runtimes of other approaches, such as LXLHMeta or OntoConnect, increased significantly, and yet others, such as BERTMap or LJLEvolutionaray, showed very poor scalability.

5.8 Approach Selection

The previous sections listed the performances and runtimes of the various OA systems.

Based on the performance metrics and runtimes outlined in Section 5.5, it becomes clear that either LogMap$_{Exp}$ or LogMap are the most suited to align CPC with CCS. Note that while SANOM also achieves a good *f1-score*, it predicts far too many false positives and, thus, has a relatively low *precision* value. When comparing LogMap$_{Exp}$ and LogMap, it becomes evident that LogMap is better suited to align CPC with CCS since LogMap has higher *precision*. Additionally, the performance on Anatomy in Section 5.7 shows that LogMap is scalable, performs very well in other problem domains, and therefore is well suited to generate the full alignment between CPC and CCS.

[3] https://hub.docker.com/r/jchakra1/ontosim_imgbatch

Approach	Precision	Recall	F1-Score	#TP	#FP	#GT
AML	0.96	0.88	0.92	1336	60	1516
LogMap	0.91	0.85	0.88	1286	126	1516
LogMap$_{Exp}$	0.81	0.86	0.83	1301	314	1516
LogMapLt	0.99	0.64	0.78	971	12	1516
LogMap-ML$_{Word2Vec_label}$	0.65	0.68	0.67	1038	547	1516
LogMap-ML$_{Owl2Vec*_label}$	0.63	0.69	0.66	1041	603	1516
SANOM	0.64	0.59	0.62	898	500	1516
SANOM$_{WordNet}$	0.64	0.59	0.62	898	500	1516
OntoConnect	0.46	0.83	0.59	1256	1491	1516
LXLHMeta	0.99	0.15	0.26	225	2	1516
Baseline	1.00	0.13	0.23	200	1	1516
LJLEvolutionary	0.33	0.14	0.19	207	422	1516
LogMap-ML$_{Owl2Vec*_path}$	nan	nan	nan	nan	nan	nan
LogMap-ML$_{Word2Vec_path}$	nan	nan	nan	nan	nan	nan
BERTMap-US$_{200}$	nan	nan	nan	nan	nan	nan
BERTMap-US$_{25}$	nan	nan	nan	nan	nan	nan

Table 5.7: Results on Anatomy of all models sorted by F1-Score

Approach	Sec
LogMapLt	6
Baseline	9
LogMap	21
AML	22
LogMap$_{Exp}$	26
LXLHMeta	206
OntoConnect	255
LogMap-ML$_{Word2Vec_label}$	661
LogMap-ML$_{Owl2Vec*_label}$	865
SANOM	978
SANOM$_{WordNet}$	1112
LJLEvolutionary	1272
BERTMap-US	$>$ 17509

Table 5.8: Runtime in seconds on *Anatomy*

Considering the performances and runtimes on the CPC-CSO ground truth, that are described in Section 5.6, it is immediately obvious that AML is the best suited approach. AML is very efficient in terms of runtime, performs best in terms of *f1-score* (while having at the same time a high *precision* value), and is the best performing system on the *Anatomy* test case. Following these considerations, AML was used to generate the

full alignment between CPC and CSO.

5.9 Summary

This section has shown the performances and runtimes of various OA systems when aligning CPC_{subset} with CCS_{subset}, CPC_{subset} with CSO_{subset} and the mouse ontology with the human ontology from the *Anatomy* test case. Although still limited, it became clear that the SOTA matchers LogMap and AML are best suited to align CPC-CCS and CPC-CSO, respectively. Compared to the other systems, LogMap and AML showed also superior runtimes. Especially the machine learning centered approaches were either too slow or/and suffered from the relatively small ground truth and the absence of supervised data.

The next chapter describes the alignment between CPC and CCS that has been retrieved through the application of LogMap, as well as the alignment between CPC and CSO that has been generated using AML.

CHAPTER 6

Analysis of Final Alignments

The previous chapter evaluated all the OA systems regarding their applicability/performance for aligning CPC-CCS and CPC-CSO. For this purpose, all the OA systems have been executed on a subset of CPC, CCS, and CSO, respectively. The evaluation was then conducted on a manually generated reference alignment for each ontology pair, i.e. CPC_{subset}-CCS_{subset} and CPC_{subset}-CSO_{subset}. LogMap turned out to be the best suited OA system to align CPC with CCS. Further, to align CPC with CSO, AML proved to be appropriate. Also, when evaluating the systems on the *Anatomy* test case from the OAEI, LogMap and AML showed their effectiveness.

This chapter describes and evaluates the final alignment for CPC-CCS and CPC-CSO obtained by LogMap resp. AML. The evaluation of these alignments answers the third and last research question, namely:

RQ3: What is the performance of the best performing approach on the ground truth when applying it to generate a full alignment between CPC and CCS/CSO?

The remainder of this chapter is structured as follows: Section 6.1 describes the approach for evaluating the alignments. The subsequent section describes major observations that have arisen during the evaluation. The actual results are then presented in Section 6.3. The chapter concludes with key take-aways and a brief summary in Section 6.4.

6.1 Approximating Metrics

To the best of our knowledge no alignment between CPC and CCS/CSO has been generated and evaluated in previous works. As a consequence, there is no alignment against which a new alignment could be compared. Obviously, a gold standard isn't available either. Therefore, some way is required to evaluate the alignments that does not rely on reference alignments.

Typically, the correspondences of an alignment have to be manually looked into by the user in order to assess their quality [68]. Based on a manual assessments and a partial (possibly empty) gold standard, Chen et al. [16] described how *precision* and *recall* can be approximated for alignments where no gold standard exists. In a nutshell, the authors proposed to approximate *precision* through randomly drawing correspondences from an alignment and evaluating these correspondences for their correctness. Regarding *recall* an estimation of the actual correspondences is necessary, therefore Chen et al. [16] proposed to construct the union over the output of multiple OA systems. Afterwards, correspondences can be drawn at random and their correctness can be assessed. In this way it is possible to approximate the proportion of true correspondences in a larger sample of correspondences. Next, the approximation procedures are explained in more detail.

To approximate *precision* Chen et al. [16] use the following:

$$Precision^{\approx} = \frac{TP_{G,M}}{|M|} = \frac{|M \cap G| + \frac{|S_v|}{|S|} \times |M \setminus G|}{|M|} \tag{6.1}$$

G is a partial (possibly empty) gold standard. Here, the ground truth that was generated to evaluate the OA systems in Chapter 5 was used as G. M corresponds to the mappings generated by a system (either LogMap or AML depending on the task). S is a randomly chosen subset of $M \setminus G$, and $S_v \subseteq S$ constitutes the true positives of S. In this work $|S| = 100$ is chosen.

Regarding *recall*, Chen et al. [16] stated the following:

$$Recall^{\approx} = \frac{TP_{G,M}}{|G| + \frac{|S'_v|}{|S'|} \times |M' \setminus G|} \tag{6.2}$$

$TP_{G,M}$ and G are already known from $Precision^{\approx}$. M', on the other hand, comprises the union of mappings of multiple OA systems. S' and S'_v are then analogously defined to S and S_v, i.e. a random subset of M' and true mappings, respectively. This time a smaller S' was chosen, namely $|S'| = 50$. To obtain M' it was necessary to have a variety of different matchers that have, in the ideal case, high *recall*. However, if we consider the runtimes of Chapter 5 (Tables 5.3, 5.6, and 5.8), then it becomes clear that the only systems that are feasible to execute on the whole ontologies are LogMapLt, Baseline, LogMap, LogMap$_{Exp}$, AML, and probably LXLHMeta. All these systems have achieved only a low *recall* on the ground truth - see Chapter 5 (Tables 5.1 and 5.4). Further, when inspecting the heatmaps (Figures 5.1 and 5.2) of output mappings we can see that they all generate similar mappings, implying that the union over all systems would not increase the *recall* significantly. Due to these considerations, the reported *recall* approximations in this chapter shall be treated with caution.

Having approximated *precision* and *recall*, $f1 - score^{\approx}$ can be computed as well using the standard formula:

$$f1 - score^{\approx} = \frac{2 \times precision^{\approx} \times recall^{\approx}}{precision^{\approx} + recall^{\approx}} \tag{6.3}$$

6.2 Manual Assessment

To calculate the performance approximations outlined in the previous section, it is necessary to manually assess the correctness of each correspondence in S and S', thus arriving at S_v and S'_v. Note that this process has to be performed twice, one time for CPC-CCS and one time for CPC-CSO. The process of determining S_v/S'_v is described next. Subsequently observations are listed that appeared during the manual assessment of S. Regardless of CPC-CCS or CPC-CSO, the process and the observations were essentially the same. Therefore, there is no distinction between both cases in the remainder of this section.

Construction of S_v and S'_v. When creating S_v and S'_v the rules employed during ground truth construction (see Section 5.1) have been considered here again, however, for the vast majority of correspondences it was enough to compare the two hierarchies, in which the two involved concepts reside, to determine whether they represent a valid mapping or not.

Still, often it was quite difficult to determine whether two concepts represent the same notion as this, obviously, lies within the eye of the beholder to some degree. For example, the concept with label *bluetooth* in CPC is a sub-concept of *protocol or standard connector for transmission of analog or digital data to or from an electrophonic musical instrument*, while in CSO the concept with label *bluetooth* is about mobile devices. Of course, both concepts refer to the same underlying technology, still, its fields of application deviate heavily. Typically in such scenarios, such correspondences have not been considered as equivalent. This might be one of the reasons why $precision^{\approx}$ generally tends to be lower and $recall^{\approx}$ tends to be larger (in both alignment tasks, namely CPC-CCS and CPC-CSO).

A further fact that might have additionally deteriorated $precision^{\approx}$ and increased $recall^{\approx}$ is the fact that several times it was not within the expertise of the author to determine whether a correspondence is accurate or not. This was mostly the case if there the two concepts involved in a particular correspondence were about non-cs domains such as chemicals. In such a case, typically 0 was assigned - i.e. not considered as a true match.

To obtain an evaluation outcome that is as transparent as possible, for each correspondence a justification was given why the underlying correspondence was rejected/approved. The results of this manual assessment in conjunction with justifications can be found as tsv files in the GitHub repository of this thesis. Note that the last column of the respective tsv files indicates uncertainty. For example, uncertainty might originate from mappings in other domains than computer science where the author's knowledge is limited.

Characteristics of correspondences in S. When inspecting the sample of correspondences, regardless of LogMap (CPC-CCS) or AML (CPC-CSO), there were a few observations regarding characteristics of the correspondences that are outlined in the following.

LogMap and AML put too much focus on lexical correlation and seem to neglect structural information. This becomes evident when considering the top reason for rejection, namely that two concepts have been aligned that have the same name, but a completely different semantic meaning because they reside in two disjoint hierarchies. An appropriate example for this behaviour is represented by the following correspondence that was found by AML: the CPC concept *G16Y10-75* with label *information technology; communication* was mapped to the CSO concept *information technology*. While, at the first glance, this correspondence might seem like a equivalent relationship, when inspecting the two hierarchies it becomes clear that the CPC concept is about the economic sector, while the CSO concept represents a field of computer science research. LogMap and AML have generated a large variety of similar correspondences that have then been rejected during the manual assessment. The limitation originating from such examples was not very serious during the alignment of the sub-ontologies (Chapter 5) as all the sub-ontologies represented a similar domain to begin with. However, especially the full CPC ontology comprises a large variety different fields, despite the fact that the sections A,B,C,D,E,F, and H have been identified as unrelated already during the pre-processing stage (see Section 3.3).

Similar to the latter observation, LogMap and AML essentially only predicted correspondences involving concepts with very similar or even identical labels.

There also were quite a few correspondences that point at the lack of semantic understanding of LogMap and AML, such as *single input, plural outputs* (CPC) was mapped to *multiple input single outputs* (CSO). This points to a limitation of the employed matchers, namely LogMap and AML, that they are not able to grasp the semantics of a label instead they purely rely on lexical information (at least when dealing with labels).

6.3 Results

As stated in Chapter 3, CCS, CSO, and CPC comprise 2113, 14290, and 53849 classification entries, respectively. Table 6.1 shows the runtimes of each approach that has been executed to generate the alignments for CPC-CCS and CPC-CSO. Note that LXLHMeta was killed after > 5 hours on CPC-CSO, therefore the resulting alignment is not available for *recall* estimation of the CPC-CSO alignment. Further, the alignment generated by LXLHMeta on CPC-CCS has also been excluded for CPC-CCS *recall* estimation since it predicted too many (mostly wrong) correspondences. Therefore a sample of 50 correspondences (S') would have had overwhelmingly many correspondences from LXLHMeta and would, hence, be very biased towards that specific matcher (with LXLHMeta the union (M') would comprise 8549 correspondences, while without it would only comprise 376 correspondences). Note in bold we see the approaches for which *precision* is approximated, i.e. the approaches that have been found most suited to align CPC with CCS and CPC with CSO.

Appraoch	Runtime in sec	
	CPC-CCS	CPC-CSO
LogMap	**146**	435
LogMapLt	12	13
LogMapExp	99	440
AML	67	**83**
Baseline	551	755
LXLHMeta	4990	>19736

Table 6.1: Runtimes to generate the final alignments - in bold the runtime of the most suited matcher for the respective alignment task

6.3.1 CPC-CCS

This section approximates *precision*, *recall*, as well as *f1-score* for the CPC-CCS alignment.

Precision. LogMap found 94 correspondences when aligning CPC with CCS. These correspondences represent M. Four of these correspondences were already part of the related ground truth, therefore of $|M \setminus G|$ was equal to 87 and $|M \cap G|$ was found to be equal to 7. Because there are less than 100 correspondences, all of them were added to S. Out of these 87 mappings 42 were found to be valid. These 42 correspondences form S_v. Using all these values, it is possible to approximate *precision* using the formula 6.1. Thus:

$$Precision^{\approx} = \frac{TP_{G,M}}{|M|} = \frac{|M \cap G| + \frac{|S_v|}{|S|} \times |M \setminus G|}{|M|} = \frac{7 + \frac{42}{87} \times 87}{94} \approx 0.52$$

Recall. To approximate *recall* the union over the alignments generated by LogMap, LogMapLt, LogMap$_{Exp}$, AML, and Baseline has been built. This union forms M'. The size of M' equals to 376. M' contained 13 correspondences from the ground truth, thus $|M' \setminus G| = 363$. Remember that $|G| = 42$. Out of $M' \setminus G$ a random sample of 50 correspondences has been drawn. 21 of these random correspondences were correct mappings, therefore $|S'| = 50$ and $|S'_v| = 21$. Hence:

$$Recall^{\approx} = \frac{TP_{G,M}}{|G| + \frac{|S'_v|}{|S'|} \times |M' \setminus G|} = \frac{49}{42 + \frac{21}{50} \times 363} \approx 0.25$$

Based on $Precision^{\approx}$ and $Recall^{\approx}$, $f1 - score^{\approx}$ is:

$$f1 - score^{\approx} = \frac{2 \times precision^{\approx} \times recall^{\approx}}{precision^{\approx} + recall^{\approx}} \approx 0.34$$

Interpretation. When comparing the metrics obtained in Chapter 5 one can observe that the *precision* metrics deviate significantly, i.e. 0.52 (in this chapter) vs. 0.75 (during the experiments). Reasons for this discrepancy were already outlined in Section 6.2, e.g. it is far more likely that two identical words in the ground truth represent actual true correspondences than when aligning the whole ontologies.

Regarding *recall* estimation the values are very similar, i.e. 0.25 (in this chapter) vs. 0.21 (during the experiments). It is, however, very likely that the actual *recall* on the full alignment is significantly lower since all the correspondences, that have been used for *recall* estimation, are biased towards mappings found by lexical matchers (LogMapLt, LogMap$_{Exp}$, AML, and Baseline).

Due to the fact that the *recall* estimations were very similar to those computed on the ground truth, the estimated *f1-score* is also very similar, i.e. 0.34 vs. 0.33.

6.3.2 CPC-CSO

This section approximates *precision*, *recall*, as well as *f1-score* for the CPC-CSO alignment.

Precision. Regarding CPC-CSO, AML found 1334 correspondences. Out of the sample S ($|S| = 100$), 45 were classified as correct correspondences ($|S_v| = 45$). $|M \cap G|$ was equal to 21, while $|M \setminus G|$ corresponded to 1313. Therefore, $Precision^{\approx}$ for CPC-CSO is:

$$Precision^{\approx} = \frac{TP_{G,M}}{|M|} = \frac{|M \cap G| + \frac{|S_v|}{|S|} \times |M \setminus G|}{|M|} = \frac{21 + \frac{45}{100} \times 1313}{1334} \approx 0.46$$

Recall. The union of all alignments on CPC-CSO was equal to 2608 (corresponds to M'). Out of $M' \setminus G$ (size 2582), a sample of size 50 was randomly chosen (denoted by S'). When inspecting the correspondences in S', 19 were found to be correct (denoted by S'_v). The ground truth for CPC-CSO comprises 103 correspondences. Therefore, *recall* is approximated through:

$$Recall^{\approx} = \frac{TP_{G,M}}{|G| + \frac{|S'_v|}{|S'|} \times |M' \setminus G|} = \frac{21 + \frac{45}{100} \times 1334}{103 + \frac{19}{50} \times 2582} \approx 0.56$$

Based on $Precision^{\approx}$ and $Recall^{\approx}$, $f1 - score^{\approx}$ is:

$$f1 - score^{\approx} = \frac{2 \times precision^{\approx} \times recall^{\approx}}{precision^{\approx} + recall^{\approx}} \approx 0.51$$

Interpretation. Similar as with CPC-CCS, also here the *precision* metric deviates significantly from its counterpart in Chapter 5, i.e. 0.46 vs. 0.83. Essentially the same reasons as in CPC-CCS apply here as well.

The estimated *recall* measure, on the other hand, is larger than its counterpart on the ground truth, i.e. 0.56 vs. 0.34. The most probable reasons are the bias towards lexical matchers in the recall estimation sample, and the reasons outlined in Section 6.2.

The estimated *precision* value is smaller than its equivalent on the ground truth, while the estimated *recall* sample is larger, therefore the estimated *f1-score* is very similar, i.e. 0.51 vs. 0.48.

6.4 Summary

To conclude the analysis of the alignments for CPC-CCS and CPC-CSO, this section lists the key results.

Firstly, it became clear that LogMap and AML failed to take structural information of the ontologies into consideration. This became evident when observing the number of correspondences between concepts with similar labels in different knowledge branches. This behaviour of the OA systems was the main reason for the unexpected low *precision* values. These values were unexpected since the performance on the ground truths, that was described in Chapter 5, gave reason to hope for decent *precision* figures.

Secondly, as it was already expected by observing the predicted correspondences on the ground truth, LogMap and AML predicted for the most part only correspondences that involved concepts with very similar/identical labels.

The approximated *recall* measures, calculated in this chapter, were higher than the achieved *recall* values on the ground truth - as described in Chapter 5. However, it is important to note that the reported *recall* measures need to be considered with caution since the underlying estimation process is connected with various limitations and assumptions.

Even though there are limitations in the *precision* & *recall* estimation processes, the quality indications calculated for the two final alignments provide a reasonable metric to assess the goodness of the two alignments. Both alignments represent a useful, but limited, starting point in aligning the patent ontology CPC with the two computer science ontologies CCS and CSO.

CHAPTER 7

Summary

This chapter is the last of this work, therefore it is the appropriate place to draw conclusions, as well as to provide an outlook towards possible future work. The first section summarizes the most important steps of the work, describes the major outcomes/results for each step, and derives the limitations that are associated with the individual steps. The next and final section of this thesis points at several open issues and, thus, provides directions for future work.

7.1 Conclusion & Limitations

Ontologies about similar domains are very often heavily heterogeneous. Ontologies may deviate in their granularity, structure, relationships among concepts, and in many other aspects. These discrepancies foster the need for automated OA solutions. This work investigated various OA approaches regarding the alignment of the patent ontology CPC with the computer science ontologies CCS and CSO. The eventual goal was to deliver an alignment between these ontologies.

The first major step was to conduct a literature review. The literature review was oriented at the literature review template proposed in [13]. As search engine *Google Scholar* was employed. To filter the literature, only literature that was published after 2018 was considered and the string *ontology alignment* or *ontology matching* had to be contained within each publication's title. The outcome of this initial step was a set of publications proposing new OA solutions. The resulting publications were then re-implemented and passed to the next step of this thesis. The search process imposed two limitations: (a) the resulting set of publications are biased towards publications that are findable through *Google Scholar*, (b) due to the fact that only literature after 2018 and literature whose titles contain either the string *ontology alignment* or *ontology matching* were considered, it might have happened that promising literature was missed.

Based on the results of the literature survey, the second step was about the evaluation of the found approaches. To gauge the performance of each approach, the manual creation of two ground truths was necessary, i.e. one ground truth for the CPC-CCS pair and one for CPC-CSO. Executing each of the identified approaches on the ground truths resulted in the realization that the SOTA matchers LogMap and AML are the most suited matchers to align CPC with CCS and CPC with CSO. More precisely, LogMap was found to be best suited for CPC-CCS, resp. AML for CPC-CSO. The major limitation of this step stems from the manual creation of the ground truths. Finding correspondences between concepts is to some degree subjective to the person that creates these correspondences. Although best effort was made to ensure that this step is as objective and reproducible as possible, e.g. through the definition of assignment rules, the outcome might still deviate when compared to the outcome of another person. When evaluating OA systems for a specific use case this is, however, a very common limitation as the results of the experiments conducted in [66] show.

To deliver the final contribution of this work, the full preprocessed ontologies were aligned using the best suited matcher for the respective ontology pair, i.e. LogMap for CPC-CCS and AML for CPC-CSO. To evaluate the resulting alignments, the *precision* and *recall* estimation procedures described in [16] were considered. The results show that both alignments are a useful starting point for a sophisticated alignment between the ontologies, but are still of limited quality. Note that the estimation procedures involved a manual assessment of randomly selected alignments and impose, as a consequence, the same limitation as the ground truths.

Although, the achieved results of this thesis provide a first decent building block in aligning patent classifications with computer science terms, the results are connected with various limitations as explained in the previous paragraphs. These limitations lead to several possible future research directions as well as ideas to enhance the quality of the alignments.

7.2 Outlook

This section provides an outlook towards possible future work.

7.2.1 Embedding of Additional Knowledge

All the three ontologies were preprocessed to schema-level ontologies whereby, as a result, each ontological concept has only *rdfs:label* annotations and is related to other concepts exclusively through *owl:subClassOf* relations. Many of the considered OA systems might deliver enhanced performances if more information was included. Especially, adding disjointness constraints, such as *owl:disjointWith*, and adding descriptions to concepts, e.g. through *rdfs:comment* annotations, might prove to be very beneficial in regards to the predictive power of each matcher. To add additional constraints to all three ontologies, additional manual work would be necessary since the three ontologies are not shipped

with such relations. For example, one might add *owl:disjointWith* relations between concepts that represent disjoint knowledge branches. To add descriptions additional preprocessing strategies would be necessary. For example, to embed the descriptions of the CPC classification entries the related descriptions would have to be retrieved from their respective sources and be embedded into the related concepts.

In this work, no external sources were utilized with the exception of WordNet and pre-trained language models, such as Word2Vec. To further include more knowledge, it might make sense to utilize auxiliary ontologies and/or additional external data sources. As an example for a possibly useful auxiliary ontology, one might name the IPC ontology that is already referenced from various CPC classification entries. Using such an additional ontology might enable to embody further descriptions and labels for each concept.

7.2.2 Increasing of Variety of Approaches

During the literature survey several restrictions have been made towards the properties of each matcher. Lifting one or more of those restrictions would enable the employment of additional OA systems. As a possible relaxation one might lift the requirement that each matcher must represent an unsupervised OA system. Due to the fact that a ground truth is now available, future work might try supervised matchers that are trained on the data from the ground truth. Another way to increase the variety of approaches might be to consider also publications from other search engines than *Google Scholar*.

Although the implementations were already in-part adapted to the characteristics of this use case, the adaptations were committed rather cautiously and for the most part the implementations are heavily oriented at the related publications. A possible point for future work might be to adapt the approaches to be better suited to the characteristics of each ontology, such as through refinement of parameters.

List of Figures

List of Tables

Acronyms

AML AgreementMakerLight. 28–30, 71

BERT Bidirectional Encoder Representation from Transformers. 35, 36, 46, 52

CCS ACM Computing Classification System. 2–7, 13–19, 37, 41, 43–45, 48, 50, 52, 55, 57, 59, 61–65, 67, 68

CPC Cooperative Patent Classification. 2–7, 13, 15–19, 37, 41, 43–45, 48, 50, 52, 53, 55–57, 59, 61–65, 67–69

CSO Computer Science Ontology. 2–7, 13–19, 37, 41, 43–45, 52, 53, 56, 57, 59, 61, 62, 64, 65, 67, 68

EA Evolutionary Algorithm. 38

GA Genetic Algorithm. 38, 40

IDF Inverse Document Frequency. 31, 36

IPC International Patent Classification. 16, 69

LSTM Long Short-Term Memory. 34

OA Ontology Alignment. 2–7, 9–15, 19, 21, 23–33, 35, 38, 41, 43, 44, 46–48, 52, 53, 55, 57, 59, 60, 65, 67–69

OAEI Ontology Alignment Evaluation Initiative. 2, 3, 5, 23, 26–29, 32, 33, 37, 38, 46, 48, 53, 55, 59

OWL Web Ontology Language. 9, 10, 14, 16, 33

RDF Resource Description Framework. 9, 10, 13, 14, 16

SA Simulated Annealing. 29–31

SKOS Simple Knowledge Organization System. 13, 14

SOTA State-of-the-Art. 12, 23, 25–27, 32, 57, 68

TF Term Frequency. 31

Bibliography

[1] Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha. Using ontologies in the semantic web: A survey. In *Ontologies*, pages 79–113. Springer, 2007.

[2] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

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

[4] Xingsi Xue and Jiawei Lu. A compact brain storm algorithm for matching ontologies. *Ieee Access*, 8:43898–43907, 2020.

[5] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662–673, 2002.

[6] Angelo A Salatino, Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. The computer science ontology: a large-scale taxonomy of research areas. In *International Semantic Web Conference*, pages 187–205. Springer, 2018.

[7] Zoe Kotti, Georgios Gousios, and Diomidis Spinellis. Impact of software engineering research in practice. *arXiv preprint arXiv:2204.03366*, 2022.

[8] CPC cooperative patent classification - ontology meta information. Retrieved from `https://www.epo.org/searching-for-patents/helpful-resources/first-time-here/classification/cpc.html`, 2022. Accessed: 2022-08-04.

[9] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[10] Ian Horrocks, Jiaoyan Chen, and L Jaehun. Tool support for ontology design and quality assurance. In *ICBO 2020 integrated food ontology workshop (IFOW)*, 2020.

[11] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 527–541. Springer, 2013.

[12] M Pour, Alsayed Algergawy, Florence Amardeilh, Reihaneh Amini, Omaima Fallatah, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Pascal Hitzler, et al. Results of the ontology alignment evaluation initiative 2021. In *CEUR Workshop Proceedings 2021*, volume 3063, pages 62–108. CEUR, 2021.

[13] Jorge Biolchini, Paula Gomes Mian, Ana Candida Cruz Natali, and Guilherme Horta Travassos. Systematic review in software engineering. *System engineering and computer science department COPPE/UFRJ, Technical Report ES*, 679(05):45, 2005.

[14] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[15] Barbara A Kitchenham, Tore Dyba, and Magne Jorgensen. Evidence-based software engineering. In *Proceedings. 26th International Conference on Software Engineering*, pages 273–281. IEEE, 2004.

[16] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, Denvar Antonyrajah, Ali Hadian, and Jaehun Lee. Augmenting ontology alignment by semantic embedding and distant supervision. In *European Semantic Web Conference*, pages 392–408. Springer, 2021.

[17] Majid Mohammadi, Wout Hofman, and Yao-Hua Tan. Simulated annealing-based ontology matching. *ACM Transactions on Management Information Systems (TMIS)*, 10(1):1–24, 2019.

[18] Jeff Z Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009.

[19] Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, Lynn Andrea Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10(2):1–53, 2004.

[20] Jiaoyan Chen, Pan Hu, Ernesto Jimenez-Ruiz, Ole Magnus Holter, Denvar Antonyrajah, and Ian Horrocks. Owl2vec*: Embedding of owl ontologies. *Machine Learning*, 110(7):1813–1845, 2021.

[21] Erik B Myklebust, Ernesto Jimenez-Ruiz, Jiaoyan Chen, Raoul Wolf, and Knut Erik Tollefsen. Knowledge graph embedding for ecotoxicological effect prediction. In *International Semantic Web Conference*, pages 490–506. Springer, 2019.

[22] Ian Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51 (12):58–67, 2008.

[23] Jérôme Euzenat et al. Semantic precision and recall for ontology alignment evaluation. In *Ijcai*, volume 7, pages 348–353, 2007.

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

[25] Sophie Neutel and Maaike HT de Boer. Towards automatic ontology alignment using bert. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

[26] Elodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez, and Cassia Trojahn. Survey on complex ontology matching. *Semantic Web*, 11(4):689–727, 2020.

[27] Jaewoo Kang and Jeffrey F Naughton. On schema matching with opaque column names and data values. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 205–216, 2003.

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

[29] Jorge Martinez-Gil and José F Aldana-Montes. Evaluation of two heuristic approaches to solve the ontology meta-matching problem. *Knowledge and Information Systems*, 26(2):225–247, 2011.

[30] ACM computing classification system. Retrieved from `https://dl.acm.org/ccs`, 2012. Accessed: 2022-11-04.

[31] Bernard Rous. Major update to acm's computing classification system. *Communications of the ACM*, 55(11):12–12, 2012.

[32] Alistair Miles and Sean Bechhofer. Skos simple knowledge organization system reference. *W3C recommendation*, 2009.

[33] Francesco Osborne and Enrico Motta. Klink-2: integrating multiple web sources to generate semantic topic networks. In *International Semantic Web Conference*, pages 408–424. Springer, 2015.

[34] Francesco Osborne, Enrico Motta, and Paul Mulholland. Exploring scholarly data with rexplore. In *International semantic web conference*, pages 460–477. Springer, 2013.

[35] CSO computer science ontology - about. Retrieved from `https://cso.kmi.open.ac.uk/about`, 2022. Accessed: 2022-11-28.

[36] CPC cooperative patent classification - home. Retrieved from `https://www.cooperativepatentclassification.org/home`, 2022. Accessed: 2022-11-28.

[37] David Beckett. Rdf 1.1 n-triples. *URL: https://www. w3. org/TR/n-triples*, 2014.

[38] Yuan He, Jiaoyan Chen, Denvar Antonyrajah, and Ian Horrocks. Bertmap: A bert-based ontology alignment system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5684–5691, 2022.

[39] Jaydeep Chakraborty, Srividya K Bansal, Luca Virgili, Krishanu Konar, and Beyza Yaman. Ontoconnect: Unsupervised ontology alignment with recursive neural network. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1874–1882, 2021.

[40] Amir Laadhar, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faiez Gargouri. 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)*, pages 169–174, 2019.

[41] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Deepalignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 787–798, 2018.

[42] Jiawei Lu, Xingsi Xue, Guoxiang Lin, and Yikun Huang. A new ontology meta-matching technique with a hybrid semantic similarity measure. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 37–45. Springer, 2020.

[43] Qing Lv, Chengcai Jiang, and He Li. Solving ontology meta-matching problem through an evolutionary algorithm with approximate evaluation indicators and adaptive selection pressure. *IEEE Access*, 9:3046–3064, 2020.

[44] Zhaoming Lv and Rong Peng. A novel meta-matching approach for ontology alignment using grasshopper optimization. *Knowledge-Based Systems*, 201:106050, 2020.

[45] Qing Lv, Jinyuan Shi, Huanting Shi, and Chengcai Jiang. A novel compact fireworks algorithm for solving ontology meta-matching. *Applied Intelligence*, pages 1–24, 2022.

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

[47] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

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

[49] Alsayed Algergawy, Daniel Faria, Alfio Ferrara, Irini Fundulaki, Ian Harrow, Sven Hertling, Ernesto Jiménez-Ruiz, Naouel Karam, Abderrahmane Khiat, Patrick Lambrix, et al. Results of the ontology alignment evaluation initiative 2019. In *CEUR Workshop Proceedings*, volume 2536, pages 46–85, 2019.

[50] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[51] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[52] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*, 1994.

[53] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[54] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[56] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[57] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[58] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[59] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2):2, 2011.

[60] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*, number CONF, 2011.

[61] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[62] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[63] Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017.

[64] Sven Hertling, Jan Portisch, and Heiko Paulheim. Melt-matching evaluation toolkit. In *International conference on semantic systems*, pages 231–245. Springer, Cham, 2019.

[65] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The alignment api 4.0. *Semantic web*, 2(1):3–10, 2011.

[66] Anna Tordai, Jacco Van Ossenbruggen, Guus Schreiber, and Bob Wielinga. Let's agree to disagree: on the evaluation of vocabulary alignment. In *Proceedings of the sixth international conference on knowledge capture*, pages 65–72, 2011.

[67] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks. Logmap and logmaplt results for oaei 2012. *Ontology Matching*, 152, 2013.

[68] Jérôme Euzenat and Chan Le Duc. Methodological guidelines for matching ontologies. In *Ontology engineering in a networked world*, pages 257–278. Springer, 2012.