Breaking-down the Ontology Alignment Task with a Lexical Index and Neural Embeddings

Ernesto Jiménez-Ruiz[1,2], Asan Agibetov[3], Matthias Samwald[3], Valerie Cross[4]

[1] The Alan Turing Institute, London, United Kingdom
[2] Department of Informatics, University of Oslo, Norway
[3] Section for Artificial Intelligence and Decision Support, Medical University of Vienna, Vienna, Austria
[4] Miami University, Oxford, OH 45056, United States

Abstract. Large ontologies still pose serious challenges to state-of-the-art ontology alignment systems. In this paper we present an approach that combines a lexical index, a neural embedding model and locality modules to effectively divide an input ontology matching task into smaller and more tractable matching (sub)tasks. We have conducted a comprehensive evaluation using the datasets of the Ontology Alignment Evaluation Initiative. The results are encouraging and suggest that the proposed methods are adequate in practice and can be integrated within the workflow of state-of-the-art systems.

Keywords: ontology alignment, divide and conquer, module extraction, partitioning, neural embedding model, lexical index, matching subtasks

1 Introduction

The problem of (semi-)automatically computing an alignment between independently developed ontologies has been extensively studied in the last years [1, 2]. As a result, a number of sophisticated ontology alignment systems currently exist.[1] The Ontology Alignment Evaluation Initiative[2] (OAEI) [3] has played a key role in the benchmarking of these systems by facilitating *(i)* their comparison on the same basis, and *(ii)* the reproducibility of the evaluation and results. The OAEI includes different tracks organised by different research groups. Each track contains one or more matching tasks involving small-size (*e.g.*, conference), medium-size (*e.g.*, anatomy), large (*e.g.*, phenotype) or very large (*e.g.*, largebio) ontologies.

Large ontologies still pose serious challenges to ontology alignment systems. For example, only 6 out of 10 systems participating in the OAEI 2017 *largebio track* were able to complete the largest tasks [3]. OAEI systems are typically able to cope with small and medium size ontologies, but fail to complete large tasks in a given time frame and/or with the available resources (*e.g.*, memory). Prominent examples across the OAEI campaigns are: *(i)* YAM++ version 2011 (best results in *conference* track, but

[1] Ontology matching surveys and approaches: `http://ontologymatching.org/`
[2] OAEI evaluation campaigns: `http://oaei.ontologymatching.org/`

failed to complete the *anatomy* task); *(ii)* CODI version 2011.5 (best results in *anatomy* but could not cope with the *largebio* track); *(iii)* MAMBA version 2015 (top system in the *conference* track but could not complete the *anatomy* track); *(iv)* FCA-Map version 2016 (completed both *anatomy* and *phenotype* tasks but did not complete the largest *largebio* tasks); and *(v)* POMap version 2017 (one of the top systems in *anatomy* but could not finish the largest *largebio* tasks).

Shvaiko and Euzenat [1] list some potential solutions to address the challenges that large ontologies pose to ontology alignment systems, namely: parallelization, distribution, approximation, partitioning and optimization. In this paper we propose a novel method to effectively divide the matching task into several (independent) smaller (sub)tasks. This method relies on an efficient lexical index (as in LogMap [4]), a neural embedding model [5] and locality modules [6]. Unlike other state-of-the-art approaches, our method provides guarantees about the preservation of the coverage of the relevant ontology alignments as defined in Section 2.2.

The remainder of the paper is organised as follows. Section 2 introduces the main concepts that will be used in the paper. Section 3 presents the methods and strategies to divide the ontology matching task into a set of smaller subtasks. The conducted evaluation is provided in Section 4. Section 5 summarizes the related literature. Finally, Section 6 concludes the paper and suggests some lines of immediate future research.

2 Preliminaries

In this section we introduce the background concepts that are used throughout the paper.

2.1 Basic definitions

A *mapping* (also called *match* or *correspondence*) between entities[3] of two ontologies[4] $\mathcal{O}_1$ (*i.e.*, source) and $\mathcal{O}_2$ (*i.e.*, target) is typically represented as a 4-tuple $\langle e_1, e_2, r, c\rangle$ where e_1 and e_2 are entities of $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively; $r \in \{\sqsubseteq, \sqsupseteq, \equiv\}$ is a semantic relation; and c is a confidence value, usually, a real number within the interval $(0, 1]$. In our approach we simply consider mappings as a pair $\langle e_1, e_2\rangle$. An ontology *alignment* is a set of mappings $\mathcal{M}$ between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$.

An ontology *matching task* $\mathcal{MT}$ is composed of a pair of ontologies $\mathcal{O}_1$ (typically called source) and $\mathcal{O}_2$ (typically called target) and possibly an associated *reference alignment* $\mathcal{M}^{RA}$. The objective of a matching task is to discover an overlapping of $\mathcal{O}_1$ and $\mathcal{O}_2$ in the form of an alignment $\mathcal{M}$. The *size or search space* of a matching task is typically bound to the size of the Cartesian product between the entities of the input ontologies: $|Sig(\mathcal{O}_1)| \times |Sig(\mathcal{O}_2)|$ being $Sig(\mathcal{O})$ the signature (*i.e.*, entities) of $\mathcal{O}$.

An ontology *matching system* is a program that, given as input the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ of a matching task, generates an ontology alignment $\mathcal{M}^S$.

[3] We refer to (OWL 2) classes, data and object properties and named individuals as entities.
[4] We assume ontologies are expressed in OWL 2 [7].

The standard evaluation measures for an alignment $\mathcal{M}^S$ are *precision* (P), *recall* (R) and *f-measure* (F) computed against a reference alignment $\mathcal{M}^{RA}$ as follows:

$$P = \frac{|\mathcal{M}^S \cap \mathcal{M}^{RA}|}{|\mathcal{M}^S|},\ R = \frac{|\mathcal{M}^S \cap \mathcal{M}^{RA}|}{|\mathcal{M}^{RA}|},\ F = 2 \cdot \frac{P \cdot R}{P + R} \tag{1}$$

2.2 Matching subtasks and quality measures: size ratio and coverage

We denote *division* of an ontology matching task $\mathcal{MT}$, composed by the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, as the process of finding matching subtasks $\mathcal{MT}_i = \langle \mathcal{O}_1^i, \mathcal{O}_2^i \rangle$ (with i=1,...,n), where $\mathcal{O}_1^i \subset \mathcal{O}_1$ and $\mathcal{O}_2^i \subset \mathcal{O}_2$. The size of the matching subtasks aims at being smaller than the original task in terms of search space. Let $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1, \ldots, \mathcal{MT}_n\}$ be the result of dividing a matching task $\mathcal{MT}$. The *size ratios* of the matching subtasks $\mathcal{MT}_i$ and $\mathcal{D}^n_{\mathcal{MT}}$ are computed as follows:

$$\mathsf{SizeRatio}(\mathcal{MT}_i, \mathcal{MT}) = \frac{|Sig(\mathcal{O}_1^i)| \times |Sig(\mathcal{O}_2^i)|}{|Sig(\mathcal{O}_1)| \times |Sig(\mathcal{O}_2)|} \tag{2}$$

$$\mathsf{SizeRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{MT}) = \sum_{i=1}^{n} \mathsf{SizeRatio}(\mathcal{MT}_i, \mathcal{MT}) \tag{3}$$

The ratio $\mathsf{SizeRatio}(\mathcal{MT}_i, \mathcal{MT})$ is expected to be less than 1.0 while the aggregation $\sum_{i=1}^{n} \mathsf{SizeRatio}(\mathcal{MT}_i, \mathcal{MT})$, being n the number of matching subtasks, can be greater than 1.0 (as matching subtasks may overlap), that is, the aggregated size of the matching subtasks may be larger than the original task size in terms of (aggregated) search space.

The *coverage* of the matching subtask aims at providing guarantees about the preservation of the (potential) outcomes of the original matching task. That is, it indicates if the relevant ontology alignments in the original matching task can still be computed with the matching subtasks. The coverage is calculated with respect to a relevant alignment $\mathcal{M}$, possibly the reference alignment $\mathcal{M}^{RA}$ of the matching task if it exists. The formal notion of coverage is given in Definitions 1 and 2.

Definition 1 (Coverage of a matching task). *Let $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ be a matching task and $\mathcal{M}$ an alignment. We say that a mapping $m = \langle e_1, e_2 \rangle \in \mathcal{M}$ is covered by the matching task if $e_1 \in Sig(\mathcal{O}_1)$ and $e_2 \in Sig(\mathcal{O}_2)$. The coverage of $\mathcal{MT}$ w.r.t. $\mathcal{M}$ (denoted as $\mathsf{Coverage}(\mathcal{MT}, \mathcal{M})$) represents the set of mappings $\mathcal{M}' \subseteq \mathcal{M}$ covered by $\mathcal{MT}$.*

Definition 2 (Coverage of the matching task division). *Let $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1, \ldots, \mathcal{MT}_n\}$ be the result of dividing a matching task $\mathcal{MT}$ and $\mathcal{M}$ an alignment. We say that a mapping $m \in \mathcal{M}$ is covered by $\mathcal{D}_{\mathcal{MT}}$ if m is at least covered by one of the matching subtask $\mathcal{MT}_i$ (with i=1,...,n) as in Definition 1. The coverage of $\mathcal{D}_{\mathcal{MT}}$ w.r.t. $\mathcal{M}$ (denoted as $\mathsf{Coverage}(\mathcal{D}_{\mathcal{MT}}, \mathcal{M})$) represents the set of mappings $\mathcal{M}' \subseteq \mathcal{M}$ covered by $\mathcal{D}_{\mathcal{MT}}$. The coverage will typically be given as a ratio with respect to the (covered) alignment:*

$$\mathsf{CoverageRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{M}) = \frac{|\mathsf{Coverage}(\mathcal{D}_{\mathcal{MT}}, \mathcal{M})|}{|\mathcal{M}|} \tag{4}$$

2.3 Locality-based modules in ontology alignment

Logic-based module extraction techniques compute ontology fragments that capture the meaning of an input signature with respect to a given ontology. In this paper we rely on bottom-locality modules [6], which will be referred to as locality-modules or simply as modules. Locality modules play an important role in ontology alignment tasks. For example, they provide the scope or context (*i.e.*, sets of *semantically related* entities [6]) for the entities in a given mapping or set of mappings as formally presented in Definition 3.

Definition 3 (Context of a mapping and an alignment). *Let $m = \langle e_1, e_2 \rangle$ be a mapping between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. We define the context of m (denoted as $\mathsf{Context}(m, \mathcal{O}_1, \mathcal{O}_2)$) as a pair of modules $\mathcal{O}'_1 \subseteq \mathcal{O}_1$ and $\mathcal{O}'_2 \subseteq \mathcal{O}_2$, where $\mathcal{O}'_1$ and $\mathcal{O}'_2$ include the semantically related entities to e_1 and e_2, respectively [6]. Similarly, the* context *for an alignment $\mathcal{M}$ between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ is denoted as $\mathsf{Context}(\mathcal{M}, \mathcal{O}_1, \mathcal{O}_2) = \langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle$, where $\mathcal{O}'_1$ and $\mathcal{O}'_2$ are modules including the semantically related entities for the entities $e_1 \in Sig(\mathcal{O}_1)$ and $e_2 \in Sig(\mathcal{O}_2)$ in each mapping $m = \langle e_1, e_2 \rangle \in \mathcal{M}$.*

2.4 Context as matching task

The context of an alignment between two ontologies represents the overlapping of these ontologies with respect to the aforesaid alignment. Intuitively, the ontologies in the context of an alignment will cover all the mappings in that alignment. Definition 4 formally presents the context of an alignment as the *overlapping* matching task to discover that alignment.

Definition 4 (Overlapping matching task). *Let $\mathcal{M}$ be an alignment between $\mathcal{O}_1$ and $\mathcal{O}_2$, and $\mathsf{Context}(\mathcal{M}, \mathcal{O}_1, \mathcal{O}_2) = \langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle$ the context of $\mathcal{M}$. We define $\mathcal{MT}^{\mathcal{M}}_{\mathcal{O}_1\text{-}\mathcal{O}_2} = \langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle$ as the overlapping matching task for $\mathcal{M}$. A matching task $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ can be reduced to the task $\mathcal{MT}^{\mathcal{M}}_{\mathcal{O}_1\text{-}\mathcal{O}_2} = \langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle$ without information loss in terms of finding $\mathcal{M}$.*

A matching system should aim at computing $\mathcal{M}$ with both the reduced task $\mathcal{MT}^{\mathcal{M}}_{\mathcal{O}_1\text{-}\mathcal{O}_2}$ and the original matching task $\mathcal{MT}$. For example, in the small OAEI *largebio* tasks [3] systems are given, instead of the original matching task (*e.g.*, whole FMA and NCI ontologies), the context of the reference alignment as a (reduced) overlapping matching task (*e.g.*, $\mathcal{MT}^{RA}_{\text{fma-nci}} = \mathsf{Context}(\mathcal{M}^{RA}_{\text{fma-nci}}, \mathcal{O}_{\text{FMA}}, \mathcal{O}_{\text{NCI}}) = \langle \mathcal{O}'_{\text{FMA}}, \mathcal{O}'_{\text{NCI}} \rangle$).

3 Methods

The approach presented in this paper relies on an 'inverted' lexical index (we will refer to this index as LexI), commonly used in information retrieval applications, and also used in ontology alignment systems like LogMap [4] or ServOMap [8].

Table 1: Inverted lexical index Lexl (left) and entity index (right). For readability, stemming techniques have not been applied and index values have been split into elements of $\mathcal{O}_1$ and $\mathcal{O}_2$. '-' indicates that the ontology does not contain entities for that entry.

Index key	**Index value**	
	Entities $\mathcal{O}_1$	**Entities $\mathcal{O}_2$**
{ acinus }	7661,8171	118081
{ mesothelial, pleural }	19987	117237
{ hamate, lunate }	55518	-
{ feed, breast }	-	113578,111023

ID	**URI**
7661	$\mathcal{O}_1$:Serous_acinus
8171	$\mathcal{O}_1$:Hepatic_acinus
19987	$\mathcal{O}_1$:Mesothelial_cell_of_pleura
55518	$\mathcal{O}_1$:Lunate_facet_of_hamate
118081	$\mathcal{O}_2$:Liver_acinus
117237	$\mathcal{O}_2$:Pleural_Mesothelial_Cell
113578	$\mathcal{O}_2$:Breast_Feeding
111023	$\mathcal{O}_2$:Inability_To_Breast_Feed

3.1 The lexical index Lexl

Lexl encodes the labels of all entities of the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, including their lexical variations (*e.g.*, preferred labels, synonyms), in the form of pairs *key-value* where the key is a set of words and the value is a set of entity identifiers[5] such that the set of words of the key appears in (one of) the entity labels. Table 1 shows a few example entries of Lexl for two input ontologies.

Lexl is created as follows. *(i)* Each label associated to an ontology entity is split into a set of words; for example, the label "Lunate facet of hamate" is split into the set {"lunate", "facet", "of", "hamate"}. *(ii)* Stop-words are removed, for example,"of" is removed from the set of words (*i.e.*, {"lunate", "facet", "hamate"}). *(iii)* Stemming techniques are applied to each word (*i.e.*, {"lunat", "facet", "hamat"}). *(iv)* Combinations of (sub)set of words serve as keys in Lexl; for example, {"lunat", "facet"}, {"hamat", "lunat"} and so on.[6] *(v)* Entities leading to the same (sub)set of words are associated to the same key in Lexl, for example, the entity $\mathcal{O}_1$:Lunate_facet_of_hamate with numerical identifier 55518 is associated to the Lexl key {"hamat", "lunat"} (see Table 1). Finally, *(vi)* entries in Lexl pointing to entities of only one ontology are not considered (see last two rows of Lexl in Table 1). Note that a single entity label may lead to several entries in Lexl, and each entry in Lexl points to one or many entities.

Each entry in Lexl, after discarding entries pointing to only one ontology, is a source of candidate mappings. For instance the example in Table 1 suggests that there is a (potential) mapping $m = \langle \mathcal{O}_1\text{:Serous\_acinus}, \mathcal{O}_2\text{:Liver\_acinus}, \equiv, c \rangle$ since the entities $\mathcal{O}_1$:Serous_acinus and $\mathcal{O}_2$:Liver_acinus are associated to the same entry in Lexl {*acinus*}. These mappings are not necessarily correct but will link lexically-related entities, that is, those entities sharing at least one word among their labels (*e.g.*, "acinus"). Given a subset of entries of Lexl (*i.e.*, $l \subseteq$ Lexl), the function Mappings$(l) = \mathcal{M}^l$ provides the set of mappings derived from l. We refer to the set of all (potential) mappings suggested by Lexl (*i.e.*, Mappings(Lexl)) as $\mathcal{M}^{\text{Lexl}}$. Note that $\mathcal{M}^{\text{Lexl}}$ represents a manageable subset of the Cartesian product between the entities of the input ontologies.

[5] The indexation module associates unique numerical identifiers to entity URIs.

[6] In order to avoid a combinatorial blow-up, the number of computed subsets of words is limited.

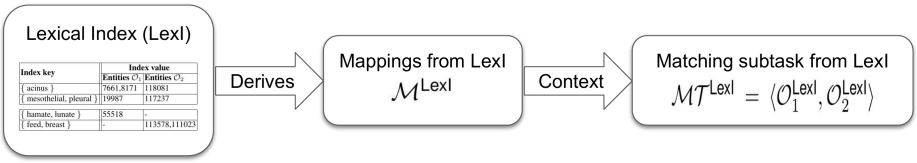


Fig. 1: Pipeline to extract the extended overlapping matching subtasks from LexI.

Most of the state-of-the-art ontology matching systems rely, in one way or another, on lexical similarity measures to either discover or validate candidate mappings [1, 2]. Thus, mappings outside $\mathcal{M}^{\mathsf{LexI}}$ will rarely be discovered by standard matching systems.

Dealing with limited lexical overlapping. The construction of LexI, which is the basis of the methods presented in this section, shares a limitation with state-of-the-art systems when the input ontologies are lexically disparate or do not provide enough lexical information. In this case, the set of mappings $\mathcal{M}^{\mathsf{LexI}}$ may be too small or even empty. As a standard solution, if the ontologies have a small lexical overlapping or are in different languages, LexI can be enriched with general-purpose lexicons (*e.g.*, WordNet or the UMLS lexicon), more specialised background knowledge (*e.g.*, UMLS Metathesaurus) or with translated labels using online translation services like the ones provided by Google, IBM or Microsoft.

3.2 Overlapping estimation

The mappings in $\mathcal{M}^{\mathsf{LexI}}$ can be used to extract an (over)estimation of the overlapping between the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$.

Definition 5 (Extended overlapping matching task). *Let $\mathcal{M}^{LexI}$ be the alignment computed from LexI for $\mathcal{O}_1$ and $\mathcal{O}_2$, and $\mathsf{Context}(\mathcal{M}^{LexI}, \mathcal{O}_1, \mathcal{O}_2) = \langle \mathcal{O}_1^{LexI}, \mathcal{O}_2^{LexI} \rangle$ the context of $\mathcal{M}^{LexI}$. We define $\mathcal{MT}^{LexI} = \langle \mathcal{O}_1^{LexI}, \mathcal{O}_2^{LexI} \rangle$ as the extended overlapping matching task.*

$\mathcal{MT}^{\mathsf{LexI}} = \langle \mathcal{O}_1^{\mathsf{LexI}}, \mathcal{O}_2^{\mathsf{LexI}} \rangle$ can also be seen as the result of reducing or dividing the task $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ where only one matching subtask is given as output (*i.e.*, $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{\mathsf{LexI}}\}$). Figure 1 shows an overview of the pipeline where LexI leads to the division $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{\mathsf{LexI}}\}$.

Hypothesis 1 *If $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ is a matching task, $\mathcal{M}^S$ the mappings computed for $\mathcal{MT}$ by a lexical-based matching system, and $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{LexI}\}$ the reduction of the matching task $\mathcal{MT}$ using the notion of overlapping (over)estimation, then $\mathcal{D}^1_{\mathcal{MT}}$ covers (almost) all the mappings in $\mathcal{M}^S$, that is, $\mathsf{CoverageRatio}(\mathcal{D}^1_{\mathcal{MT}}, \mathcal{M}^S) \approx 1.0$.*

Hypothesis 1 suggests that a matching system will unlikely discover mappings with $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ that cannot be discovered with $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{\mathsf{LexI}}\}$. This intuition is not only supported by the observation that most of the ontology matching systems rely on lexical similarity, but also by the use of the notion of context (see Definition 3 and Definition 4) in the creation of the *extended overlapping matching task*.

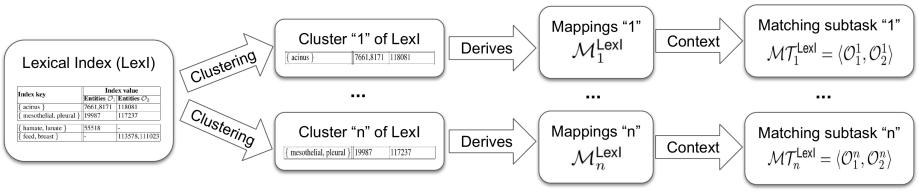


Fig. 2: Pipeline to extract matching subtasks from LexI.

3.3 Creation of matching subtasks from LexI

Considering all entries in LexI (*i.e.*, one cluster) may lead to a very large number of candidate mappings $\mathcal{M}^{\mathsf{LexI}}$ and, as a consequence, to large overlapping modules $\mathcal{O}_1^{\mathsf{LexI}}$ and $\mathcal{O}_2^{\mathsf{LexI}}$. These modules, although smaller than $\mathcal{O}_1$ and $\mathcal{O}_2$, can still be challenging for many ontology matching systems. A solution is to divide the entries in LexI in more than one cluster.

Definition 6 (Matching subtasks from LexI). *Let $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ be a matching task, LexI the lexical index of the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, and $\{c_1, \ldots, c_n\}$ a set of n clusters of entries in LexI. We denote the set of matching subtasks from LexI as $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$ where each cluster c_i leads to the matching subtask $\mathcal{MT}_i^{\mathsf{LexI}} = \langle \mathcal{O}_1^i, \mathcal{O}_2^i \rangle$, such that $\mathsf{Mappings}(c_i) = \mathcal{M}_i^{\mathsf{LexI}}$ is the set of mappings suggested by the LexI entries in c_i and $\mathcal{O}_1^i$ and $\mathcal{O}_2^i$ represent the context of $\mathcal{M}_i^{\mathsf{LexI}}$ w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$.*

Figure 2 shows an overview of the pipeline where LexI is split into n clusters and these clusters lead to n matching subtasks $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$.

Since the matching subtasks in Definition 6 also rely on LexI and the notion of context of the derived mappings from each cluster of entries in LexI, it is expected that the resulting matching subtasks in $\mathcal{D}^n_{\mathcal{MT}}$ will have a coverage similar to $\mathcal{D}^1_{\mathcal{MT}}$.

Hypothesis 2 *If $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ is a matching task and $\mathcal{M}^S$ the mappings computed for $\mathcal{MT}$ by a lexical-based matching system, then, with independence of the clustering strategy of LexI and the number of matching subtasks n, $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$ will cover (almost) all the mappings in $\mathcal{M}^S$ (i.e., $\mathsf{CoverageRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{M}^S) \approx 1.0$).*

Intuitively each cluster of LexI will lead to a smaller set of mappings $\mathcal{M}_i^{\mathsf{LexI}}$ (with respect to $\mathcal{M}^{\mathsf{LexI}}$) and to a smaller matching task $\mathcal{MT}_i^{\mathsf{LexI}}$ (with respect to both $\mathcal{MT}^{\mathsf{LexI}}$ and $\mathcal{MT}$) in terms of search space. Hence $\mathsf{SizeRatio}(\mathcal{MT}_i^{\mathsf{LexI}}, \mathcal{MT})$ will be smaller than 1.0, as mentioned in Section 2.2. Reducing the search space in each matching subtask $\mathcal{MT}_i^{\mathsf{LexI}}$ has the potential of enabling the use of systems that can not cope with the original matching task $\mathcal{MT}$ in a given time-frame or with (limited) computational resources. The aggregation of ratios may be greater than 1.0 and will depend on the clustering strategy.

Hypothesis 3 *Given a matching task $\mathcal{MT}$ and an ontology matching system that fails to complete $\mathcal{MT}$ under a set of given computational constraints, there exists a division*

of the matching task $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$ *for which that system is able to compute an alignment of the individual matching subtasks* $\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}$ *under the same constraints.*

Decreasing the search space may also improve the performance of systems able to cope with $\mathcal{MT}$ in terms f-measure.

Hypothesis 4 *If* $\mathcal{MT} = \langle \mathcal{O}_1, \mathcal{O}_2 \rangle$ *is a matching task,* $\mathcal{M}^S$ *the mappings computed for* $\mathcal{MT}$ *by a state-of-the-art matching system and* F *the f-measure of* $\mathcal{M}^S$ *w.r.t. a given reference alignment* $\mathcal{M}^{RA}$*, then the set of mappings* $\mathcal{M}^S_p = \mathcal{M}^S_1 \cup \ldots \cup \mathcal{M}^S_n$ *computed by the same system over the matching subtasks in* $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$ *leads to an f-measure* F' *such that* $F' \geq F$.

Hypothesis 4 is based on the observation that systems in the OAEI *largebio* track [3] show a better performance when, instead of the original matching task (*e.g.*, whole FMA and NCI ontologies), they are given the overlapping matching task for the reference alignments (as in Definition 4).

3.4 Clustering strategies

We have implemented two clustering strategies which we refer to as: *naive* and *neural embedding*. Both strategies receive as input the index LexI and the number of desired clusters n, and provide as output a set of clusters $\{c_1, \ldots, c_n\}$ from LexI. As in Definition 6, these cluster lead to the set of matching subtasks $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$.

The choice of strategy, according to Hypothesis 2, will not have an impact on the coverage; but it may influence the size of the matching subtasks. Note that, neither of the strategies aims at computing optimal clusters of the entries in LexI, but clusters that can be efficiently computed.

Naive strategy. This strategy implements a very simple algorithm that randomly splits the entries in LexI into a given number of clusters of the same size. The matching tasks resulting from this strategy are expected to have a high overlapping as different entries in LexI leading to similar set of mappings may fall into different clusters. Although the overlapping of matching subtasks will impact the global search space, it is still expected to be smaller than in the original matching task.

Neural embedding strategy. This strategy aims at identifying more accurate clusters, leading to matching tasks with less overlapping, and thus, reducing the global size of the computed division of the matching task $\mathcal{D}^n_{\mathcal{MT}}$. It relies on StarSpace toolkit[7] and its neural embedding model [5], which aims at learning *entity embeddings*. Each entity[8] is described by a finite set of discrete *features* (bag-of-features). The model is trained by assigning a d-dimensional vector to each of the discrete features in the set that we want

[7] StarSpace: `https://github.com/facebookresearch/StarSpace`

[8] Note that in the context of neural embedding models the term entity refers to objects of different kind, *e.g.*, a word, a sentence, a document or even an ontology entity.

to embed directly. Ultimately, the look-up matrix (the matrix of embeddings - latent vectors) is learned by minimizing the loss function in Equation 5.

$$\sum_{(k,v)\in E^+, v^-\in E^-} L^{batch}(sim(k,v), sim(k,v_1^-), \ldots, sim(k,v_k^-)) \quad (5)$$

In this loss function, we need to indicate the generator of positive entry pairs $(k, v) \in E^+$ – in our setting those are *key-value* pairs from LexI– and the generator of negative entries $(k, v^-) \in E^-$ (the so-called *negative examples*) – in our setting, the pairs (k, v^-) that do not appear in LexI. The similarity function sim is task-dependent and should operate on d-dimensional vector representations of the entities, in our case we use the standard Euclidean dot product. The aforementioned neural embedding model corresponds to the `TagSpace` training setting of StarSpace (see [5] for more details). Applied to the lexical index LexI, the neural embedding model would learn vector representations for the individual words in the index keys, and for the individual entity identifiers in the index values. Since an index key is a set of words (see Table 1), we use the *mean vector* representation of the vectors associated to each word (in principle other *aggregated* representation could be applied). Based on these *aggregated* neural embeddings we then perform standard clustering with the K-means algorithm.

Hypothesis 5 *There exists a number of clusters or matching subtasks 'n' for which the clustering strategies can compute* $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{LexI}, \ldots, \mathcal{MT}_n^{LexI}\}$ *for a given matching task* $\mathcal{MT}$ *such that* $\mathsf{SizeRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{MT}) < 1.0$.

Hypothesis 5 suggests that there exists a division $\mathcal{D}^n_{\mathcal{MT}}$ of $\mathcal{MT}$ such that the size (or search space) of $\mathcal{D}^n_{\mathcal{MT}}$ is smaller than $\mathcal{MT}$, and $\mathcal{D}^n_{\mathcal{MT}}$ can be computed by the proposed naive and neural embedding strategies.

4 Evaluation

In this section we aim at providing empirical evidence to support the Hypothesis 1-5 introduced in Section 3. We rely on the datasets of the Ontology Alignment Evaluation Initiative (OAEI) [3], more specifically, on the matching tasks provided in the *anatomy*, *largebio* and *phenotype* tracks. Table 2 provides an overview of these OAEI tasks and the related ontologies.

The methods have been implemented in Java[9] and Python[10] (neural embedding strategy), tested on a Ubuntu Laptop with an Intel Core i7-4600U CPU@2.10GHz (4 cores) and allocating up to 15 Gb of RAM. Datasets, evaluation results, logs and other supporting resources are available in the *Zenodo* repository [10].

We have performed the following experiments, which we describe in detail in the following sections:

- We have computed the extended overlapping matching task (*i.e.*, $\mathcal{D}^1_{\mathcal{MT}}$) for each of the matching tasks as in Definition 5 and calculated the coverage with respect to the available reference alignments (Section 4.1).

[9] Java codes: `https://github.com/ernestojimenezruiz/logmap-matcher`
[10] Python codes: `https://github.com/plumdeq/neuro-onto-part`

Table 2: Matching tasks. AMA: Adult Mouse Anatomy. DOID: Human Disease Ontology. FMA: Foundational Model of Anatomy. HPO: Human Phenotype Ontology. MP: Mammalian Phenotype. NCI: National Cancer Institute. NCIA: Anatomy fragment of NCI. ORDO: Orphanet Rare Disease Ontology. SNOMED: Systematized Nomenclature of Medicine – Clinical Terms. Phenotype ontologies downloaded from BioPortal.

OAEI track	**Source of $\mathcal{M}^{RA}$**	**Task**	**Ontology**	**Version**	**Size (classes)**
Anatomy	Manually created	AMA-NCIA	AMA	v.2007	2,744
			NCIA	v.2007	3,304
Largebio	UMLS-Metathesaurus	FMA-NCI	FMA	v.2.0	78,989
		FMA-SNOMED	NCI	v.08.05d	66,724
		SNOMED-NCI	SNOMED	v.2009	306,591
Phenotype	Consensus alignment (vote=2) [9]	HPO-MP	HPO	v.2016-BP	11,786
			MP	v.2016-BP	11,721
		DOID-ORDO	DOID	v.2016-BP	9,248
			ORDO	v.2016-BP	12,936

Table 3: Coverage results for $\mathcal{D}^1_{\mathcal{MT}}$

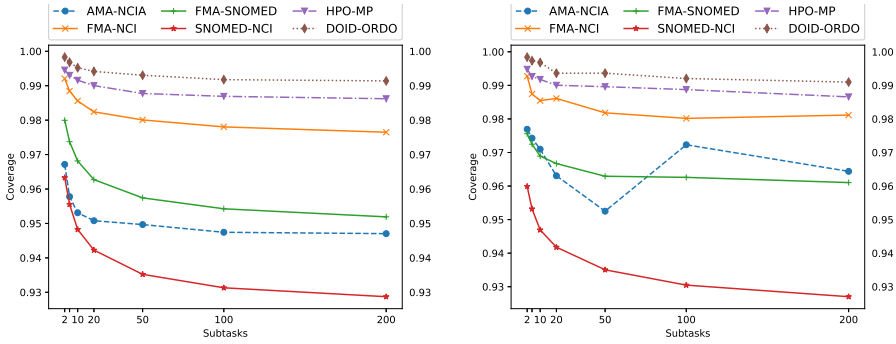
Task	**|Lexl|**	$\mathcal{D}^1_{\mathcal{MT}}$ **statistics**				
		$\lvert\mathcal{O}_1^1\rvert$	$\lvert\mathcal{O}_2^1\rvert$	**SizeRatio**	**CoverageRatio**	**time (s)**
AMA-NCIA	4,048	2,518	2,841	0.784	0.982	0.55
FMA-NCI	11,507	33,744	35,409	0.226	0.994	10.3
FMA-SNOMED	29,677	55,469	119,488	0.273	0.982	28.8
SNOMED-NCI	45,940	190,911	56,076	0.521	0.968	28.2
HPO-MP	10,514	8,165	10,041	0.589	0.995	1.93
DOID-ORDO	13,375	7,166	10,741	0.637	0.999	2.81

– We have applied the *naive* and *neural embedding*[11] strategies to compute divisions $\mathcal{D}^n_{\mathcal{MT}}$ of the matching tasks and evaluated their adequacy with respect to coverage and size (Section 4.2).
– We have evaluated the performance of a selection of OAEI matching systems over the computed matching subtasks and compared with their original results (if any) in the OAEI campaigns (Section 4.3).

4.1 Coverage of the extended overlapping matching task

We have evaluated the coverage of $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{\mathsf{Lexl}}\}$ computed for each of the matching tasks in Table 2 with respect to the available reference alignments. Table 3 summarizes the obtained results. The second column of the table gives the number of entries in Lexl, while the last column represents the time to compute Lexl, the derived mappings $\mathcal{M}^{\mathsf{Lexl}}$ and the context of $\mathcal{M}^{\mathsf{Lexl}}$ (*i.e.*, the overlapping matching task). The obtained coverage (ratio) values range from 0.967 to 0.999, which strongly supports our intuitions behind Hypothesis 1. Furthermore, since we have calculated the

[11] Please refer to [10] for information about the used StarSpace input parameters.



(a) Naive strategy

(b) Neural embedding strategy

Fig. 3: CoverageRatio of $\mathcal{D}^n_{\mathcal{MT}}$ with respect to the number of matching subtasks n.

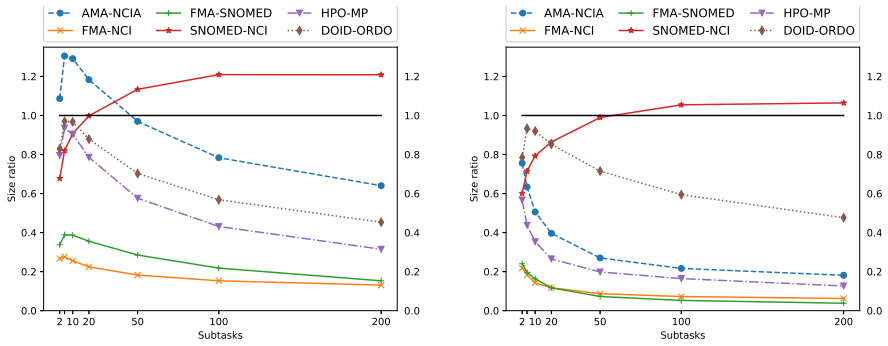
coverage with respect to the reference alignments instead of system mappings (*i.e.*, $\mathsf{CoverageRatio}(\mathcal{D}^1_{\mathcal{MT}}, \mathcal{M}^{RA})$), the results also suggest that the information loss with respect to system-generated alignments will be minimal. At the same time the size (ratio) of the matching tasks is significantly reduced for the largest matching tasks. For example, for the FMA-NCI case, the resulting task size has been reduced to 27.3% of the original task size. The achieved high coverage in combination with the reduction of the search space and the small computation times provide empirical evidence of the suitability of LExI to reduce the alignment task at hand.

4.2 Adequacy of the clustering strategies

We have evaluated the adequacy of the clustering strategies in terms of coverage (as in Equation 4) and size (as in Equation 3) of the resulting division $\mathcal{D}^n_{\mathcal{MT}}$ of the matching task. We have compared the two strategies for different number of clusters or resulting matching subtasks $n \in \{2, 5, 10, 20, 50, 100, 200\}$. For the naive strategy, as a random split of LExI is performed, we run 10 experiments for each of the values of n to evaluate the effect of different random selections. The variations in the size of the obtained matching tasks was negligible.[12] The results reported for the naive strategy represent the average of the 10 experiments.

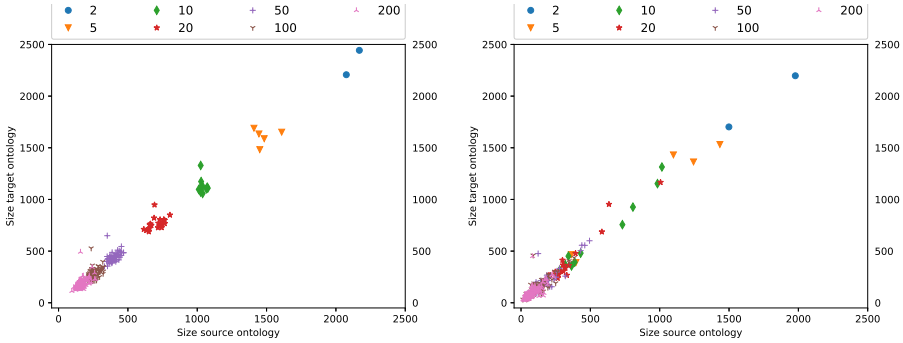
Coverage ratio. Figure 3 shows the coverage of the different divisions $\mathcal{D}^n_{\mathcal{MT}}$ of the matching task for the naive (left) and neural embedding (right) strategies. As in the case of $\mathcal{D}^1_{\mathcal{MT}} = \{\mathcal{MT}^{Lex}\}$ the coverage ratio is very good, being 0.927 in the worst case ($n = 200$ in SNOMED-NCI) and 0.99 in the best case ($n = 2$ in FMA-NCI). This means that, in the worst case, almost 93% of the available reference mappings are *covered* by the matching subtasks in $\mathcal{D}^n_{\mathcal{MT}}$. The differences in terms of coverage between the naive and neural embedding strategies are minimal, with the neural embedding strategy providing slightly better results on average. These results reinforce Hypothesis 2 as the coverage with respect to system-generated mappings is expected to be even better.

[12] Details about matching task sizes and standard deviations can be found in [10].



(a) Naive strategy (b) Neural embedding strategy

Fig. 4: SizeRatio of $\mathcal{D}^n_{\mathcal{MT}}$ with respect to the number of matching subtasks n.

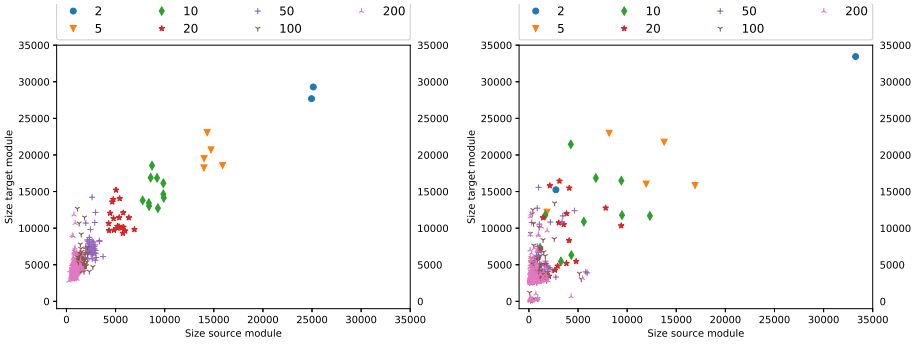


(a) Naive strategy (b) Neural embedding strategy

Fig. 5: Source and target module sizes in the computed subtasks for AMA-NCIA.

Size ratio. The results in terms of the size (*i.e.*, search space) of the selected divisions $\mathcal{D}^n_{\mathcal{MT}}$ are presented in Figure 4 for the naive (left) and neural embedding (right) strategies. The results with the neural embedding strategy are extremely positive, while the results of the naive strategy, although slightly worse as expected, are surprisingly very competitive. Both strategies improve the search space with respect to the original $\mathcal{MT}$ for all cases with the exception of the naive strategy in the AMA-NCIA case with $n < 50$, and the SNOMED-NCI case with $n > 20$, which validates Hypothesis 5. SNOMED-NCI confirms to be the hardest case in the *largebio* track. Here the size ratio increases with the number of matching subtasks n and gets stable with $n > 100$.

Size of the source and target modules. The scatter plots in Figures 5 and 6 visualize the size of the source modules against the size of the target modules for the matching tasks in each division $\mathcal{D}^n_{\mathcal{MT}}$. For instance, the (orange) triangles represent points $\left(|Sig(\mathcal{O}^i_1)|, |Sig(\mathcal{O}^i_2)|\right)$ being $\mathcal{O}^i_1$ and $\mathcal{O}^i_2$ the source and target modules (with i=1,...,5) in the matching subtasks of $\mathcal{D}^5_{\mathcal{MT}}$. Figure 5 shows the plots for the AMA-NCIA case while Figure 6 for the FMA-NCI case, using the naive (left) and neural em-



(a) Naive strategy (b) Neural embedding strategy

Fig. 6: Source and target module sizes in the computed subtasks for FMA-NCI.

bedding (right) strategies. The naive strategy leads to rather balanced an similar tasks (note differentiated cloud of points) for each division $\mathcal{D}^n_{\mathcal{MT}}$ for both cases. The neural embedding strategy has more variability in the size of the tasks within a given division $\mathcal{D}^n_{\mathcal{MT}}$. In the FMA-NCI case the tasks generated by the neural embedding strategy are also less balanced and the target module tends to be larger than the source module. Nonetheless, on average, the (aggregated) size of the matching tasks in the neural embedding strategy are significantly reduced as shown in Figure 4.

Computation times. The time to compute the divisions of the matching task is tied to the number of locality modules to extract, which can be computed in polynomial time relative to the size of the input ontology [6]. The creation of Lexl does not add an important overhead, while the training of the neural embedding model in the advance strategy ranges from 21s in AMA-NCI to 224s in SNOMED-NCI. Overall, for example, the required time to compute the division with 50 matching subtasks ranges from 2s in AMA-NCIA to 413s in SNOMED-NCI with the naive strategy, and from 24s (AMA-NCIA) to 647s (SNOMED-NCI) with the neural embedding strategy. Complete list of relevant times can be obtained from [10].

4.3 Evaluation of OAEI systems

In this section we support Hypothesis 3 by showing that the division of the alignment task enables systems that, given some computational constraints, were unable to complete an OAEI task. We have selected the following five systems from the latest OAEI campaigns:[13] MAMBA [11], GMap [12], FCA-Map [13], KEPLER [14], and POMap [15]. MAMBA and GMap failed to complete the OAEI 2015 Anatomy track [16] with 8Gb of allocated memory, while FCA-Map, KEPLER and POMap could not complete the largest tasks in the *largebio* track within a 12 hours time-frame (with 16Gb of allocated memory) [3, 17].[14] Note that GMap and MAMBA were also tested

[13] Other systems were also considered but they threw an exception during execution.

[14] In a preliminary evaluation round a 4 hours time-frame was given, which was later extended.

Table 4: Evaluation of systems that failed to complete OAEI tasks in the 2015-2017 campaigns. (*) GMap was tested allocating 8Gb of memory. Time reported in hours (h).

Tool	Task	Year	Matching subtasks	Naive strategy				Neural embedding strategy			
				P	R	F	t (h)	P	R	F	t (h)
GMap (*)	Anatomy	2015	5	0.87	0.81	0.84	1.3	0.88	0.82	0.85	0.7
			10	0.85	0.81	0.83	1.7	0.86	0.82	0.84	0.8
MAMBA	Anatomy	2015	20	0.88	0.63	0.73	2.3	0.89	0.62	0.73	1.0
			50	0.88	0.62	0.73	2.4	0.89	0.62	0.73	1.0
FCA-Map	FMA-NCI	2016	20	0.56	0.90	0.72	4.4	0.62	0.90	0.73	3.1
			50	0.58	0.90	0.70	4.1	0.60	0.90	0.72	3.0
KEPLER	FMA-NCI	2017	20	0.45	0.82	0.58	8.9	0.48	0.80	0.60	4.3
			50	0.42	0.83	0.56	6.9	0.46	0.80	0.59	3.8
POMap	FMA-NCI	2017	20	0.54	0.83	0.66	11.9	0.56	0.79	0.66	5.7
			50	0.55	0.83	0.66	8.8	0.57	0.79	0.66	4.1

in the OAEI 2015 with 14Gb of memory. This new setting allowed GMap to complete the task [16].

Table 4 shows the obtained results in terms of computation times, precision, recall and f-measure over different divisions $\mathcal{D}^n_{\mathcal{MT}}$ computed by the naive and neural embedding strategies. For example, MAMBA was run over divisions with 20 and 50 matching subtasks (*i.e.*, $n \in \{20, 50\}$). Note that GMap was tested allocating only 8Gb of memory as with this constraint it could not complete the task in the OAEI 2015. The results can be summarized as follows:

i) The computation times are encouraging since the (independent) matching tasks have been run sequentially without any type of parallelization.

ii) Times also include loading the ontologies from disk for each matching task. This step could be avoided if subtasks are directly provided by the presented framework.

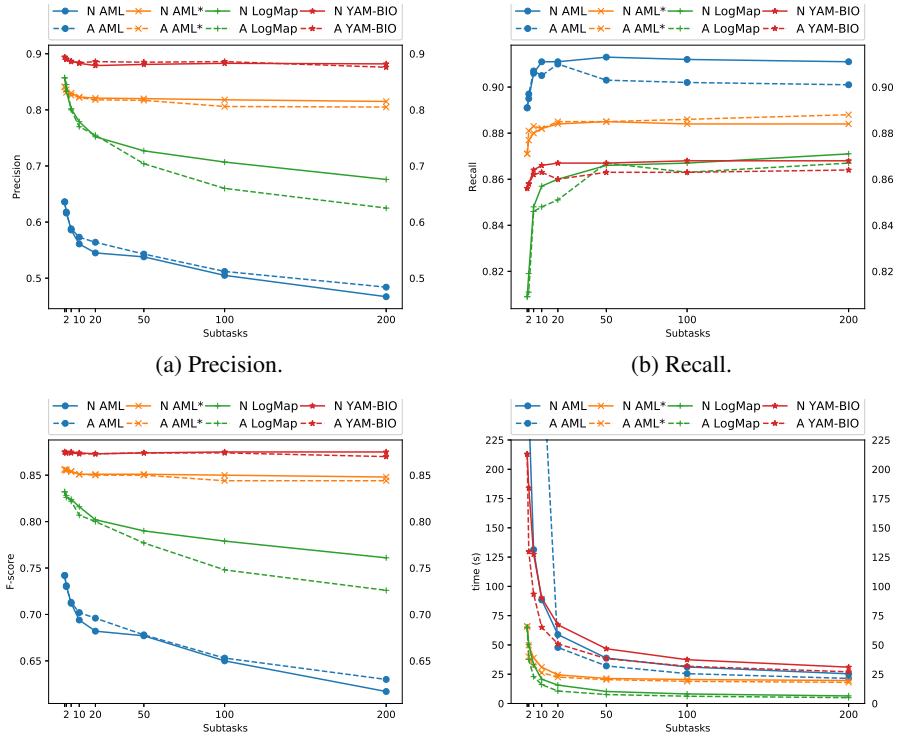
iii) We did not perform an exhaustive analysis, but memory consumption was lower than 8Gb in all tests; thus, systems like GMap could run under limited resources.

iv) The increase of number of matching subtasks is beneficial for FCA-Map, KEPLER and POMap in terms of computation times. However, this is not the case for MAMBA and GMap.

v) The division generated by the neural embedding strategy leads to smaller computation times than the naive strategy counterparts, as expected from Figure 4.

vi) The f-measure is slightly reduced as the size of n increases. This result does not support our intuitions behind Hypothesis 4.

Comparison with OAEI results. There are *baseline* results in the OAEI for the selected systems [3, 16, 17], with the exception of MAMBA where the results are novel for the *anatomy* track. As mentioned before, GMap, if 14Gb were allocated, was able to complete the *anatomy* task and obtained an f-measure of 0.861. KEPLER, POMap and FCA-Map completed the OAEI task involving small fragments of FMA-NCI (*i.e.*, the *overlapping matching* task as in Definition 4) with an f-measure of 0.891, 0.861 and 0.935, respectively. The f-measure using the divisions of the matching task is slightly



(a) Precision.

(b) Recall.

(c) F-measure.

(d) Time (s) per matching task.

Fig. 7: Performance of top-systems in FMA-NCI task for the divisions $\mathcal{D}^n_{\mathcal{MT}}$. Original OAEI 2017 results: YAM-BIO *(P: 0.82, R: 0.89, F: 0.85, t: 279s)*, AML *(P: 0.84, R: 0.87, F: 0.86, t: 77s)*, LogMap *(P: 0.86, R: 0.81, F: 0.83, t: 92s)*.

lower for GMap, which once more, does not support our Hypothesis 4. The results are much lower for the cases of KEPLER, POMap and FCA-Map, but they cannot be fully comparable as systems typically reduce their performance when dealing with the whole *largebio* ontologies [3]. The authors of FCA-Map have also recently reported results for an improved version of FCA-Map [18]. They completed the FMA-NCI task in near 7 hours, with a precision of 0.41, a recall of 0.87 and a f-measure of 0.56. The results obtained with $\mathcal{D}^{20}_{\mathcal{MT}}$ and $\mathcal{D}^{50}_{\mathcal{MT}}$ are thus very positive, since both strategies lead to much better numbers in terms of computation times and f-measure.

Performance of top OAEI systems. We have also evaluated the top systems in the OAEI 2017 *largebio* track [3] (LogMap, AML and YAM-BIO) to *i)* confirm the dismissal of Hypothesis 4, and *ii)* evaluate the effect of the divisions of a matching task in the performance of a system. Figure 7 shows the results for the divisions $\mathcal{D}^n_{\mathcal{MT}}$ of FMA-NCI with $n \in \{2, 5, 10, 20, 50, 100, 200\}$. Solid lines represent the results for the divisions computed with the naive strategy while the neural embedding strategy results are represented with dashed lines. For example, in the figure legends, "N AML" stands for the

results of AML with the divisions using the (N)aive strategy while "A AML" stands for the results of AML with the (A)dvanced (*i.e.*, neural embedding) strategy divisions. The results for the naive and neural embedding strategies are very similar, with the exception of LogMap, for which results are slightly different for $n > 50$. YAM-BIO maintains almost constant values for precision, recall and f-measure. The f-measure of YAM-BIO is improved with respect to the original OAEI results. The results for AML and LogMap are less positive as the number of matching subtasks (*i.e.*, n) increases. Recall increases with n but remains relatively constant for $n > 50$, however precision is highly impacted by n. These results weaken the validity of Hypothesis 4. The decrease in LogMap's precision may be explained by the fact that LogMap limits the cases of many to many correspondences and, when the alignment task is divided, that filter is probably not triggered leading to an increase of the false positives. Regarding AML performance, we contacted the developers of AML to get a better insight about the results. For the type and size of the computed matching tasks AML automatically applies a less conservative matching pipeline which leads to an increase of the recall, but also to a notable decrease in precision. We also evaluated AML forcing a (conservative) pipeline (referred to as AML* in Figure 7). AML* obtains the expected results, which are very similar to the original OAEI results for all the divisions $\mathcal{D}^n_{\mathcal{MT}}$. The times reported in Figure 7d represent averages per matching task. The times for AML were also higher than expected. As expected the necessary time to complete a task is reduced with n. The total required time, however, is increased for all three evaluated systems. For example LogMap requires around 100s to complete the two matching tasks in $\mathcal{D}^2_{\mathcal{MT}}$, while it needs more than 800s to complete the 100 matching tasks in $\mathcal{D}^{100}_{\mathcal{MT}}$. This is explained by the fact that these systems implement efficient indexing and matching techniques and a large portion of the execution time is devoted to loading, processing and initialization of the matching task. Nevertheless, if several tasks are run in parallel, the wall-clock times can be reduced significantly. For example, the HOBBIT platform adopted for the OAEI 2017.5 and 2018 evaluation campaigns includes 16 hardware cores devoted for the system evaluation [19]. Thus, total wall-clock times could potentially be split by 16.

5 Related work

The use of partitioning and modularization techniques have been extensively used within the Semantic Web to improve the efficiency when solving the task at hand (*e.g.*, ontology visualization [20,21], ontology reuse [22], ontology debugging [23], ontology classification [24]). Partitioning has also been widely used to reduce the complexity of the ontology alignment task. In the literature there are two major categories of partitioning techniques, namely: *independent* and *dependent*. Independent techniques typically use only the structure of the ontologies and are not concerned about the ontology alignment task when performing the partitioning. Whereas dependent partitioning methods rely on both the structure of the ontology and the ontology alignment task at hand. Our approach, although we do not compute (non-overlapping) partitions of the ontologies, can be considered a type of dependent technique.

Prominent examples of ontology alignment systems including partitioning techniques are Falcon-AO [25], COMA++ [26] and TaxoMap [27]. Falcon-AO and COMA++

perform independent partitioning where the clusters of the source and target ontologies are independently extracted. Then pairs of similar clusters (*i.e.*, matching subtasks) are aligned using standard techniques. TaxoMap [27] implements a dependent technique where the partitioning is combined with the matching process. TaxoMap proposes two methods, namely: PAP (partition, anchor, partition) and APP (anchor, partition, partition). The main difference of these methods is the order of extraction of (preliminary) anchors to discover pairs of partitions to be matched (*i.e.*, matching subtasks).

Algergawy et al. [28] have recently presented SeeCOnt, which proposes a seeding-based clustering technique to discovers independent clusters in the input ontologies. Their approach has been evaluated with the Falcon-AO system by replacing its native PBM (Partition-based Block Matching) module [29].

The above approaches, although they present interesting results, did not provide any guarantees about the coverage (as in Definition 2) of the discovered partitions or divisions. In [30] we performed a preliminary study with the PBM method of Falcon-OA, and the PAP and APP methods of TaxoMap. The results in terms of coverage with the *largebio* tasks were very low, which directly affected the results of the evaluated systems. These rather negative results encouraged us to work on the approach presented in this paper.

Our dependent approach, unlike traditional partitioning methods, computes overlapping self-contained modules (*i.e.*, locality modules). Locality modules guarantee the extraction of all semantically related entities for a given signature, which enhances the coverage results and enables the inclusion of the relevant information required by an alignment system. It is worth mentioning that the need of self-contained and covering modules, although not thoroughly studied, was also highlighted in a preliminary work by Paulheim [31].

6 Conclusions and future work

We have developed a novel framework to split the ontology alignment task into several matching subtasks based on a lexical index and locality modules. These independent matching subtasks can be potentially run in parallel in evaluation platforms like the HOBBIT [19]. We have also presented two clustering strategies of the lexical index. One of them relies on a simple splitting method, while the other relies on a fast (log-linear) neural embedding model. We have performed a comprehensive evaluation of both strategies which suggests that the obtained divisions are suitable in practice in terms of both coverage and size. The naive strategy leads to well-balanced set of tasks, while the overall reduction of the search space with the neural embedding strategy was very positive. The division of the matching task also allowed us to obtain results for five systems which failed to complete these OAEI matching tasks in the past.

The results in terms of f-measure were not as good as expected for some of the systems. The f-measure also tended to decrease as the number of matching subtasks increased. These results, although not supporting our original intuitions, do not undermine the value of the proposed framework as we cannot control the internal behaviour of the ontology alignment system. Computed matching subtasks for a given division $\mathcal{D}^n_{\mathcal{MT}}$ may have a high overlapping, especially when relying on the naive strategy. That

is, the same mapping can be proposed from different matching subtasks. This can enhance the discovery of true positives, but may also bring in a number of false positives, as for the case of LogMap in the reported evaluation. The adoption of the presented framework within the pipeline of an ontology alignment system may also lead to improved results, as for the case of YAM-BIO and AML with a conservative pipeline. It is worth mentioning that the OAEI system SANOM (v.2018) is already integrating the strategies presented in this paper within its matching workflow.

Both the naive and the neural embedding strategies require the size of the number of matching subtasks or clusters as input. The (required) matching subtasks may be known before hand if, for example, the matching tasks are to be run in parallel in a number of available CPUs. For the cases where the resources are limited or where a matching system is known to cope with small ontologies, we plan to design an algorithm to estimate the number of clusters so that the size of the matching subtasks in the computed divisions is appropriate to the system and resource constraints. As immediate future we also plan to study different notions of *context* of an alignment (*e.g.*, the tailored modules proposed in [32]). Locality-based modules, although they have led to very good results, can still be large in some cases.

Acknowledgements. EJR was funded by the Centre for Scalable Data Access (SIRIUS), the RCN project BigMed, and The Alan Turing project AIDA. We would also like to thank the anonymous reviewers that helped us to improve this contribution.

References

1. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1) (2013) 158–176
2. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
3. Achichi, M., et al.: Results of the Ontology Alignment Evaluation Initiative 2017. In: International Workshop on Ontology Matching. (2017) 61–113
4. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: European Conf. Artif. Intell. (ECAI). (2012)
5. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: StarSpace: Embed All The Things! arXiv (2017)
6. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. **31** (2008) 273–318
7. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. Web Semantics **6**(4) (2008) 309–322
8. Diallo, G.: An effective method of large scale ontology matching. J. Biomedical Semantics **5** (2014) 44
9. Harrow, I., Jiménez-Ruiz, E., et al.: Matching disease and phenotype ontologies in the ontology alignment evaluation initiative. J. Biomedical Semantics **8**(1) (2017)
10. Jiménez-Ruiz, E., Agibetov, A., Samwald, M., Cross, V.: Supporting resources: additional results, logs and datasets (2018) : `https://doi.org/10.5281/zenodo.1214149`.
11. Meilicke, C.: MAMBA - results for the OAEI 2015. In: 10th International Workshop on Ontology Matching. (2015) 181–184
12. Li, W., Sun, Q.: GMap: results for OAEI 2015. In: 10th International Workshop on Ontology Matching. (2015) 150–157

13. Zhao, M., Zhang, S.: FCA-Map results for OAEI 2016. In: 11th International Workshop on Ontology Matching. (2016)
14. Kachroudi, M., Diallo, G., Yahia, S.B.: OAEI 2017 results of KEPLER. In: 12th International Workshop on Ontology Matching. (2017) 138–145
15. Laadhar, A., Ghozzi, F., Megdiche, I., Ravat, F., Teste, O., Gargouri, F.: POMap results for OAEI 2017. In: 12th International Workshop on Ontology Matching. (2017) 171–177
16. Cheatham, M., et al.: Results of the Ontology Alignment Evaluation Initiative 2015. In: International Workshop on Ontology Matching. (2015) 60–115
17. Achichi, M., et al.: Results of the Ontology Alignment Evaluation Initiative 2016. In: International Workshop on Ontology Matching. (2016) 73–129
18. Zhao, M., Zhang, S., Li, W., Chen, G.: Matching biomedical ontologies based on formal concept analysis. J. Biomedical Semantics **9**(1) (2018) 11:1–11:27
19. Röder, M., Ngomo, A.N., Strohbach, M.: Deliverable 2.1: Detailed Architecture of the HOBBIT platform. EU Project: Holistic Benchmarking of Big Linked Data (2016)
20. Stuckenschmidt, H., Schlicht, A.: Structure-based partitioning of large ontologies. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. (2009)
21. Agibetov, A., Patanè, G., Spagnuolo, M.: Grontocrawler: Graph-Based Ontology Exploration. In: Eurographics Italian Chapter Conference. (2015) 67–76
22. Jiménez-Ruiz, E., Grau, B.C., Sattler, U., Schneider, T., Berlanga, R.: Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support. In: ESWC. (2008)
23. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A modularization-based approach to finding all justifications for OWL DL entailments. In: Asian Semantic Web Conference. (2008) 1–15
24. Armas Romero, A., Cuenca Grau, B., Horrocks, I.: MORe: Modular Combination of OWL Reasoners for Ontology Classification. In: International Semantic Web Conference. (2012)
25. Hu, W., Qu, Y., Cheng, G.: Matching large ontologies: A divide-and-conquer approach. Data Knowl. Eng. **67** (2008) 140–160
26. Algergawy, A., Massmann, S., Rahm, E.: A clustering-based approach for large-scale ontology matching. In: ADBIS. (2011) 415–428
27. Hamdi, F., Safar, B., Reynaud, C., Zargayouna, H.: Alignment-based partitioning of large-scale ontologies. In: Advances in Knowledge Discovery and Management. (2009) 251–269
28. Algergawy, A., Babalou, S., Kargar, M.J., Davarpanah, S.H.: Seecont: A new seeding-based clustering approach for ontology matching. In: ADBIS. (2015)
29. Hu, W., Zhao, Y., Qu, Y.: Partition-Based Block Matching of Large Class Hierarchies. In: Asian Semantic Web Conference. (2006) 72–83
30. Pereira, S., Cross, V., Jiménez-Ruiz, E.: On partitioning for ontology alignment. In: International Semantic Web Conference (Posters & Demonstrations). (2017)
31. Paulheim, H.: On Applying Matching Tools to Large-scale Ontologies. In: OM. (2008)
32. Armas Romero, A., Kaminski, M., Cuenca Grau, B., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. J. Artif. Intell. Res. **55** (2016)