We Divide, You Conquer:
From Large-scale Ontology Alignment to Manageable Subtasks 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 subtasks. 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.

1 Introduction

Large-scale ontology matching tasks 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 [2]. 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 [3] (best results in *conference* track, but failed to complete the *anatomy* task); *(ii)* CODI version 2011.5 [4] (best results in *anatomy* but could not cope with the *largebio* track); *(iii)* MAMBA version 2015 [5] (top system in the *conference* track but could not complete the *anatomy* track); *(iv)* FCA-Map version 2016 [6] (completed both *anatomy* and *phenotype* tasks but did not complete the largest *largebio* tasks); and *(v)* POMap version 2017 [7] (one of the top systems in *anatomy* but could not finish the largest *largebio* tasks).

In this paper we propose a novel method to effectively divide the matching task into several (independent) smaller subtasks. This method relies on an efficient lexical index (as in LogMap [8]), a neural embedding model [9] and locality modules [10]. 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.

2 Preliminaries

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

[⋆] An extended version of this paper is available in arXiv.org [1].

2.1 Basic definitions

A *mapping* (also called *match* or *correspondence*) between entities[1] of two ontologies[2] $\mathcal{O}_1$ and $\mathcal{O}_2$ 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 (implicit) 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 the ontology $\mathcal{O}$.

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 $\mathcal{M}^S$.

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:

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

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:

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

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

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).

The *coverage* of the matching subtask aims at providing guarantees about the preservation of the (potential) outcomes of the original matching task (*i.e.*, information loss). 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.

[1] We refer to (OWL 2) classes, data and object properties and named individuals as entities.

[2] We assume ontologies are expressed in OWL 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 the result of dividing a matching task $\mathcal{MT}$ be $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1, \ldots, \mathcal{MT}_n\}$ 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 is often 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 [10], 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 context, *i.e.*, sets of *semantically related* entities [10], 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 [10]. 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 (explicit) overlapping of these ontologies with respect to the aforesaid alignment. Intuitively, the ontologies in the context of an alignment 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 original matching task $\mathcal{MT}$ and the reduced task $\mathcal{MT}^{\mathcal{M}}_{\mathcal{O}_1\text{-}\mathcal{O}_2}$. For example, in the small OAEI *largebio* tasks [2] 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.*, $\mathcal{MT}^{RA}_{\text{fma-nci}} = \mathsf{Context}(\mathcal{M}^{RA}_{\text{fma-nci}}, \mathcal{O}_{\text{FMA}}, \mathcal{O}_{\text{NCI}}) = \langle \mathcal{O}'_{\text{FMA}}, \mathcal{O}'_{\text{NCI}} \rangle$).

Table 1: Inverted lexical index LexI (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 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 [8].

3.1 The lexical index LexI

LexI 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[3] such that the set of words of the key appears in (one of) the entity labels. Table 1 shows a few example entries of LexI for two input ontologies.

LexI 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 LexI; for example, {"lunat", "facet"}, {"hamat", "lunat"} and so on.[4] *(v)* Entities leading to the same (sub)set of words are associated to the same key in LexI, for example, the entity $\mathcal{O}_1$:$\mathsf{Lunate\_facet\_of\_hamate}$ with numerical identifier 55518 is associated to the LexI key {"hamat", "lunat"} (see Table 1). Finally, *(vi)* entries in LexI pointing to entities of only one ontology are not considered (see last two rows of LexI in Table 1). Note that a single entity label may lead to several entries in LexI, and each entry in LexI points to one or many entities.

Each entry in LexI, 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 $m = \langle \mathcal{O}_1{:}\mathsf{Serous\_acinus}, \mathcal{O}_2{:}\mathsf{Liver\_acinus}, \equiv, c\rangle$ since the entities $\mathcal{O}_1$:$\mathsf{Serous\_acinus}$ and $\mathcal{O}_2$:$\mathsf{Liver\_acinus}$ are associated to the same entry in LexI {*acinus*}. These mappings are not necessarily correct but 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 LexI (*i.e.*, $l \subseteq \mathsf{LexI}$), the function $\mathsf{Mappings}(l) = \mathcal{M}^l$ provides the set of mappings derived from l. We refer to the set of all (potential) mappings suggested by LexI (*i.e.*, $\mathsf{Mappings}(\mathsf{LexI})$) as $\mathcal{M}^{\mathsf{LexI}}$. Note that $\mathcal{M}^{\mathsf{LexI}}$ represents a manageable subset of the Cartesian product between the entities of the input ontologies.

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

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

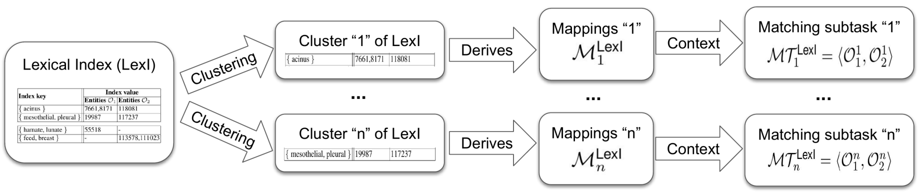

Lexical Index (LexI)
Index key
Index value
Entities $\mathcal{O}_1$
Entities $\mathcal{O}_2$
{ acinus } 7661,8171 118081
{ mesothelial, pleural } 19987 117237
{ hamate, lunate } 55518 -
{ feed, breast } - 113578,111023
Clustering
Clustering
Cluster "1" of LexI
{ acinus } 7661,8171 118081
Cluster "n" of LexI
{ mesothelial, pleural } 19987 117237
Derives
Derives
Mappings "1"
$\mathcal{M}_1^{\mathsf{LexI}}$
Mappings "n"
$\mathcal{M}_n^{\mathsf{LexI}}$
Context
Context
Matching subtask "1"
$\mathcal{MT}_1^{\mathsf{LexI}} = \langle \mathcal{O}_1^1, \mathcal{O}_2^1 \rangle$
Matching subtask "n"
$\mathcal{MT}_n^{\mathsf{LexI}} = \langle \mathcal{O}_1^n, \mathcal{O}_2^n \rangle$
...

Fig. 1: Pipeline to extract 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 [11]. Thus, mappings outside $\mathcal{M}^{\mathsf{LexI}}$ will rarely be discovered by standard matching systems.

3.2 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 $\mathcal{M}^{\mathsf{LexI}}$. The context of $\mathcal{M}^{\mathsf{LexI}}$ leads to (two) large overlapping modules $\mathcal{O}_1^{\mathsf{LexI}}$ and $\mathcal{O}_2^{\mathsf{LexI}}$ that, although smaller than the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, may still be challenging for many ontology matching systems. A solution is to divide the entries in LexI in more than one cluster.

Definition 5 (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 $\{c_1, \ldots, c_n\}$ 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 1 shows an overview of the pipeline where LexI is split into n clusters and these clusters lead to n matching subtasks $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$.[5]

Hypothesis 1 *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 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.,* $\mathsf{CoverageRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{M}^S) \approx 1.0$*).*

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 $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$. This intuition is supported not only 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 matching subtasks.

Intuitively each cluster of LexI leads 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 $\mathsf{SizeRatio}(\mathcal{MT}_i^{\mathsf{LexI}}, \mathcal{MT})$ is expected to 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.

[5] The number of clusters n is a parameter given as input. See Section 6 for a discussion of possibles ways of automatically obtaining n.

Hypothesis 2 *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.*

3.3 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 5, these clusters lead to the matching subtasks in $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$.

The choice of strategy, according to Hypothesis 1, 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[6] and its neural embedding model [9], which aims at learning *entity embeddings*. Each entity[7] 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 to embed directly. 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. Based on these *aggregated* neural embeddings we then perform standard clustering with the K-means algorithm.

Hypothesis 3 *There exists a number of clusters or matching subtasks 'n' for which the clustering strategies can compute $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}_1^{\mathsf{LexI}}, \ldots, \mathcal{MT}_n^{\mathsf{LexI}}\}$ for a given matching task $\mathcal{MT}$ such that* $\mathsf{SizeRatio}(\mathcal{D}^n_{\mathcal{MT}}, \mathcal{MT}) < 1.0$.

Hypothesis 3 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.

Table 2: OAEI matching tasks. 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) [12]	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

4 Evaluation

In this section we support Hypothesis 1-3 (Section 3). We rely on the datasets of the Ontology Alignment Evaluation Initiative (OAEI) [2], more specifically, on the matching tasks provided in the *anatomy*, *largebio* and *phenotype* tracks (see Table 2).

The methods have been implemented in Java[8] and Python[9] and were tested on a Ubuntu Laptop with an Intel Core i7-4600U CPU@2.10GHz (4 cores). Up to 15 Gb of RAM was allocated. The next sections present the performed experiments.[10]

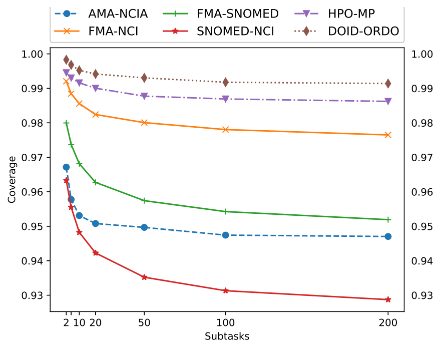
4.1 Adequacy of the clustering strategies

We have evaluated the adequacy of the clustering strategies to compute divisions $\mathcal{D}^n_{\mathcal{MT}} = \{\mathcal{MT}^{\mathsf{LexI}}_1, \ldots, \mathcal{MT}^{\mathsf{LexI}}_n\}$ for each of the matching tasks in Table 2 with respect to the available reference alignments. We report results 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 tasks.

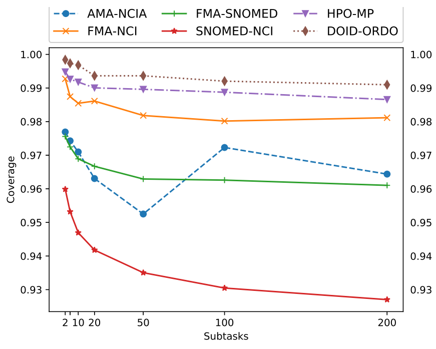
We have compared the two strategies for different number of clusters or resulting matching subtasks $n \in \{2, 5, 10, 20, 50, 100, 200\}$. For the naive strategy, as a random split of LexI is performed, we run 10 experiments for each of the values of n to evaluate the effect of different random selections. The variations in the size of the obtained matching tasks was negligible. Results represent the average of the 10 experiments

Coverage ratio. Figure 2 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. 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 1 as the coverage with respect to system-generated mappings is expected to be even better.

[6] StarSpace: `https://github.com/facebookresearch/StarSpace`
[7] 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.
[8] Java codes: `https://github.com/ernestojimenezruiz/logmap-matcher`
[9] Python codes: `https://github.com/plumdeq/neuro-onto-part`
[10] Extended evaluation material in [1] and `https://doi.org/10.5281/zenodo.1214149`

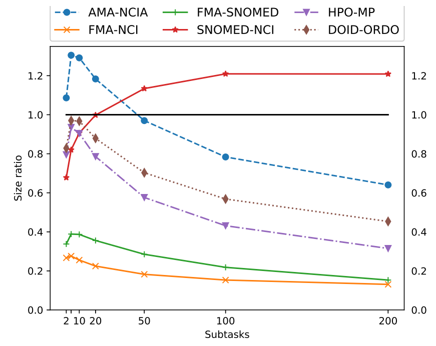


(a) Naive strategy

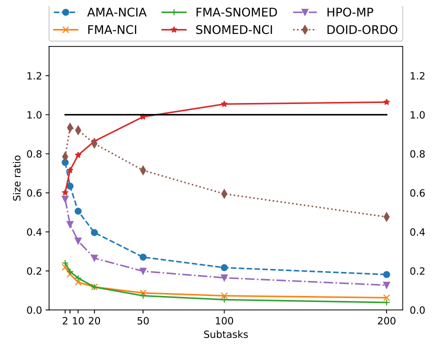


(b) Neural embedding strategy

Fig. 2: CoverageRatio of $\mathcal{D}^n_{\mathcal{MT}}$ with respect to the number of matching subtasks n.



(a) Naive strategy

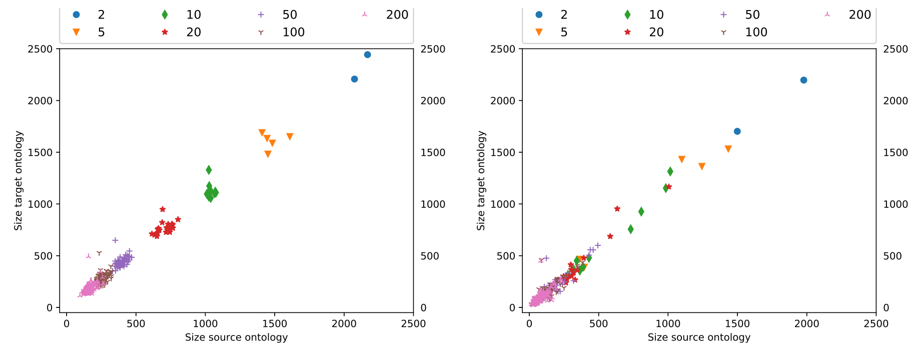


(b) Neural embedding strategy

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

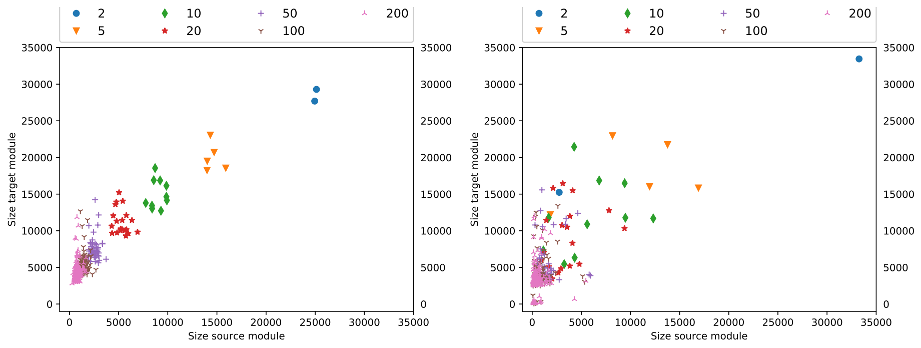
Size ratio. The results in terms of the size (*i.e.*, search space) of the selected divisions $\mathcal{D}^n_{\mathcal{MT}}$ are presented in Figure 3 for the naive (left) and neural embedding (right) strategies. The results with the neural embedding strategy are extremely positive, while the results 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 3. 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 4 and 5 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 $\big(|Sig(\mathcal{O}^i_1)|, |Sig(\mathcal{O}^i_2)|\big)$ 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 4 shows the plots for the AMA-NCIA case while Figure 5 for the FMA-NCI case, using the naive (left) and neural embedding (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 mod-



(a) Naive strategy (b) Neural embedding strategy

Fig. 4: Source and target module sizes in the computed subtasks for AMA-NCIA.



(a) Naive strategy (b) Neural embedding strategy

Fig. 5: Source and target module sizes in the computed subtasks for FMA-NCI.

ule. Nonetheless, on average, the (aggregated) size of the matching tasks in the neural embedding strategy are significantly reduced as shown in Figure 3.

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 [10]. The creation of LexI 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.

4.2 Evaluation of OAEI systems

In this section we support Hypothesis 2 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: MAMBA [5], GMap [13], FCA-Map [6], KEPLER [14], and POMap [7]. MAMBA and GMap failed to complete the OAEI 2015 Anatomy track [2] 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

Table 3: 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

memory) [2].[11] Note that GMap and MAMBA were also tested in the OAEI 2015 with 14Gb of memory. This new setting allowed GMap to complete the task [2].

Table 3 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 matching subtasks is beneficial for FCA-Map, KEPLER and POMap in terms of computation times. 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 3.
vi) The f-measure is slightly reduced as the size of n increases.

Comparison with OAEI results. There are *baseline* results in the OAEI for the selected systems [2], with the exception of MAMBA where the results are novel for the *anatomy* track. 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 lower for GMap. 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 [2]. The authors of FCA-Map have also recently reported results for an improved version of FCA-Map [15]. They completed the FMA-NCI task in near

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

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.

5 Related work

Partitioning has 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. Although our approach does not compute (non-overlapping) partitions of the ontologies, it can be considered a dependent technique.

Prominent examples of ontology alignment systems including partitioning techniques are Falcon-AO [16], COMA++ [17] and TaxoMap [18]. COMA++ and Falcon-AO 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 [18] 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).

The above approaches, although they present interesting results, did not provide any guarantees about the coverage (as in Definition 2) of the discovered partitions. In [19] 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 was also highlighted in a preliminary work by Paulheim [20].

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. 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. The achieved high coverage (*i.e.*, minimal information loss) in combination with the reduction of the search space and the small computation times suggests that the computed divisions based on LexI are suitable in practice. The division of the matching task allowed us to obtain results for five systems which failed to complete these OAEI matching tasks in the past.

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 plan to extend the conducted evaluation to better understand the impact of the division over different ontology alignment systems. We also aim at studying different notions of *context* tailored to the ontology alignment task.

Acknowledgements. EJR was funded by the Centre for Scalable Data Access (SIRIUS), the RCN project BigMed, and The Alan Turing project AIDA.

References

1. Jimenez-Ruiz, E., Agibetov, A., Samwald, M., Cross, V.: Breaking-down the Ontology Alignment Task with a Lexical Index and Neural Embeddings. arXiv (2018) Available from: `https://arxiv.org/abs/1805.12402`.
2. (Ontology Alignment Evaluation Initiative) : `http://oaei.ontologymatching.org/`.
3. Ngo, D., Bellahsene, Z., Coletta, R.: YAM++ results for OAEI 2011. In: 6th International Workshop on Ontology Matching. (2011)
4. Huber, J., Sztyler, T., Nößner, J., Meilicke, C.: CODI: combinatorial optimization for data integration: results for OAEI 2011. In: 6th Int'l Workshop on Ontology Matching. (2011)
5. Meilicke, C.: MAMBA - results for the OAEI 2015. In: 10th International Workshop on Ontology Matching. (2015) 181–184
6. Zhao, M., Zhang, S.: FCA-Map results for OAEI 2016. In: 11th International Workshop on Ontology Matching. (2016)
7. 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
8. 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)
9. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: StarSpace: Embed All The Things! arXiv (2017)
10. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. **31** (2008) 273–318
11. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
12. Harrow, I., Jiménez-Ruiz, E., et al.: Matching disease and phenotype ontologies in the ontology alignment evaluation initiative. J. Biomedical Semantics **8**(1) (2017)
13. Li, W., Sun, Q.: GMap: results for OAEI 2015. In: 10th International Workshop on Ontology Matching. (2015) 150–157
14. Kachroudi, M., Diallo, G., Yahia, S.B.: OAEI 2017 results of KEPLER. In: 12th International Workshop on Ontology Matching. (2017) 138–145
15. 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
16. Hu, W., Qu, Y., Cheng, G.: Matching large ontologies: A divide-and-conquer approach. Data Knowl. Eng. **67** (2008) 140–160
17. Algergawy, A., Massmann, S., Rahm, E.: A clustering-based approach for large-scale ontology matching. In: ADBIS. (2011) 415–428
18. 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
19. Pereira, S., Cross, V., Jiménez-Ruiz, E.: On partitioning for ontology alignment. In: International Semantic Web Conference (Posters & Demonstrations). (2017)
20. Paulheim, H.: On Applying Matching Tools to Large-scale Ontologies. In: OM. (2008)