GBKOM: A generic framework for BK-based ontology matching

Amina Annane, Zohra Bellahsene

GBKOM: A Generic framework for BK-based Ontology Matching

Amina Annane[a,b], Zohra Bellahsene[a]

[a] *Université Montpellier, Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), France*
[b] *Ecole nationale Supérieure d'Informatique, BP 68M, 16309, Oued-Smar, Alger, Algérie*

Abstract

BK-based matching exploits external background knowledge resources (BK) to fill the semantic gap between the ontologies to align. Existing BK-based matchers implement the indirect matching approach in their internal architecture, which makes any adaptation or reuse of the code difficult. Indeed, to improve a particular step in the BK-based matching process, it is necessary to code the whole process from scratch which requires a lot of time and effort. To overcome this issue, we propose a flexible framework called Generic BK-based Matcher (GBKOM). GBKOM is an extension that can be added to any existing matcher. It is a configurable framework that implements the BK-based matching process, with a rich set of parameters making it customizable and suitable for performing experimental evaluations.

GBKOM has participated, with YAM++ as a direct matcher, in the OAEI 2017 and OAEI 2017.5 campaigns, where it has been successful on the biomedical benchmarks, and top ranked in several tasks. Furthermore, we have performed experiments with two other direct matchers (i.e., LogMap and LogMapLite) to show that the effectiveness of GBKOM is independent of the direct matcher used.

Keywords: Ontology matching, Ontology alignment, Background knowledge, Indirect matching, External resource, Anchoring, Derivation, Background knowledge selection.

1. Introduction

Ontology matching is an active area of research because of its multiple applications [1]. Original ontology matching methods are based only on the exploitation of the lexical and structural content of the ontologies to align, which is known as *direct matching* or *content-based matching*. However, direct matching is less effective when the ontologies to align are highly heterogeneous: (i) use different labels to describe equivalent concepts, or (ii) are structured according to different modeling points of view [2, 3].

To overcome this semantic heterogeneity, the community has turned to the exploitation of external knowledge resource(s), commonly called background knowledge resources (BK), as a semantic bridge between the ontologies to align. In contrast to direct matching, this approach is known as *indirect matching*, *BK-based matching* or *context-based matching* [4]. We note that the objective of BK-based matching is to complement direct matching, but not to replace it. Indeed, direct matching may identify mappings that are missed by the BK-based matching and vice-versa.

Fig. 1 shows a realistic example in the context of life-sciences, originally presented in [5]. When directly matching the ontology CRISP to the ontology MeSH, no relation is found between the two concepts CRISP:Brain and MeSH:Head because there is no syntactic similarity between the concept labels. MeSH ontology contains the concept Brain but it is classified under the concept *Central nervous system*, which is no way related to the concept Head (different modeling views). To overcome this semantic heterogeneity, we may exploit an external knowledge resource, the FMA ontology in our example, in order to discover, in an indirect manner, a mapping between these two concepts. For that purpose, the concept CRISP:Brain is anchored (i.e., matched) to the concept BK:Brain and the concept MeSH:Head is anchored to the concept BK:Head. Inside the BK (i.e., FMA ontology), the concepts BK:Brain and BK:Head are related via the relation *is part of*. Hence, we can derive the correspondence: CRISP:Brain *is part of* MeSH:Head.

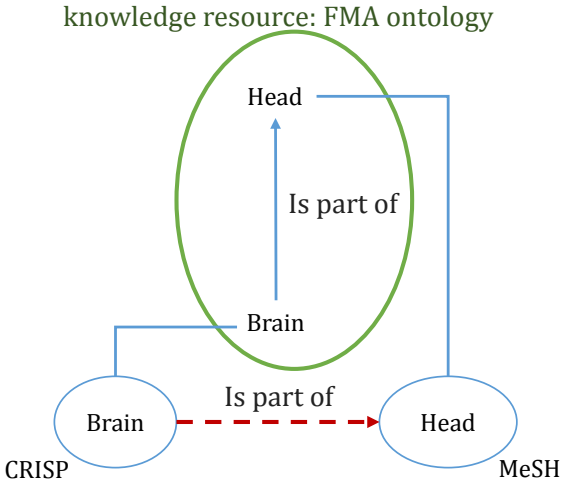
Fig. 2 provides another example of how the exploitation of external knowledge resources reduces heterogeneity between the ontologies to align. The BK concept UBERON:atlas has multiple synonyms that allow to discover the mapping between MA:Atlas and NCIT: C1 Vertebra.

BK-based matching has two main steps: (i) BK selection that determines the external resources to be used among the available ones, (ii) BK exploitation that exploits the selected resources to discover mappings. Evaluating BK selection is tightly related to the results of BK exploitation. Indeed, the benefit of exploiting a given BK can be assessed only at the end of the BK-based matching process by comparing the alignments obtained with and without this BK. Therefore, to perform experiments, one

Email addresses: a_annane@esi.dz (Amina Annane), bella@lirmm.fr (Zohra Bellahsene)

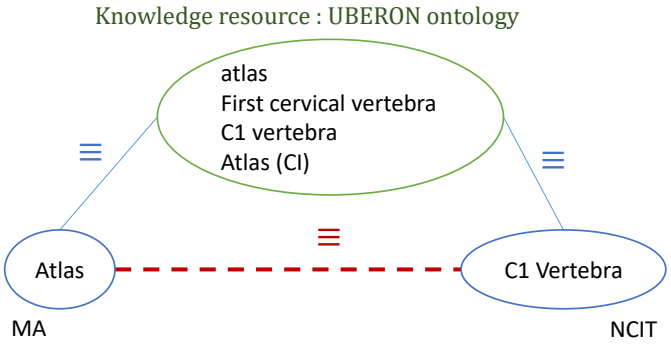
[ht]

Figure 1: Example of BK-based matching.



[ht]

Figure 2: Example of BK-based matching.



has to deal with the whole BK-based matching process, even if she/he wants to focus on a specific step (i.e., BK selection or BK exploitation). Indirect matching modules that are implemented in existing matchers such as AML or LogMapBio are tightly related to their internal architectures. Hence, reusing these modules requires a study and an adaptation of their code, which is not always an easy task. The unique existing generic BK-based matcher is Scarlet [6], however, according to the corresponding author – Marta Sabou –, the Scarlet code is heavily outdated and no more functional. In this paper, we present a Generic BK-based ontology Matcher (GBKOM). It implements our BK selection and exploitation methods described in [7]. In addition, GBKOM has been enriched with new modules to improve the derivation efficiency (see Section 6) and generate mappings with relations other than equivalence (see Section 5). Furthermore, our framework provides a rich set of parameters that enables different configurations, and may be easily coupled with any existing matcher. This is particularly interesting to perform experiments.

We have participated in OAEI 2017 and OAEI 2017.5 campaigns with GBKOM using YAM++ [8] as a direct matcher, that we called YAM-BIO. YAM-BIO obtained good results and was top ranked in several tasks. Moreover, we performed experiments with other direct matchers (LogMap and LogMapLite), to show that GBKOM is generic, and its effectiveness is independent of the direct matcher used.

To wrap up, the contributions of this paper are as follows:

- A novel and efficient derivation algorithm ;
- An algorithm to generate mappings with relations other than equivalence ;
- Empirical evaluation over openly available benchmarks to show that GBKOM is generic, and its effectiveness is independent of the used direct matcher ;
- An open source implementation of our system providing the possibility to use only a subset of modules.

Outline. The rest of this paper is organized as follows. In Section 2, we provide the basic notions used herein. Then, before presenting our approach, in Section 3, we provide a summary of BK-based matching approach. We give an overview of GBKOM architecture and explain its various parameters in Section 4. Then, we present the method dealing with BK building with internal exploration in Section 5. We present the new derivation algorithm in Section 6. After that, we evaluate the proposed algorithms, and GBKOM with different matchers in Section 7. Finally, we conclude in Section 8.

2. Preliminaries

In this section, we define the main terms that we will use in the following sections. Most of these definitions were adopted from [9, 1].

A **Similarity measure** is a function $f : E_s \times E_t \rightarrow [0..1]$ where E_s is the set of O_s entities and E_t is the set of O_t entities. For each pair of entities (e_s, e_t), a similarity measure computes a real number, generally between 0 and 1, expressing the similarity between the two entities by comparing their syntactic or structural information [10, 11]. Other similarity measures use external resources such as WordNet to compute the similarity between the two entities [12].

A **Mapping** (or a correspondence) between an entity e_s (e.g., concept, relation) belonging to ontology O_s and an entity e_t belonging to ontology O_t is a four-tuple of the form: $m = \langle e_s, e_t, r, k \rangle$ where:

- r is a relation between e_s and e_t such as equivalence ($\equiv$), subsumes ($\sqsupseteq$), subsumed by ($\sqsubseteq$), etc.
- k is a confidence score (typically in the $[0, 1]$ range) holding for the correspondence between the entities e_s and e_t. The k value is the score returned by one similarity measure, or a combination of several ones.

Ontology matching can be formally defined as a function that takes two ontologies O_s and O_t, a set of parameters P, and a set of resources R, and returns an alignment A between O_s and O_t. An **Alignment** between

O_s and O_t is a set of mappings between their entities (concepts and properties).

A **Matcher** is an algorithm that implements one similarity measure or combines several ones to discover mappings between the input ontologies. In addition, a matcher includes a decision function that selects which mappings will be kept in the produced alignment [13]. For instance the decision function may be based on a threshold value: only mappings that have a score equal to or greater than the threshold value are returned.

Background knowledge In the context of ontology matching, there is no commonly accepted or strict definition of what background knowledge is. We define it as any set of external knowledge resources that provides lexical or semantic information about the domain(s) of the ontologies to align or some of the entities therein. It could be any datasets related to the ontologies to align, other ontologies than the ones to align, other previously generated mappings, lexical sources, the Web, etc.

In this paper, we use the acronym **BK** to refer to a non empty set of background knowledge resources used within the matching process. For instance, if such a resource is an ontology, we will call it a *BK ontology*. Similarly, the expression *BK-based method* denotes a method that exploits a set of background knowledge resources within the matching process.

OAEI. Organizing and evaluating the growing number of ontology alignment systems (or methods) needs unified rules and organization. Ontology Alignment Evaluation Initiative (OAEI) is a coordinated international initiative to fill this need [14]. It has held an annual evaluation of ontology alignment systems since 2004.[1] The OAEI has different tracks such as Anatomy, Conference, MultiFarm, etc. The results of the participating systems are published for further analysis.

YAM++ is a matcher previously developed by our team at LIRMM[2] [8]; it does not rely on a specialized BK to match ontologies. YAM++ combines several syntactic and structural similarity measures. It is considered as one of the state-of-the-art ontology matchers, and was the top ranked matcher in OAEI 2013.

3. Ontology matching systems using BK

3.1. Overview of BK-based ontology matching

The idea of using background knowledge for enhancing ontology matching task is not new and has been successfully adopted in several matching systems (see Section 3.2).

Fig. 3 summarizes the whole process of BK-based ontology matching approach with its main steps: (i) BK selection and (ii) BK exploitation. In the literature, several works have addressed these two issues jointly or separately.

BK selection is the process that attempts to select effective background knowledge resources, for a given matching task, from the set of available resources that we call *knowledge resource pool* [15, 16, 17, 4]. Effective background knowledge resources, to be used in the matching process, are those containing knowledge beyond that contained in the ontologies to match but which is relevant to match them. However, there is no measure that allows evaluating the effectiveness of a given BK before exploiting it. Indeed, currently, the evaluation is based on the comparison between the alignments obtained with and without the exploitation of a given BK.

BK exploitation consists in using the BK selected during the previous step to enhance the matching result. It has three main steps: the first step, called *anchoring* consists in retrieving correspondences between the entities of the ontologies to align, and the BK entities that we call *anchors*. This is usually done by matching source and target ontologies to the selected BK. Then, the second step called *derivation* consists in deducing semantic relationships between entities of ontologies to be aligned according to the relationships linking the source and target anchors in the used BK e.g., deducing that CRISP:Brain is part of MeSH:Head in Fig. 1. Finally, the last step is called *mapping selection*; it relies in selecting the most relevant correspondences among the candidate ones, which is particularly challenging in the context of indirect matching. Indeed, the use of background knowledge resources in ontology matching is a double-edged sword: though these resources provide information to discover new correct correspondences, they also generate new incorrect mappings [4].

For more details about the BK-based matching process please refer to [18], chapter Foundations.

3.2. Related work

As we have discussed before, to the best of our knowledge, currently there is no flexible framework for BK-based ontology matching. However, several systems implement the BK-based approach in their internal architecture. We summarize here the main BK-based ontology matching systems, and we discuss the added value of GBKOM compared to existing systems.

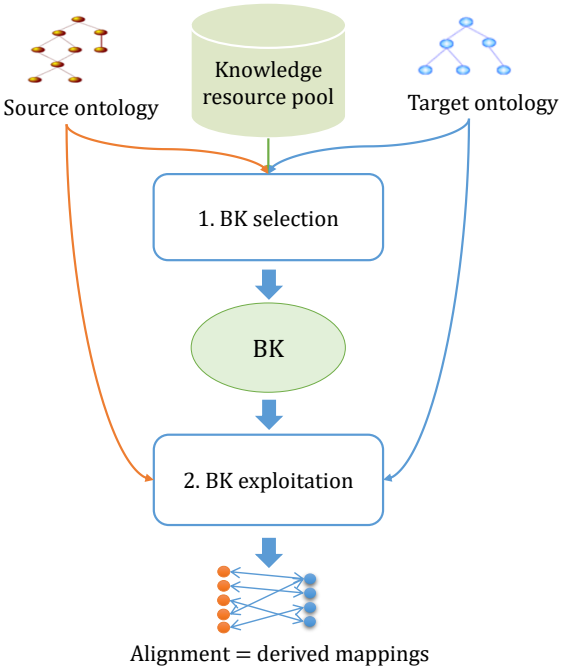
GOMMA-BK (Generic Ontology Matching and MApping management). To the best of our knowledge, it is the first system that has implemented BK-based approach in 2012 by using mapping composition [19]. GOMMA-BK allows the improvement of matching quality by using domain knowledge resources (i.e., domain-specific hub ontologies) namely: UMLS [20], UBERON [21] and FMA [22].

AML-BK is a BK-based version of AgreementMakerLight ontology matching system [23]. AML-BK used the UBERON ontology as BK in 2013. Since 2014, AML-BK takes as a BK: (i) the Lexicon file of the Medical Subject Headings (MeSH ontology), and (ii) ontologies that are automatically selected for each matching task from a knowledge resource pool composed of two ontologies UBERON

[1] `http://oaei.ontologymatching.org/`.
[2] `http://www.lirmm.fr/`.

[ht]

Figure 3: General workflow of BK-based ontology matching.



and DOID (Human Disease Ontology). The automatic selection is based on the *Mapping Gain* measure [15].

LogMap-BK/LogMapBio. They are two versions of the LogMap ontology matching system that use BK [24]. LogMap-BK uses UMLS Lexicon while LogMapBio includes an extension for selecting dynamically a set of biomedical ontologies as BK from NCBO BioPortal [17].

Note that, from 2016 on, AML does not use anymore the suffix BK in its name although it actually uses BK (the same thing for LogMap since 2012).

Indirect matching modules that are implemented in existing matchers such as AML or LogMapBio are tightly related to their internal architectures. Hence, reusing these modules requires a study and an adaptation of their code, which is not an easy task. The single generic BK-based matcher was Scarlet [6], which is no more functional. Our framework GBKOM has been designed to be used with any matcher, and offers many configurations thanks to its multiple parameters. All the systems discussed previously exploit several BK ontologies for a given matching task. However, each BK ontology is exploited independently of the others (i.e., one BK ontology at time). GBKOM allows to combine and derive mappings over several BK ontologies, which is more effective [6].

4. GBKOM framework

GBKOM is the implementation of our BK-based matching approach described in [7]. For the sake of clarity, we start by giving an overview of our approach. Then, we present the different modules composing GBKOM.

4.1. Overview of our approach

Our approach follows the general BK-Based matching presented in Section 3:

4.1.1. BK selection/BK building

BK selection in our approach is called BK building. Unlike existing works that select a subset of resources from the knowledge resource pool [6, 25, 16, 15], our approach builds a novel BK from the knowledge resource pool, which allows to improve effectiveness and efficiency. The knowledge resource pool is composed of a set of ontologies of the same domain as that of the ontologies to align that are manually identified. Furthermore, when available, mappings between the preselected ontologies may be added to the knowledge resource pool, in particular those that are manually created or human-curated. For instance, in the biomedical domain, cross-references between OBO (Open Biological and Biomedical Ontology) Foundry ontologies[3] [26] may be considered as equivalence mappings that are manually curated. The **manual** identification of ontologies that compose the knowledge resource pool among all the existing ones may be seen as a preselection, this is why we refer to these ontologies as "preselected ontologies". BK building has three main substeps (see Fig. 4): (i) mapping extraction that extracts all possible mappings between the preselected ontologies, (ii) mapping filtering that returns only mappings that are directly or indirectly related to the source ontology, which we refer to as "filtered mappings" and (iii) mapping combination that merges the filtered mappings into one graph that we call the *built BK*. The built BK is a graph where nodes are concepts coming from the preselected ontologies, and edges are mappings (i.e., filtered mappings) linking these concepts. The built BK is the output of the BK building step, and will be used in the BK exploitation step.

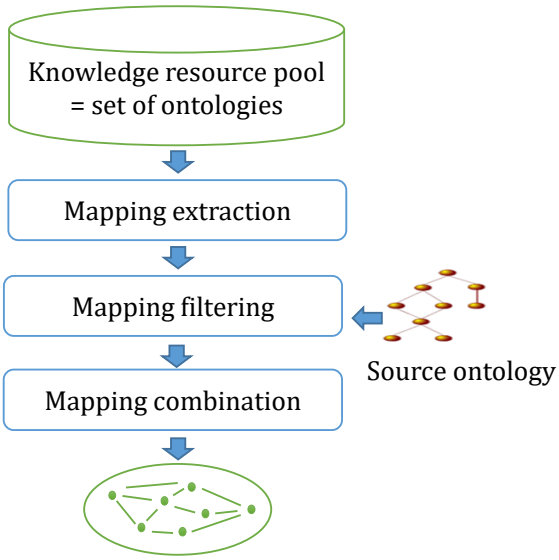
4.1.2. BK exploitation

BK exploitation comprises the three classical steps: (i) anchoring consists in matching source and target ontologies to the built BK, (ii) derivation: consists in retrieving all paths that link source concepts to target concepts through the internal structure of the built BK, and (iii) mapping aggregation and selection. Several paths may represent the same mapping (i.e., several paths link the same source and target entities), hence the need of aggregating them, then selecting the most relevant ones. Mapping scores are computed at this step, namely when using the rule-based selection method. The score of each path is the multiplication of its intermediate scores. Then, the score of a given mapping is the maximum of the different path scores representing that mapping. When using the machine-learning based method, each mapping is classified into True (i.e.selected) or False (i.e., not selected) without a specific score. Please, refer to [7] for more information about the aggregation and selection methods.

[3] `http://www.obofoundry.org/`.

[ht]

Figure 4: BK building: BK selection in our approach.



4.2. GBKOM modules and parameters

Fig. 5 shows the five main modules that compose GBKOM: (i) BK building, (ii) Anchoring, (iii) Derivation, (iv) Mapping aggregation and selection, and (v) Semantic verification. We grouped the input parameters in categories (e.g., derivation parameters, selection parameters, etc.). In the following, we will present these modules and their parameters.

4.2.1. Direct matcher

During BK building and Anchoring steps, a matcher is required to generate alignments between ontologies. In our framework, any existing direct matcher that implements a basic function *Align*, which takes as input two ontology URLs (i.e., source and target URL) and returns the URL of the generated alignment could play the role of a *direct matcher*. In this paper, we performed experiments with three different matchers: YAM++, LogMap and LogMapLite. Moreover, the matcher should store the generated alignment in RDF format with the API alignment [27] to be parsed correctly. Systems that have participated in the OAEI campaigns, may use GBKOM directly without any adaptation. Indeed, OAEI participants have to wrap their tools as SEALS packages, and the wrapping procedure includes the implementation of the function Align.[4] Basically, the direct matcher does not exploit BK resources, and GBKOM is an extension to improve its results. However, there is no restrictions and even BK based matchers may be used as a direct matcher.

4.2.2. Knowledge resource pool

In GBKOM, the knowledge resource pool is a set of knowledge resources provided by the user. More precisely, GBKOM supports two resource types: (i) ontologies, and (ii) existing mappings (i.e., precomputed mappings). Since we use Jena API to load and parse ontologies, all ontology formats supported by Jena API, such as RDF and OWL, are supported by GBKOM. However, the format of the background knowledge ontologies should be supported by the direct matcher too. Existing mappings may be provided in two formats: (a) RDF format: alignments stored using the alignment API [27] or (b) CSV format where each row has a value for the different attributes illustrated in Fig. 6. The attribute "source" refers to the name of matcher that generated the mapping, or the resource that includes it. Once GBKOM is launched, the set of resources is fixed. However, the user is free to add or remove any resource before launching the matching process.

4.2.3. Alignment repository

Alignment repository is a folder that includes several subfolders, where each one is dedicated to a given matcher (e.g., logmap folder or YAM++ folder). Each subfolder contains alignments that are generated by its matcher in RDF format. The idea is to avoid aligning the same pair of ontologies with the same matcher more than once to gain in efficiency. Hence, before performing any matching task for BK building or anchoring, GBKOM verifies if an alignment between the input ontologies exists to reuse it. Otherwise, a new alignment is generated and saved in the alignment repository.

4.2.4. BK building

The BK building module is the implementation of the approach of BK selection described in Section 4.1.1. It takes two parameters: (i) direct matcher to generate the required alignments and (ii) source ontology to filter the extracted mappings.

As we have experimentally showed in [7], following our approach, the *built BK* has a very reduced size comparing to that of the preselected ontologies, which improves the efficiency of the BK selection process. Furthermore, the *built BK* interconnects concepts from different preselected ontologies via mappings, thereby allowing deriving mappings across several intermediate ontologies. In addition, we propose a new method that allows to enrich the built BK with internal relations that offers new parameters (see Section 5).

4.2.5. Anchoring

As we have explained previously, anchoring is a direct matching between the ontologies to align and the built BK. Its main parameter is the direct matcher. Anchoring mappings are added to the built BK.

4.2.6. Candidate mapping derivation

Our framework provides two mapping derivation strategies. The first one assumes that the *built BK* is stored as a Neo4j graph database. It consists in searching all possible

[4] `http://oaei.ontologymatching.org/2017/`.

Figure 5: GBKOM architecture.

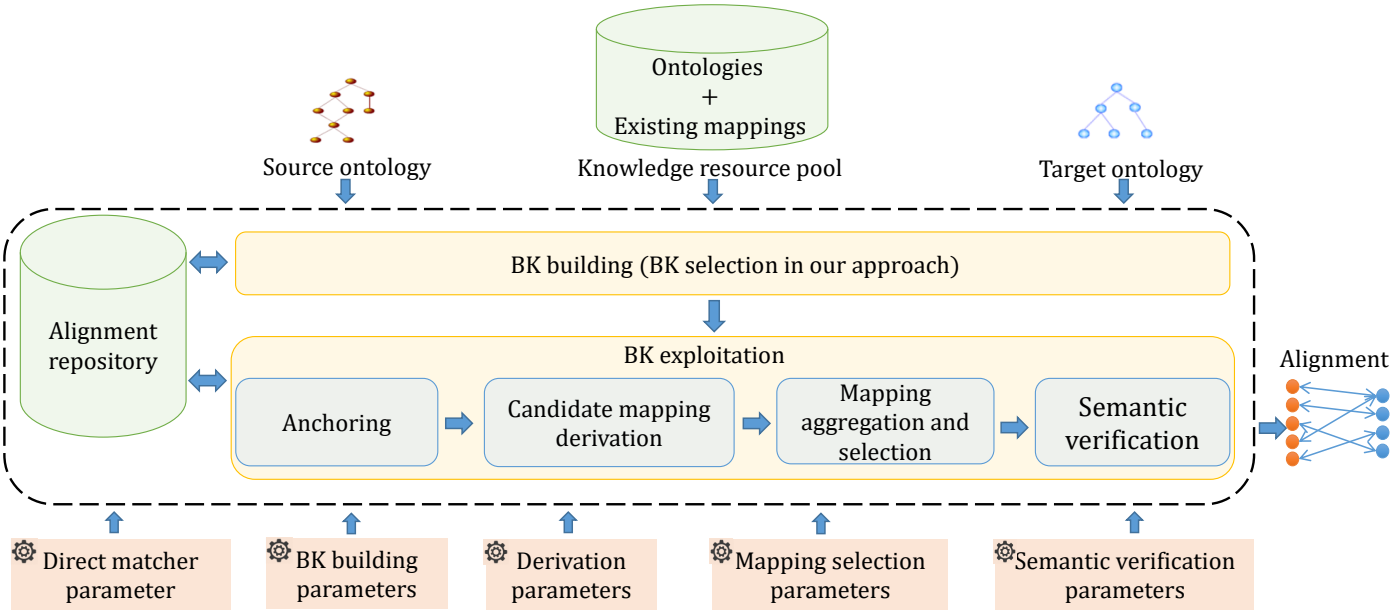


Figure 6: Example of an existing mapping format.

Attribute	**Value**
URI source	http://bioontology.org/projects/ontologies/fma/fmaOwlDlComponent_2_0#Abdominal_aorta
Ontology source	http://bioontology.org/projects/ontologies/fma/fmaOwlDlComponent_2_0
URI target	http://purl.obolibrary.org/obo/UBERON_0001516
Ontology target	http://purl.obolibrary.org/obo/uberon.owl
Score	0.99
Relation	=
Source	YAM++

paths between source and target concepts. This derivation strategy is complete, i.e., it returns all possible candidate mappings, however it is not scalable for large *built BK* graphs. Furthermore, it depends on Neo4j. We tried to address these issues by implementing Algorithm 2, which represents the second derivation strategy. Algorithm 2 returns paths between source concepts and target concepts too, but it implements some constraints that reduce the number of returned paths, which improves efficiency and consumes less memory and computational resources without decreasing effectiveness. In Section 6, we explain in detail and evaluate this Algorithm.

In both cases, the user has to specify the *Maximum path length* parameter, which is the maximum length of paths to be returned by the derivation process (by default it is 4). The length of a given path is the number of its nodes (see Table. 1).

Table 1: Mapping derivation parameters.

Parameter	Possible values
Derivation strategy	All paths or Algorithm 2
Maximum path length	Integer > 0, by default 4

4.2.7. Mapping aggregation and selection

GBKOM implements two mapping aggregation and selection methods: (i) Machine Learning (ML) based selection and (ii) Rule-based selection methods, which were proposed and evaluated in [7]. The former is based on supervised machine learning using RandomForest algorithm, while the later is based on a set of rules.

To enable the use of a classification ML algorithm, we designed a set of 27 attributes (or features) that describes each candidate mapping. Most of these attributes (21 attributes) are computed using the intermediate scores of paths representing candidate mappings. For instance, the max-multiplication attribute is computed as follows: (i) for each path returned from the derivation step, we compute the multiplication-value of its intermediate scores, (ii) then, for each candidate mapping, we compute the maximum multiplication-value of the paths representing that candidate mapping. Using the same method we compute other attributes such as max-average, min-sum, max-variance, etc. Other attributes are taken into account such as the number and length of paths representing each candidate mapping. When choosing the ML-based selection, the user has to provide one or several datasets, such that

each dataset is a folder that contains two ontologies and their validated alignment. These datasets will be used for training the classifier. When using Rule-based selection, the user may specify a threshold value to select only the mappings that have a score equivalent to or higher than this threshold value, by default its value is set to zero (see Table. 2).

Table 2: Mapping selection parameters

Parameter	Possible values
Mapping selection	ML based or Rule based
Threshold	a real value $\in$ [0..1]
Datasets	a folder for each dataset

4.2.8. Semantic verification

Currently, GBKOM reuses the LogMapRepair module [28] to verify the consistency of the generated alignment.

LogMapRepair takes as parameter the reasoner to use as parameter that may be Hermit or Alcomo. The semantic verification is optional and the user may disable it using the *Semantic verification* parameter (see Table. 3).

Table 3: Semantic verification parameters

Parameter	Possible values
Semantic verification	Yes or No
Reasoner	Hermit or Alcomo

5. Discovering richer relations with internal exploration

5.1. BK building with internal exploration

Most ontology matching systems generate only equivalence mappings. Hence, using these matchers, the *built BK* can be exploited to derive only equivalence mappings too.

To enable deriving mappings with other relations than equivalence such as subClassOf, we have to enrich the *built BK* with this kind of relations. To that end, we have extended our BK selection approach to explore the internal structure of the preselected ontologies and extract fragments rather than only concepts from these ontologies [7]. The structure exploration is controlled by two parameters:

- Exploration relations. They are the mapping relations that the user wants to generate in addition to equivalence mappings such as *subClassOf*, and *partOf* relations. In GBKOM, we assume that the relations provided by the user are **transitive**.
- Exploration length. This parameter limits the internal exploration within a given preselected ontology to a number of steps. For instance, an exploration with the relation subClassof and length of 1 returns for each concept that has a mapping in the set of filtered mappings its parents and children (see Fig. 7(b)). If we change the length parameter to 2, the structure exploration returns for each concept its parents, grandparents, children, grandchildren (see Fig. 7(c)).

These parameters are similar to those proposed in [4] for the local inference step. However, in their work, the authors proposed to reload each BK ontology in the BK exploitation step to explore its structure, which is time consuming. Here, we propose to extract the potentially effective fragments from the preselected ontologies in the BK selection step to avoid dealing or reasoning with complete ontologies in the BK exploitation step.

The result of the structure exploration is a set of triples $\langle e_i, e_j, r\rangle$, such that e_i and e_j belong to the same preselected ontology, and r belongs to the exploration relations R. These triples are merged with the concepts of the *built BK*. Originally, each edge has a score in the built BK, which is the confidence value of the mapping that the edge represents. However, the extracted triples have no confidence values. To have a uniform graph, we have assigned the value of 1 to edges with exploration relation. In the following, we use the term **enriched BK** to refer to a BK built with internal exploration.

Note that a large *exploration length* parameter may return large fragments or whole BK ontologies, which limits the benefit of our BK selection approach. Indeed, our approach aims at extracting the effective BK ontology fragments – as small as possible – rather than returning whole BK ontologies.

In Algorithm 1, we present the pseudo code of the BK building with internal exploration. First, we run the BKbuilding method that builds a BK according to the process described in Section 4.1.1, then selects a subgraph of the built BK which is related to the list of *sourceConcepts*. *sourceConcepts* may be all the source concepts or a subset of them. However, we believe it is more efficient to only consider source concepts for which there is no equivalent target concept when exploiting the BK without internal exploration. Indeed, building an enriched BK for the whole source ontology may generate a large graph, which decreases the derivation efficiency and the precision of the generated alignment. Then, we enrich the built BK with the triples returned by the *getTriples* method. This method returns a set of triples of the form $\langle e_i, e_j, r\rangle$ such that e_i or e_j belongs to *newConcepts*, and r is the exploration relation given as input. e_i and e_j belong to the same ontology given as input. To retrieve these triples, we run two SPARQL queries for concepts in *newConcepts* belonging to the input ontology. The first query looks for all triples where e_i belongs to *newConcepts*, while the second query returns all triples where e_j belongs to *newConcepts*. We present an example in Fig. 8 where the exploration relation is rdfs:subClassOf, and *newConcepts* is the set of URIs used to filter e_j values. *getConcepts* method re-

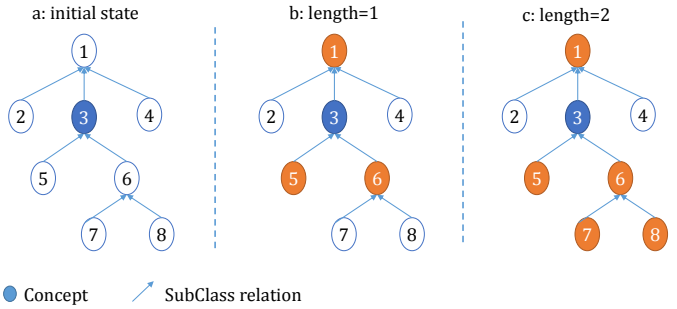
turns the set of unique concepts belonging to its input (i.e., builtBK or triples)

Algorithm 1: BK building with internal exploration

Require: *explorationRelations*, *explorationLength*, *knowledgeResourcePool*, *sourceOntology*, *sourceConcepts*

1: $builtBK \leftarrow BKbuilding(KnowledgeResourcePool, sourceOntology, sourceConcepts)$
2: $enrichedBK \leftarrow builtBK$
3: $newConcepts \leftarrow getConcepts(builtBK)$
4: $oldConcepts = \{\}$
5: **for all** $ontology \in knowledgeResourcePool$ **do**
6: **for all** $relation \in explorationRelations$ **do**
7: **for** $i \leftarrow 0; i < explorationLength; i++$ **do**
8: $triples \leftarrow getTriples(ontology, relation, newConcepts)$
9: $oldConcepts \leftarrow oldConcepts \cup newConcepts$
10: $enrichedBK \leftarrow enrichedBK \cup triples$
11: $newConcepts \leftarrow getConcepts(triples) - oldConcepts$
12: **end for**
13: **end for**
14: **end for**
15: **return** *enrichedBK*

Figure 7: Example: concept selection with structure exploration.



The implementation of Algorithm 1 offers three new parameters summarized in Table. 4. When the user assigns *yes* to the parameter *Internal BK ontology exploration*, GBKOM builds a BK *enriched* with BK ontology relations among the *Internal relations to explore* parameter and the concepts related to these relations within a distance – number of edges – less than the *Exploration length* parameter.

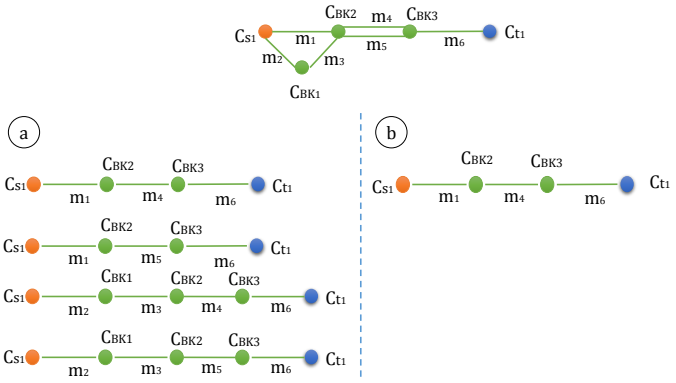
5.2. BK exploitation with internal exploration

The derivation process when exploiting an enriched BK follows the same schema: retrieving paths between source and target concepts. The difference is that the retrieved paths may include relations, other than equivalence, that belong to the exploration relations provided by the user. Hence, it is important to define a strategy to combine these relations. In GBKOM, we only consider paths with at most one type of relation in addition to equivalence. Then, we combine relations according to the following rules. Let $R = \{r_1, r_2, \ldots, r_n\}$ be the set of exploration relations r_i – transitive relations–, $\langle c_i, c_j, r_i \rangle$ a mapping belonging to a retrieved path, then for each r_i in R:

Figure 8: Example: SPARQL query to get sub-concepts.

```
PREFIX   rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT   ?ei   ?ej
WHERE    {?ei   rdfs:subClassOf   ?ej}
VALUES ?ej  {<http://purl.obolibrary.org/obo/UBERON_0000003>
             <http://purl.obolibrary.org/obo/UBERON_0000114>
             <http://purl.obolibrary.org/obo/UBERON_0000115>
                  .....}
```

Figure 9: Derivation algorithm: all paths vs. Algorithm 2.



- $\langle c_1, c_2, r_i \rangle \wedge \langle c_2, c_3, r_i \rangle \implies \langle c_1, c_3, r_i \rangle$: this rule may be applied because r_i is transitive.
- $\langle c_1, c_2, r_i \rangle \wedge \langle c_2 \equiv c_3 \rangle \implies \langle c_1, c_3, r_i \rangle$
- $\langle c_1, c_2, \equiv \rangle \wedge \langle c_2, c_3, \equiv \rangle \implies \langle c_1, c_3, \equiv \rangle$

With these rules, we assume it is not possible to combine a given semantic relation with another relation except equivalence. However, in particular cases, the user may want to define its own relation combination rules. For instance, in [5], the authors showed that for their matching task, the combination of the two relations: *is part of* and *is a*, in a specific order, gives interesting results. In the future, we plan to give possibility to the user to define the rules used for combining relations in GBKOM.

6. Candidate mapping derivation

To participate in the OAEI 2017.5 campaign, we had to execute our algorithms on the Hobbit platform [29] (see Section 7). This platform offers limited memory and computational time resources for each execution. Searching all possible paths between source and target concepts – i.e., the first derivation strategy – in a cyclic graph is a complex task which requires significant resources, especially when the graph has a large size. In Algorithm 2, we

Table 4: BK Building parameters

Parameter	Possible values
Internal exploration	Yes or No
Exploration relations	transitive relations
Exploration length	an integer > 0

attempted to reduce the complexity of the all path algorithm. The main idea is to exploit each BK concept once for a given source concept, which reduces the number of paths to explore during the derivation process, and consequently the final returned paths. In Fig. 9, we present an example to illustrate the difference between the two derivation algorithms. We suppose that all the mappings m_i are equivalence mappings. The all-paths algorithm (case a) returns four paths for the same candidate mapping between the source concept C_{s1} and the target concept C_{t1}, while Algorithm 2 (case b) returns only one path between the two concepts. Indeed, in case b, C_{BK2} and C_{BK3} have been exploited to derive the first path. Hence, they cannot be reused to derive other paths. While in case a, we go through the same BK concepts several times, which requires more time, memory and returns more paths to process.

Algorithm 2 takes as input: (i) $filteredMappings$: mappings that compose the built BK and the anchoring mappings, (ii) $sourceConcepts$ and $targetConcepts$: lists of source concepts and target concepts respectively, and (iii) $maxPathLength$: the parameter that limits the length of derivation paths.

The algorithm starts by adding all the filtered mappings to a collection of type dictionary – map in java – that is represented with the $builtBK$ variable (lines 2 to 4). For instance, if we add the six mappings of the example in Fig. 9 to $builtBK$, we obtain the dictionary in Fig. 10.

Figure 10: $builtBK$ variable after adding the 6 mappings in Fig. 9.

Key	Direct mappings
C_{s1}	{ m_1, m_2}
C_{BK1}	{m_2, m_3}
C_{BK2}	{m_1, m_3, m_4, m_5}
C_{BK3}	{m_6, m_4, m_5}
C_{t1}	{m_6}

$builtBK$ dictionary may be seen as an index of the filtered mappings such that the access keys are the first entities e_i of these mappings. Note that the six mappings are equivalence mappings, hence each mapping is indexed in both directions. However, for no-equivalence mappings, each mapping is indexed once with the provided direction.

After having added the filtered mappings to $builtBK$, we look for paths leading to target concepts from each source concept. In our algorithm, a path is an ordered list of mappings. For each source concept, we retrieve its list of mappings (line 6). Then, for each retrieved mapping, we create a new path (i.e., an empty list) to which we add this mapping. We add the created paths to the set of paths to explore (lines 7 to 11). Note that the first entity e_i of all paths to explore belongs to the source concepts.

While there remain paths to explore, we pick up the top path – the oldest and the shortest path in the list –, we retrieve the last concept that is the second entity e_j of the last mapping in this path (lines 12 to 15). If the last concept has not been exploited before, we get its mappings from $builtBK$, and we add it to the list of exploited concepts (lines 16 to 18). Then, for each of these mappings, we extend the current path by adding this mapping at the end of the path list (line 20). Finally, we verify whether the last concept of the extended path belongs to the list of target concepts. If it does, we add the path to the list of the found paths. Otherwise, we add it to the list of the paths to explore if its size is less than the $maxPathLength$ parameter (lines 22 to 28).

Algorithm 2 should be called twice when the parameter $internalExploration$ is set to "Yes". Indeed, when the built BK contains only equivalence relations, paths linking source concepts to target concepts are the same linking target concepts to source concepts. Hence, it is sufficient to only look for paths from source concepts to target concepts. However, when dealing over several types of relations, the direction matters. We need to look for paths in both directions: from source to target concepts, and also from target to source concepts. For that, we call Algorithm 2 twice by interchanging the lists of source and target concepts: (i) Algorithm 2 (filteredMapping, sourceConcepts, targetConcepts, maxPathLength): returns paths from source to target concepts. (ii) Algorithm 2 (filteredMapping, targetConcepts, sourceConcepts, maxPathLength): returns paths from target to source concepts.

7. Evaluation

In the following, we present an evaluation of the proposed algorithms as well as the alignments generated by GBKOM. We have used Anatomy and LargeBio tracks of OAEI.

Anatomy. It consists in finding an alignment of 1,516 mappings between the Adult Mouse Anatomy (MA: 2,744 concepts) and a subset of the National Cancer Institute Thesaurus (NCIT: 3,304 concepts) describing human anatomy. In the following, we refer to MA as source ontology and NCIT as target ontology.

Large Biomedical (LargeBio) [5]. It aims at finding alignments between several large and semantically rich biomedical ontologies: the Foundational Model of Anatomy (FMA) [22], National Cancer Institute Thesaurus (NCI) [30] and SNOMED Clinical Terms (SNOMED-CT) [31], which contain 78,989, 66,724 and 306,591 concepts, respectively.

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

Algorithm 2: Derivation function

```
Require: filteredMappings, sourceConcepts,
    targetConcepts, maxPathLength
 1: builtBK={}, pathsToExplore={},
    exploitedConcepts={}, foundPaths={}
 2: for all mapping ∈ filteredMappings do
 3:    addMapping(mapping, builtBK)
 4: end for
 5: for all sc ∈ sourceConcepts do
 6:    mappings ← builtBK.get(sc)
 7:    for all m ∈ mappings do
 8:       path ← {}
 9:       path.add(m)
10:       pathsToExplore.add(path)
11:    end for
12:    while pathsToExplore.size() > 0 do
13:       path ← pathsToExplore.get(0)
14:       pathsToExplore.remove(0)
15:       lastConcept ←
          path.getLastMapping().getSecondEntity()
16:       if lastConcept ∉ exploitedConcepts then
17:          mappings ← builtBK.get(lastConcept)
18:          exploitedConcepts.add(lastConcept)
19:          for all m ∈ mappings do
20:             path.add(m)
21:             lastConcept ← m.getSecondEntity()
22:             if lastConcept ∈ targetConcepts then
23:                foundPaths.add(path)
24:             else
25:                if path.size() < maxPathLength then
26:                   pathsToExplore.add(path)
27:                end if
28:             end if
29:          end for
30:       end if
31:    end while
32:    pathsToExplore.clear()
33:    exploitedConcepts.clear()
34: end for
35: return foundPaths
```

In Table. 5, we present the six matching tasks of LargeBio corresponding to the different sizes of input ontologies (small fragments/whole ontology of FMA and NCI and small/large fragments of SNOMED-CT). The last column shows the number of mappings in the reference alignment. The Unified Medical Language System (UMLS) [20] has been used as the basis to produce the reference alignments [32].

We have performed our experiments with the following parameter values:

- Direct matcher: YAM++
- Knowledge resource pool: (i) UBERON and DOID ontologies, and (ii) cross-references extracted from these ontologies as equivalence mappings that we call "OBO mappings". We have assigned a score of 1 for OBO mappings as they have been manually curated.
- Derivation strategy: Algorithm 2
- Maximum path length: 4
- Internal exploration: No (except in Section 7.1)
- Mapping selection strategy: Rule based
- Threshold: 0.0
- Semantic verification: True
- Reasoner: Hermit

7.1. BK exploitation with internal exploration

To the best of our knowledge, there is no biomedical benchmark to evaluate a matcher tool that returns other kind of mappings than equivalence. Evaluating the BK building with internal-relation enrichment requires the production of such a benchmark by experts. Therefore, we present here only a preliminary evaluation.

Setup of experiments

We performed our preliminary evaluation on the Anatomy track [33]. We chose Anatomy because it is the smallest and the single biomedical OAEI track that has a manually validated gold standard.

We executed the matching process as follows: (i) GBKOM starts by looking for the equivalence mappings according to the process illustrated in Fig. 5 until the derivation step; (ii) then, it builds an enriched BK only for the source concepts that do not have any candidate mapping at the end of the derivation step.

We have assigned the value 1 to the internal *exploration length* parameter, and rdfs:subClassOf as the *relation to explore* parameter.

Result

At the end of the derivation step using the built BK without enrichment, 1,269 MA concepts (46% of all MA concepts) had no mapping candidates i.e., no derivation path starts with one of these 1,269 concepts. An enriched BK has been built for these 1,269 mouse concepts (i.e., the

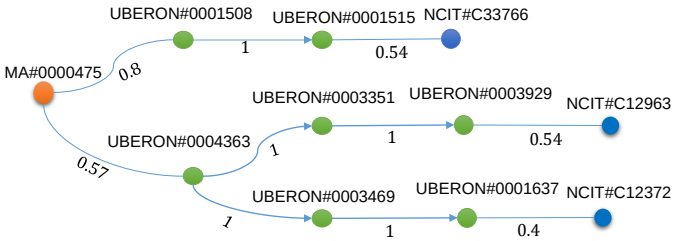
Table 5: Matching tasks of OAEI LargeBio.

Task #	Task name	#Source	#Target	#Mappings
Task 1	FMA-NCI small fragments	3,696	6,488	2,686
Task 2	NCI-FMA Whole ontologies	66,724	78,989	2,686
Task 3	FMA-SNOMED small fragments	10,157	13,412	6,026
Task 4	FMA whole with SNOMED large fragment	78,989	122,464	6,026
Task 5	NCI-SNOMED small fragments	23,958	51,128	17,210
Task 6	NCI whole with SNOMED large fragment	66,724	122,464	17,210

list of source concepts in Algorithm 1). The derivation using the enriched BK returned 965 candidate mappings with rdfs:subClassOf relation. 517 MA concepts are included in these candidate mappings, which represents 41% of the initial set of concepts (i.e., 1,269 concepts) for which no candidate mapping was found using the built BK without enrichment.

The difference between the two numbers 965 and 517 is explained by the fact that some source concepts belong to several mappings. We present an example in Fig. 11. The directed edge are subClassOf relations, while undirected edges are equivalence edges. From this subgraph three mappings are derived that have the same source concept. Currently, all the derived no-equivalence mappings are returned. Note that a high value of the *explorationlength* parameter generates a larger enriched BK, and may return more no-equivalence and less precise mappings if the *maxpathlength* parameter has a high value too. In the future, we plan to define a new method for selecting the most relevant no-equivalence mappings, and develop a new benchmark to perform an advanced evaluation of the BK-based matching with internal exploration.

Figure 11: Several subClassOf mappings for the same source concept.



We observed that some derived mappings could be inferred from the equivalence mappings without exploiting the enriched BK. For instance, in the mouse anatomy ontology, the concept (MA_0002028;pudendal artery) is a subclass of the concept (MA_0000064;artery). The concept (MA_0000064;artery) has an equivalent concept in the NCIT ontology (NCI_C12372; artery), while the concept *pudendal artery* does not. Thus, it is possible to derive that (MA_0002028; pudendal artery) is subClassOf (NCI_C12372; artery) without exploiting the enriched BK. In the future, we plan to implement techniques to select only the concepts for which no mapping can be inferred.

We manually evaluated 40 subClassOf mappings that we have randomly selected. All the evaluated mappings were correct. We present some examples in Table. 6. The list of the generated candidate mappings wit their paths, as well as the 40 mappings validated are available in the file *SubClassOfMappings.xls* on GitHub `https://goo.gl/gmGJey`.

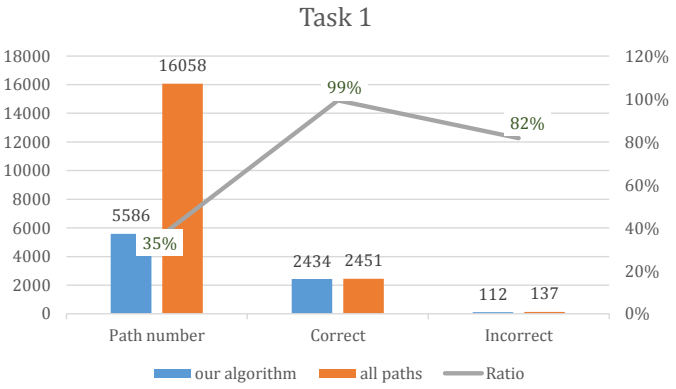
7.2. Mapping derivation algorithm

Fig. 12 and 13 show the results of our two derivation strategies on Task 1 and Task 2 of the OAEI LargeBio track. More particularly the figures show the number of returned paths, correct and incorrect candidate mappings resulted from the derivation step. In addition, we computed a ratio by dividing Algorithm 2 values by the All-paths values to compare the results of the two derivation algorithms.

As we can see, comparing to the All-paths derivation strategy, Algorithm 2 generates (i) much less paths (37%), (ii) almost the same number of correct candidate mappings (99%), and (iii) slightly less incorrect candidate mappings (82% and 93% in Task 1 and Task 2, respectively). We obtained similar results for the rest of LargeBio tasks and Anatomy.

The difference in the number of correct and incorrect mappings is explained by the fact that Algorithm 2 does not return paths that includes target ontology concepts as intermediate concepts, while the all path derivation using Neo4j does.

Figure 12: Task 1: derivation strategy comparison.



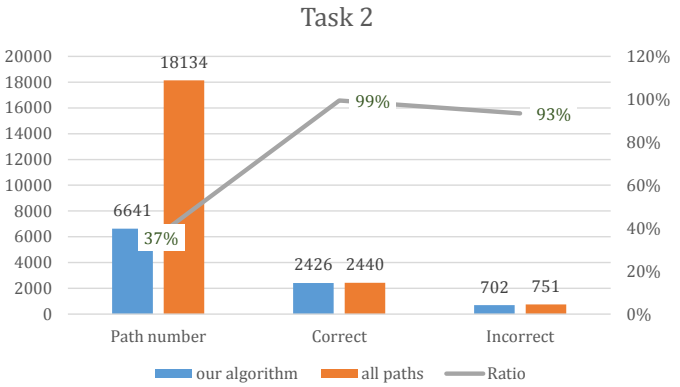
7.3. GBKOM with different direct matchers

To verify the effectiveness of GBKOM, we have performed experiments with different direct matchers namely:

Table 6: Example of mappings with subClassOf relation between mouse and NCI ontologies.

concept code	preferred label	concept code	preffered label
MA_0000061	arterial blood vessel	NCI_C12679	Blood Vessel
MA_0001871	right atrium valve	NCI_C12729	Cardiac Valve
MA_0000111	annulus fibrosus	NCI_C32599	Fibrocartilage
MA_0000554	thoracic cavity blood vessel	NCI_C12679	Blood Vessel
NCI_C53161	Hyoglossus Muscle	MA_0002296	extrinsic tongue muscle
NCI_C53174	Pronator Teres Muscle	MA_0000615	forelimb muscle
NCI_C53180	Transversus Thoracis	MA_0000548	chest muscle
NCI_C53180	Transversus Thoracis	MA_0000561	thorax muscle

Figure 13: Task 2 : derivation strategy comparison.



YAM++, LogMap and LogMapLite. Our choice is motivated by the different strategies implemented in these matchers, which almost cover all the strategies implemented in direct ontology matchers. Indeed, (i) YAM++ combines several similarity measures and rules, (ii) LogMap incorporates sophisticated reasoning and repair techniques, and (iii) LogMapLite, a lite matcher that essentially applies direct string matching techniques. Moreover, YAM++ and LogMap are state-of-the-art ontology matchers that had successfully participated in previous OAEI campaigns, and they do not implement a BK based approach as described in Section 3.

In Table. 7, 8 and 9, we present the original results of the three matchers and their results with GBKOM i.e, when each matcher is used as a direct matcher in GBKOM.

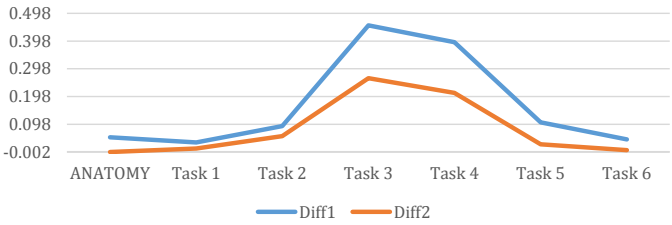
LogMap and LogMapLite. Comparing to the original results of LogMap, GBKOM shows slightly better results (F-measure) in almost all tasks, except in Task 2 because of a low precision. This low precision may be explained by the use of the preselected ontology UBERON. Indeed, NCI Thesaurus includes a small branch on mouse anatomy in addition to the human anatomy branch. Using the cross-references extracted from UBERON (considered as manual mappings), GBKOM returns mappings between human and mouse anatomy. However, UMLS, the source from which the reference alignment is extracted, is focused only on human health, and does not include mappings between the NCI mouse anatomy branch and MA (the Mouse Anatomy ontology); therefore, these mappings are considered as incorrect, which affects precision [15].

Note that LogMap exploits UMLS lexicon, a rich biomedical lexicon, as external knowledge resource, which reduces the benefit of using other external biomedical knowledge resources. Indeed, the improvement is more significant with LogMapLite, especially in Tasks 3 and 4 (see Table. 8).

As it was expected, the BK-based matching consumes more time than the direct matching. However, in many applications, the matching task is performed only once (e.g., integrating the data of two datasets structured using two different ontologies), and the most important thing is the quality of the generated alignment independently of the matching run time.

Fig. 14 shows two series: (1) diff1: the difference between the LogMap F-measure values and the LogMapLite F-measure values; (2) diff2: the difference between our framework results when using LogMap and LogMapLite. As we can see, in all matching tasks diff1 is less than diff2, which means that exploiting the BK allowed to reduce the gap between the two matchers (a sophisticated matcher and a simple matcher). However, this difference still exist which shows that the quality of GBKOM results depend on the quality of the direct matcher results.

Figure 14: F-measure difference: original results vs. our framework results.



YAM++. We participated in the OAEI 2017 [34] and OAEI 2017.5 [29] campaigns in Anatomy and Large-Bio tracks with YAM-BIO as a system, which is GBKOM with YAM++ as a direct matcher.

In OAEI 2017, we participated with a basic version of GBKOM that does not implement all the components described in Section 4. For more details about this participation, please refer to [34]. In OAEI 2017.5, we participated with the last version of GBKOM – the one described in this

Table 7: LogMap: original results vs. results with GBKOM.

TASK	Original results				Results with our framework			
Measure	Precision	Recall	F-measure	T (s)	Precision	Recall	F-measure	T (s)
Anatomy	0.918	0.846	0.880	4	0.900	0.947	**0.923**	42
Task 1	0.944	0.897	0.920	7	0.945	0.896	**0.920**	53
Task 2	0.856	0.808	**0.831**	53	0.763	0.851	0.804	174
Task 3	0.947	0.690	0.798	43	0.924	0.735	**0.819**	104
Task 4	0.840	0.645	0.730	302	0.798	0.695	**0.743**	465
Task 5	0.947	0.69	0.798	192	0.924	0.705	**0.800**	332
Task 6	0.868	0.597	0.707	622	0.795	0.683	**0.735**	923

Table 8: LogMapLite: original results vs. results with GBKOM.

TASK	Original results				Results with our framework			
Measure	Precision	Recall	F-measure	T (s)	Precision	Recall	F-measure	T (s)
Anatomy	0.962	0.728	0.829	1	0.929	0.921	**0.925**	24
Task 1	0.967	0.819	0.887	1	0.963	0.860	**0.909**	25
Task 2	0.673	0.820	0.739	7	0.674	0.841	**0.748**	59
Task 3	0.968	0.209	0.344	2	0.942	0.394	**0.555**	29
Task 4	0.852	0.209	0.336	12	0.822	0.393	**0.532**	92
Task 5	0.892	0.567	0.693	6	0.924	0.667	**0.774**	62
Task 6	0.797	0.567	0.663	12	0.818	0.658	**0.730**	116

paper (see Section 4) – with UBERON and DOID as preselected ontologies on the HOBBIT platform.[6] The Hobbit platform is a generic, modular and distributed platform for Big Linked Data systems. It was designed with the aim of providing an open-source, extensible, FAIR and scalable evaluation platform [29].

In Table. 9, we report the original results of YAM++, and its results with GBKOM in OAEI 2017.5 campaign.[7] We do not present the time required to obtain the original results because we do not have this information. Indeed, we did not run YAM++ on the Hobbit platform. Presenting the time obtained on another platform is not comparable to the one that is obtained on Hobbit.

As we can see, there is a significant improvement of the quality of generated alignments with GBKOM comparing to the original ones. More particularly, we observe a real increase in recall and a slight drop in precision.

OAEI 2017.5 campaign was aiming to test the Hobbit platform by the matching systems. Hence, most of participants such as AML or LogMap have reused the OAEI 2017 versions. Additionally, because of the technical constraints imposed by this evaluation platform such as the maximum computation time, some systems, such as LogMap-Bio, have not participated since it requires much time to select background knowledge ontologies from NCBO BioPortal. Therefore, we compare our OAEI 2017.5 results to the participant results in OAEI 2017.

With an F-measure of 0.929, YAM-BIO is the **third top-ranked** system in Anatomy (the best system had an F-measure of 0.943), and with an average F-measure of 0.832, YAM-BIO is the **top-ranked** system in LargeBio (see Table. 10).

Finally, we may observe that there is no ideal ontology matching strategy. Choosing a particular strategy (or the system that implements that strategy) depends on the user needs in terms of precision, recall and computation time. For instance, even if YAM-BIO has almost the same F-measure (0.763) as AML in Task 6, AML has a higher precision 0.904 – OAEI 2017 – vs. 0.842, while YAM-BIO has a higher recall 0.697 vs. 0.668.

Discussion on the OAEI evaluation. We believe, it would be interesting, when possible, to publish participant results with and without exploitation of specialized background knowledge resources. On one hand, it allows to better assess the benefit of exploiting background knowledge resources on the matching results and computation time. On the other hand, it enables a fair comparison with the systems that do not use background knowledge resources.

Some components are common in all ontology matcher architectures; others do not always exist — such as background knowledge resource selection or semantic verification. This makes the comparison of computation time particularly cumbersome and not always fair. We believe that it would be more appropriate to evaluate execution times for each separate component. For example, YAM-BIO used a predefined background knowledge while LogMap-Bio made a dynamic selection from an online repository necessarily taking additional time. Splitting running time by components will also help the community to identify

[6] `https://master.project-hobbit.eu/home`.

[7] Results may be consulted on the HOBBIT platform `https://goo.gl/A496ug`.

Table 9: YAM++: original results vs. results with GBKOM.

Task	Original results			Results with GBKOM (OAEI 2017.5)			
Measure	Precision	Recall	F-measure	Precision	Recall	F-measure	T(s)
Anatomy	0.967	0.744	0.841	0.946	0.913	**0.929**	176
Task1	0.906	0.864	0.884	0.971	0.902	**0.935**	197
Task2	0.828	0.859	0.843	0.818	0.894	**0.855**	518
Task3	0.646	0.713	0.678	0.962	0.741	**0.837**	244
Task4	0.658	0.707	0.682	0.879	0.738	**0.802**	755
Task5	0.875	0.641	0.740	0.927	0.703	**0.800**	478
Task6	0.839	0.623	0.715	0.842	0.697	**0.763**	962

Table 10: Comparison of GBKOM (YAM-BIO*) results in OAEI 2017.5 to average LargeBio results in OAEI 2017.

Measure	Precision	Recall	F-measure
AML	0.896	0.774	0.827
LogMap	**0.900**	0.721	0.797
LogMapBio	0.871	0.734	0.794
LogMapLite	0.858	0.532	0.610
Tool1	0.869	0.367	0.454
YAM-BIO	0.894	0.770	0.824
YAM-BIO*	**0.900**	**0.779**	**0.832**

less efficient components to improve them, and most efficient ones to reuse them.

8. Conclusion

In this paper, we have presented a generic framework for BK-based Ontology Matching, called GBKOM, which offers a great flexibility in different aspects: (i) the modules could be used independently and (ii) several parameters are provided and may be easily customized according to the user needs. The framework can be used with any direct matcher, and the BK selection could be used separately from the BK exploitation. Indeed, the BK selection module generates two files: (i) an OWL file containing all the selected concepts with their labels; it may be seen as a fictional ontology that is created to enable anchoring to the target ontology by matching systems, and (ii) a CSV file containing all the filtered mappings in the following format (URI source, URI ontology source, URI target, URI ontology target, score, relation, manualMapping). ManualMapping is a boolean property that takes "true" or "false as value". Therefore, a user willing to test its new method of BK exploitation, she/he could reuse the result of the BK selection (both files) without going through the exploitation step of GBKOM. If she/he is only interested by the BK selection module without worrying about the exploitation, it is enough to generate these two files as inputs for the module BK exploitation.

GBKOM is publicly available to the community in a GitHub project.[8] It has been evaluated in two OAEI campaigns. Indeed, we have used YAM-BIO – GBKOM with YAM++ – matcher to participate in the OAEI 2017 and OAEI 2017.5 campaigns in two tracks: Anatomy and LargeBio. The results obtained in those tracks were very close to top-ranked state-of-the-art systems, thanks to the different content matching techniques implemented in YAM++, the BK used and our BK selection and exploitation methods. Furthermore, we performed experiments with two other matchers LogMap, and LogMapLite. In both cases, GBKOM improved the original alignments generated by these matchers. This shows that the effectiveness of GBKOM is independent of the direct matcher used.

[8] `https://github.com/AminaANNANE/GenericBKbasedMatcher`.

9. Acknowledgment

The authors wish to acknowledge the Eiffel Excellence Scholarship program, University of Montpellier (France), and Ecole nationale Supérieure d'Informatique (Algeria).

References

[1] J. Euzenat, P. Shvaiko, Ontology Matching (second edition), Springer, 2013 (2013).

[2] Z. Aleksovski, M. Klein, W. Ten Kate, F. Van Harmelen, Matching unstructured vocabularies using a background ontology, in: 15th International Conference on Knowledge Engineering and Knowledge Management, EKAW, Podebrady, Czech Republic, 2006, pp. 182–197 (2006).

[3] C. Pesquita, D. Faria, E. Santos, F. M. Couto, To repair or not to repair: reconciling correctness and coherence in ontology reference alignments, in: 8th International Workshop on Ontology Matching, OM, Sydney, Australia, 2013, pp. 13–24 (2013).

[4] A. Locoro, J. David, J. Euzenat, Context-based matching: design of a flexible framework and experiment, Journal on Data Semantics 3 (1) (2014) 25–46 (2014).

[5] Z. Aleksovski, W. Ten Kate, F. Van Harmelen, Exploiting the structure of background knowledge used in ontology matching, in: 1st International Workshop on Ontology Matching, OM, Athens, Georgia, USA, 2006, pp. 13–24 (2006).

[6] M. Sabou, M. d'Aquin, E. Motta, Exploring the semantic web as background knowledge for ontology matching, Journal on Data Semantics (2008) 156–190 (2008).

[7] A. Annane, Z. Bellahsene, F. Azouaou, C. Jonquet, Building an effective and efficient background knowledge resource to enhance ontology matching, Journal of Web Semantics 51 (2018) 51 – 68 (2018). doi:https://doi.org/10.1016/j.websem.2018.04.001.
URL `http://www.sciencedirect.com/science/article/pii/S1570826818300179`

[8] D. Ngo, Z. Bellahsene, Overview of YAM++:(not) Yet Another Matcher for ontology alignment task, Journal of Web Semantics 41 (2016) 30 – 49 (2016).
[9] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007 (2007). doi:10.1007/978-3-540-49612-0.
[10] M. Cheatham, P. Hitzler, String similarity metrics for ontology alignment, in: 12th International Semantic Web Conference, ISWC, Sydney, Australia, 2013, pp. 294–309 (2013).
[11] D. Ngo, Z. Bellahsene, K. Todorov, Opening the black box of ontology matching, in: 10th Extended Semantic Web Conference, ESWC, Montpellier, France, 2013, pp. 16–30 (2013).
[12] T. Pedersen, S. Patwardhan, J. Michelizzi, Wordnet: : Similarity - measuring the relatedness of concepts, in: 19th National Conference on Artificial Intelligence, San Jose, California, USA, 2004, pp. 1024–1025 (2004).
[13] F. Duchateau, Z. Bellahsene, YAM: A step forward for generating a dedicated schema matcher, Transactions on Large-Scale Data and Knowledge-Centered Systems XXV 25 (2016) 150–185 (2016).
[14] R. Amini, Towards best practices for crowdsourcing ontology alignment benchmarks, Ph.D. thesis, Wright State University (2016).
[15] D. Faria, C. Pesquita, E. Santos, I. F. Cruz, F. M. Couto, Automatic background knowledge selection for matching biomedical ontologies, PloS One 9 (11) (2014) e111226 (2014).
[16] M. Hartung, A. Groß, T. Kirsten, E. Rahm, Effective mapping composition for biomedical ontologies, in: 9th Extended Semantic Web Conference, ESWC, Heraklion, Crete, Greece, 2012, pp. 176–190 (2012).
[17] X. Chen, W. Xia, E. Jiménez-Ruiz, V. Cross, Extending an ontology alignment system with BioPortal: a preliminary analysis, in: 13th International Semantic Web Conference, ISWC, Posters and Demonstrations, Riva del Garda, Italy, 2014, pp. 313–316 (2014).
[18] A. Annane, Using Background Knowledge to Enhance Biomedical Ontology Matching, Theses, Université Montpellier ; Ecole Nationale Supérieure d'Informatique (ESI) (Oct. 2018).
URL https://tel.archives-ouvertes.fr/tel-02092875
[19] A. Groß, M. Hartung, T. Kirsten, E. Rahm, Mapping composition for matching large life science ontologies, in: 2nd International Conference on Biomedical Ontology, ICBO, Buffalo, NY, USA, 2011, pp. 109–116 (2011).
[20] O. Bodenreider, The Unified Medical Language System (UMLS): integrating biomedical terminology, Nucleic Acids Research 32 (2004) 267–270 (2004).
[21] C. J. Mungall, C. Torniai, G. V. Gkoutos, S. E. Lewis, M. A. Haendel, Uberon, an integrative multi-species anatomy ontology, Genome biology 13 (1) (2012) 1–20 (2012).
[22] C. Rosse, J. L. Mejino, A reference ontology for biomedical informatics: the foundational model of anatomy, Journal of Biomedical Informatics 36 (6) (2003) 478 – 500 (2003).
[23] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, F. M. Couto, The agreementmakerlight ontology matching system, in: On the Move to Meaningful Internet Systems, OTM, Graz, Austria, 2013, pp. 527–541 (2013).
[24] E. Jiménez-Ruiz, B. C. Grau, V. Cross, LogMap family participation in the OAEI 2016, in: 11th International Workshop on Ontology Matching, OM, Kobe, Japan, 2016, pp. 185–189 (2016).
[25] C. Quix, P. Roy, D. Kensche, Automatic selection of background knowledge for ontology matching, in: International Workshop on Semantic Web Information Management, SWIM, Athens, Greece, 2011, pp. 5:1–5:7 (2011).
[26] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, et al., The obo foundry: coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology 25 (11) (2007) 1251–1255 (2007).
[27] J. David, J. Euzenat, F. Scharffe, C. Trojahn dos Santos, The Alignment API 4.0, Semantic Web Journal 2 (1) (2011) 3–10 (2011).
[28] E. Jiménez-Ruiz, C. Meilicke, B. C. Grau, I. Horrocks, Evaluating mapping repair systems with large biomedical ontologies, in: 26th International Workshop on Description Logics, Ulm, Germany, 2013, pp. 246–257 (2013).
[29] E. e. a. Jiménez-Ruiz, Introducing the HOBBIT platform into the ontology alignment evaluation campaign, in: 13th International Workshop on Ontology Matching, (OM), Monterey, CA, USA, 2018, pp. 49–60 (2018).
[30] N. Sioutos, S. de Coronado, M. W. Haber, F. W. Hartel, W.-L. Shaiu, L. W. Wright, NCI thesaurus: A semantic model integrating cancer-related clinical and molecular information, Journal of Biomedical Informatics 40 (1) (2007) 30 – 43 (2007).
[31] K. Donnelly, SNOMED-CT: The advanced terminology and coding system for eHealth, Studies in health technology and informatics 121 (2006) 279 (2006).
[32] M. Cheatham, Z. Dragisic, J. Euzenat, D. Faria, A. Ferrara, G. Flouris, I. Fundulaki, R. Granada, V. Ivanova, E. Jiménez-Ruiz, Results of the ontology alignment evaluation initiative 2015, in: 10th International Workshop on Ontology Matching, OM, Bethlehem, USA, 2015, pp. 60–115 (2015).
[33] Z. Dragisic, V. Ivanova, H. Li, P. Lambrix, Experiences from the anatomy track in the ontology alignment evaluation initiative, Journal of Biomedical Semantics 8 (1) (2017) 56:1–56:28 (2017).
[34] A. Annane, Z. Bellahsene, F. Azouaou, C. Jonquet, YAM-BIO: results for OAEI 2017, in: 12th International Workshop on Ontology Matching, OM, Vienna, Austria, 2017, pp. 201–206 (2017).