# AgreementMakerLight

Daniel Faria [a,*], Emanuel Santos [b], Booma Sowkarthiga Balasubramani [c], Marta C. Silva [a], Francisco M. Couto [a] and Catia Pesquita [a]

[a] *LASIGE, Faculdade de Ciências da Universidade de Lisboa, Portugal*
[b] *STEM Department, Concordia International School Hanoi, Vietnam*
[c] *University of Illinois at Chicago, USA*

**Abstract.** Ontology matching establishes mappings between entities of related ontologies, with applications ranging from enabling semantic interoperability to supporting ontology and knowledge graph development. Its demand within the Semantic Web community is on the rise, as the popularity of knowledge graph supporting information systems or artificial intelligence applications continues to increase.

In this article, we showcase AgreementMakerLight (AML), an ontology matching system in continuous development since 2013, with demonstrated performance over nine editions of the Ontology Alignment Evaluation Initiative (OAEI), and a history of real-world applications across a variety of domains. We overview AML's architecture and algorithms, its user interfaces and functionalities, its performance, and its impact.

AML has participated in more OAEI tracks since 2013 than any other matching system, has a median rank by F-measure between 1 and 2 across all tracks in every year since 2014, and a rank by run time between 3 and 4. Thus, it offers a combination of range, quality and efficiency that few matching systems can rival. Moreover, AML's impact can be gauged by the 263 (non-self) publications that cite one or more of its papers, among which we count 34 real-world applications.

Keywords: Ontology Matching, Instance Matching, Tool

## 1. Introduction

Ontologies are a cornerstone of the Semantic Web, serving as knowledge models for describing data on the web, namely under knowledge graphs [1]. They are also critical for the realization of the FAIR data principles [2] as they provide the means for enabling findability, interoperability and reusability of (meta)data. Yet, ontologies represent particular points of view of their domains, reflecting the concrete application that motivated their development. Thus, distinct ontologies may overlap in domain but differ substantially in how they model that domain, with respect to terminology, granularity and/or organization. When related datasets are described using distinct but overlapping ontologies, we have a semantic interoperability problem, which can be addressed by establishing mappings between the ontologies [3].

Ontology matching (or alignment) is the process of finding mappings between entities of ontologies that overlap in domain [3]. This is essential to a full realization of the Semantic Web vision, as it links entities from different ontologies while still maintaining the distributed nature of semantic resources. It is also highly relevant for ontology development, as it is facilitates the common practice of creating explicit mappings to existing ontologies in the form of cross-references, as well as the reuse and integration of existing ontologies.

Ontology matching can be performed (semi-)automatically through the use of ontology matching systems that identify potential mappings based on the various features of ontology entities. The annual Ontology Alignment Eval-

*Corresponding author. E-mail: dpfaria@fc.ul.pt.

uation Initiative (OAEI) [4] serves as a testing ground for these systems, by providing a wide range of benchmarks and homogeneous testing conditions.

In this article, we showcase AgreementMakerLight (AML), one of the most successful ontology matching systems with respect to performance in the OAEI as well as use in real-world applications. Originally released in 2013 [5], AML has been in continuous development since, and features numerous new functionalities.

The rest of the article is organized as follows: Section 2 details the key concepts in ontology matching; Section 3 overviews AML's architecture, algorithms and user interface; Section 4 reviews its evaluation results; Section 5 compiles its impact and its applications to real-world ontology matching problems; and Section 6 concludes the article with our perspective on the present and future of AML.

## 2. Problem Definition

In its traditional form, the ontology matching process takes as input two ontologies—*source* and *target*—and outputs a set of mappings between their entities, called an alignment [3]. A mapping establishes a directional link or correspondence between two ontology entities, typically in the form of a 4-tuple $< e_s, e_t, r, c >$, where $e_s$ and $e_t$ are the *source* and *target* entities, $r$ is the semantic relation between them, and $c$ represents an optional confidence score that usually reflects a measure of similarity between the entities. Variants of the ontology matching process include complex matching [6, 7], where mappings can feature class and property expressions and other semantic constructs instead of ontology entities, and holistic matching [8], where more than two input ontologies are matched, but both are beyond the scope of this work.

We can distinguish three main sub-categories of ontology matching: schema matching, where the entities to match are classes and/or properties; instance matching, where the entities to match are ontology individuals; and instance-to-schema matching where the goal is to match instance-level data to an ontology schema, namely by assigning individuals to classes. Moreover, we can distinguish between automatic ontology matching, where the process is carried out by an ontology matching system without user intervention (though manual validation of the resulting alignment can still take place); interactive matching, where user input during the matching process contributes to the configuration or decisions of the ontology matching system; and, evidently, manual matching, where no ontology matching system is involved.

Ontology matching systems typically employ three types of algorithms: pre-processing algorithms, which load the input ontologies and extract and process the information that will be used to match them (e.g. by normalizing labels and synonyms); matching algorithms (or matchers) which exploit the information of the ontologies to generate candidate mappings between them, resulting in preliminary alignments; and filtering algorithms (or filters) which remove mapping candidates from preliminary alignments according to predefined criteria, to increase the quality of the final alignment. In some systems, matching and filtering are meshed together.

Matchers can be classified according to the ontology features they rely on: terminological matchers rely on lexical information (i.e. annotations such as labels and synonyms); structural matchers rely on the relations between entities; semantic matchers interpret the semantics of the ontologies, usually using a reasoner; and data matchers rely on attributive information (i.e. the data values that characterize individuals).

Filters can be classified according to the principles that guide them, including: similarity, cardinality, coherence, conservativity, and locality. Similarity filters simply filter mappings below a given confidence score threshold. Cardinality filters aim to ensure or approximate a maximum cardinality in the alignment (i.e., a maximum number of mappings per entity), generally 1-to-1. Coherence filters, or alignment repair algorithms, remove mappings that cause unsatisfiabilities or incoherences when the ontologies are considered together with the alignment [9]. Conservativity filters exclude mappings that cause new logical inferences within either of the matched ontologies when they are considered together with the alignment, such as two classes being inferred as equivalent if they are mapped to the same class of the other ontology [10]. Thus, they encompass strict 1-to-1 cardinality filtering. Finally, locality filters are predicated on the assumption that mappings should be co-located within the structure of the ontologies, and can be either high level if they look to the top branches or blocks of the ontology, or low level if they look to the direct vicinity of mappings [10]. High level locality filtering, or blocking, can be performed *a priori* (i.e., before full matching) in the case of very large ontologies, to reduce the dimension of the matching space.

Ontology alignments can be stored independently from the ontologies or incorporated into them, and can have varying degrees of semantic enforcing. The *de facto* standard is to encode them as external files using the the RDF Alignment format[1], but they can also be encoded in OWL either semantically (e.g. with equivalent class or subclass axioms) or no semantics (e.g. with annotations such as cross-references).

## 3. AgreementMakerLight

AML is an ontology matching system that can perform schema and/or instance matching either automatically or interactively. It can also be used to perform repair of an existing ontology alignment as well as for alignment validation.

AML is programmed in Java and developed in Eclipse. It is open source and available through GitHub[2] under the Apache License 2.0. Its latest release, AML v3.2, is available as a zipped folder containing a runnable JAR file featuring both a command line interface and a graphical user interface. Additionally, users can also download and compile the source code.

### *3.1. Core Architecture*

AML comprises four main modules: data, pre-processing, matching, and filtering. Its architecture is schematized in Figure 1.

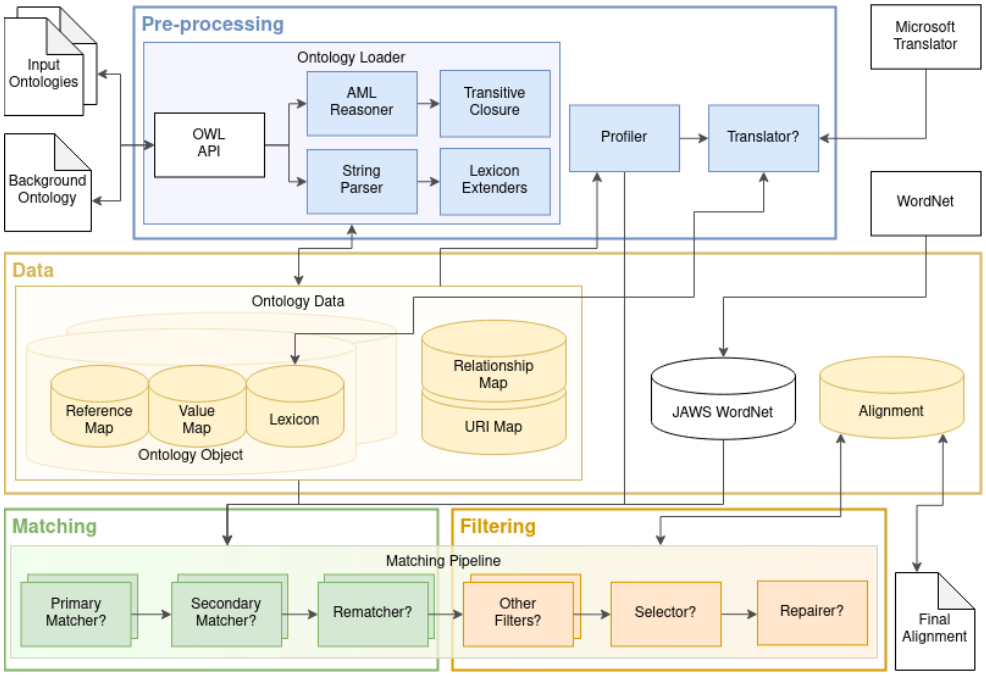


Fig. 1. AML Architecture. The pre-processing module handles the loading of ontologies and extracting all the relevant information to populate AML's data structures. It also handles the profiling of the matching task, which configures which matching and filtering algorithms will compose AML's matching pipeline, as well as their settings. Question marks indicate conditional steps.

[1] https://moex.gitlabpages.inria.fr/alignapi/format.html
[2] https://github.com/AgreementMakerLight/AML-Project

#### 3.1.1. Data

The data module comprises AML's data structures for representing ontology and alignment information. These are key-value tabular data structures geared to enable $O(n)$ matching, by making use of inverted indexing [11]. The chief data structures in AML are the following:

- The URIMap stores the URIs and corresponding integer indexes assigned by AML to all entities of the two input ontologies.
- The *RelationshipMap* encompasses all semantic relations between the entities of the two input ontologies, including subclass and equivalence relations between classes (with transitive closure), disjoint classes, property domains and ranges, subproperty and equivalent property relations between properties, instance relations between individuals and classes, and relations between individuals.
- The *Lexicon* contains the lexical data of an ontology by language, including the local name of entities (when it is not an alphanumeric code), annotations of entities with lexical annotation properties (e.g. *label*, *hasExactSynonym*), and data values of individuals for data properties that have a label ending on *name* or *title*. All lexical entries are assigned a weight which reflects the precision of the property (e.g. labels are considered more precise than exact synonyms and these are more reliable than other synonyms) [12]. Each ontology has its own instance of Lexicon, including background knowledge ontologies.
- The *ValueMap* holds all attributes (i.e. data properties and their data values) for the individuals of an ontology, other than those placed in the Lexicon (i.e. *name* and *title* properties). Unlike the Lexicon, entries in the ValueMap are not normalized. Each ontology has its own instance of ValueMap.
- The *ReferenceMap* stores cross references and logical definitions of classes [13], which are common in biomedical ontologies. Each ontology has its own instance of ReferenceMap, including background knowledge ontologies.
- The *Alignment* represents an ontology alignment, storing all mappings between classes in both a list (to enable sorting) and a key-valued table (to enable efficient search and reasoning). Each of AML's matching algorithms will produce an instance of Alignment, and these can be combined through standard set operations—union, intersection and difference—as well as by adding only non-conflicting mappings (i.e. mappings where neither entity is already mapped) or only better mappings (i.e. mappings that have a higher confidence score than existing mappings for the entities that are already mapped).

#### 3.1.2. Pre-processing

The pre-processing module handles the loading of the input ontologies into memory, their profiling to configure the matching problem, and when necessary, their translation.

Ontology loading is mediated by the OWL API [14], which can read OWL ontologies in all official serializations, constructing an object-oriented ontology representation in Java. AML extracts from the latter all relevant information for ontology matching, populating its data structures. With respect to lexical information, all entries in the *Lexicon* except those detected as formulas (i.e. non-Latin-alphabet-word-based entries) are normalized by removing most non-word non-number characters, removing all diacritics, introducing spaces when case-changes or non-word characters are used as word separators, and converting all characters to lower case.

After loading, the transitive closure of class relations in the *RelationshipMap* is computed via the Semi-Naive algorithm [15], and two *lexicon extender* algorithms are applied to enrich the *Lexicons* of the two ontologies by generating synonyms for all lexical through the removal of all stop words (listed in AML's StopList.txt file) and, separately, of all substrings within parenthesis.

The matching problem is then profiled by computing parameters that enable AML to decide on the matching strategy to use, including whether the ontologies need to be translated, which type(s) of entities should be matched and which matching algorithms to employ, and whether to match individuals only of the same class. The profiling parameters computed include: the number of entities in the ontologies, the proportion of entities of each type (classes, properties and individuals), the language(s) of their lexical entries, and classes in common between the input ontologies (in the case of instance matching problems). Default matching settings can be overridden either by declaring them in AML's config.ini file or when running AML's manual matcher via the graphical user interface.

Finally, when the ontologies to match do not have significant overlap between their language(s), AML performs automatic translation using Microsoft® Translator. The translation is done for each ontology, by querying

Microsoft® Translator for each lexical entry (the full entry, rather than word-by-word) and translating it both into the primary language of the other ontology and into English. To improve performance, AML stores locally all translation results in dictionary files, and queries the Translator only when no stored translation is found. Users that wish to make use of this feature of AML are required to register their own instance of Microsoft® Translator and place a file with their authentication token in AML's store/ folder.

#### *3.1.3. Matching*

The matching module encompasses AML's matchers, which leverage the data in AML's data structures from the input ontologies and/or external knowledge sources, to produce mappings between ontology entities. Each AML matcher can be classified in three dimensions: its matching strategy, the entity type(s) which it can match; and the matching mode(s) it supports. Additionally, matchers can be self-contained or be ensembles of other matchers.

With respect to matching strategy, matchers can be divided into pairwise and hash-based, with the former requiring an explicit pairwise comparison of the entities of the two ontologies and therefore having $O(n^2)$ time complexity, and the latter relying on hash maps with inverted indices to enable entity lookup across ontologies and therefore having $O(n)$ time complexity.

Concerning entity types, matchers can apply to classes and/or properties and/or individuals, with each matcher storing the entity types it can match, and requiring a choice of entity type when called to match two ontologies. Some matchers are exclusively for a single type while others are transversal.

Matchers can implement one or more of the following matching modes: *primary matcher*, *secondary matcher*, *rematcher*, and *lexicon extender*. In *primary matcher* mode, the matcher performs a complete match, assessing all entities of the specified type of the two ontologies. In *secondary matcher* mode, an input alignment is required, and only entities (of the specified type) that are not mapped in that alignment will be matched. Some matchers match all non-mapped entities in this mode, whereas others match only entity pairs in the vicinity of the mappings in the input alignment, which improves both efficiency and precision. In *rematcher* mode, an input alignment is also required, but the matcher will only match the entities (of the specified type) that are already mapped in that alignment, by computing new confidence scores for the mapping according to the metric employed by the matcher. This mode is useful for refining mappings, namely in two-stage matching algorithms where a first efficient global search determines the pairs of entities to match, then a more computationally-intensive algorithm is used to calculate the similarity of each of pair. In *lexicon extender* mode, instead of matching the ontologies, the algorithm extends their lexicons with additional synonyms, which can then be used by other matchers to match the ontologies. This mode is available for matchers involving background knowledge as a scalable way to enable inexact matching with string or word similarity algorithms: rather than use these algorithms to compare each input ontology with the background knowledge source, if the background knowledge source is used to enrich the input ontologies, the algorithms can be applied only between them, reducing the computational cost. Note that lexicon extenders are only considering as part of the matching stage when they involve matching the input ontologies against background knowledge sources; lexicon extenders that rely only on internal ontology information are considered part of the pre-processing stage (wherein they were listed).

The matchers currently available in AML are summarized in Table 1.

#### *3.1.4. Filtering*

AML's filtering module encompasses its filters, which are responsible for producing a final high-quality alignment from the preliminary alignment produced by the matching module. AML includes two modes for filtering: *filterer* and *flagger*. The *filterer* mode is aimed at automatic use, as the algorithms automatically classify mappings as incorrect, whereas the *flagger* mode is intended for manual use, as the problematic mappings are 'flagged' which highlights them in the user interface for users to assess their correctness.

AML's filters and their principles are listed in Table 2.

### *3.2. User Interfaces & Functionalities*

AML features both a command line interface (CLI) and a graphical user interface (GUI) [19], with the latter being called when AML is executed with no arguments, and the former being called when command line arguments are given.

Table 1
Overview of the matchers available in AML

| Matcher | Strategy | Modes | Types | Data Structures | Principle |
|---|---|---|---|---|---|
| LexicalMatcher | Hash | P | All | Lexicon | String equivalence |
| SpacelessLexicalMatcher | Hash | P | All | Lexicon | String equivalence ignoring spaces |
| WordMatcher | Hash | PSR | All | Lexicon | String similarity based on word overlap |
| StringMatcher | Pairwise | PSR | All | Lexicon | String similarity with the ISub metric [16] |
| DirectXRefMatcher | Hash | P | Class | Reference Map | Direct cross-references between the ontologies or shared cross-references or logical definitions |
| MediatingXRefMatcher | Hash | PL | Class | Reference Map | Direct cross-references from a background knowledge ontology |
| MediatingMatcher | Hash | PL | Class | Lexicon | String equivalence using a background knowledge ontology as mediator |
| WordNetMatcher | Hash | PL | All | Lexicon | String equivalence using WordNet synonyms of full Lexicon entries |
| BackgroundKnowledgeMatcher | Hash | P | Class | Lexicon, Reference Map | Ensemble of three preceding matchers that tests all available knowledge sources [17] |
| ThesaurusMatcher | Hash | P | All | Lexicon | String equivalence using synonyms inferred from lexical analysis [12] |
| AcronymMatcher | Pairwise | P | All | Lexicon | String equivalence using acronyms |
| HybridStringMatcher | Pairwise | PR | All | Lexicon | String similarity through a hybrid approach combining word overlap with WordNet synonyms and the ISub metric |
| MultiWordMatcher | Pairwise | P | All | Lexicon | WordNet similarity between two-word names where one word is shared and the other is related through WordNet |
| NeighborSimilarityMatcher | Pairwise | SR | Class | Relationship Map | Structural similarity based on mapped subclasses and/or superclasses |
| BlockRematcher | Pairwise | R | Class | Relationship Map | Structural similarity based on overlap between blocks (high-level divisions) of the ontologies |
| InstanceBasedClassMatcher | Hash | P | Class | Relationship Map | Structural similarity based on fraction of shared instances |
| ValueMatcher | Hash | PS | Instance | Value Map | String equivalence between values for the same property or matching properties |
| ValueStringMatcher | Pairwise | PS | Instance | Value Map | Hybrid string similarity between values for the same property or matching properties |
| Value2LexiconMatcher | Pairwise | PS | Instance | Value Map, Lexicon | Hybrid string similarity between property values and lexicon entries |
| ProcessMatcher | Pairwise | P | Instance | Relationship Map, Lexicon | Combines hybrid string similarity with similarity flooding across individual relations |

Modes: P = Primary Matcher; S = Secondary Matcher; R = Rematcher; L = Lexicon Extender

The CLI is intended for automatic use of AML, either to match two ontologies or to repair an existing ontology alignment, with command line options detailed in a README file. Match settings can be configured through a config.ini file.

The GUI supports automatic or custom ontology matching, automatic filtering (including repair) of an input alignment, and manual alignment validation. To support the latter, AML provides two views of the alignment: a primary list view of the whole alignment, where mappings can be classified by the user with color-coding, as displayed in Figure 2; and secondary a mapping panel with a local graph view of the alignment and information about the mapped classes, including conflicting mappings, which can be accessed by clicking on a mapping. AML's filters can be run in flag mode, which highlights the problem-causing mappings in orange, deferring to the user the decision on which to remove. Alignments can be saved in either RDF and TSV, including information on the classification of which mapping, which allows alignment validation to be carried out over multiple sessions.

Table 2
Filters available in AML

| Filter | Principle | Types | Description |
|---|---|---|---|
| Selector | Cardinality | All | Aims to produce a (near) 1-to-1 alignment with different permissivity settings |
| CardinalitySelector | Cardinality | All | Aims to produce a (near) n-to-n alignment where n is specified as an argument |
| Repairer | Coherence | Class | Filters mappings that would cause unsatisfiable classes [18] |
| DomainAndRangeFilterer | Coherence | Property | Filters mappings between properties that do not have compatible domain and range declarations |
| ObsoleteFilterer | Deprecation | Class | Filters mappings involving obsolete/deprecated classes |

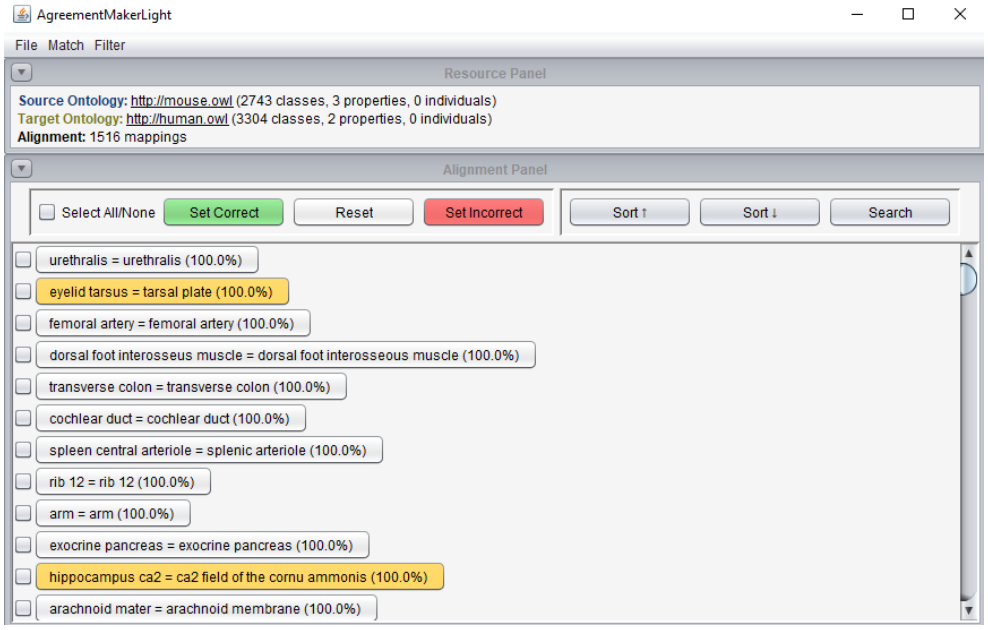


Fig. 2. AML's graphical user interface displaying the alignment panel with alignment validation functionalities. Mappings highlighted in orange were flagged automatically by AML's filters.

## 4. Performance

AML's performance in ontology matching can be assessed through its results in the OAEI, where it has entered in all editions since 2013.

Concerning range, AML was the ontology matching system that participated in the most total OAEI tracks since 2013 (83), as well as by the variety of types and domains spanned by these, as summarized in Table 3. Moreover, AML's range was extended over the years, with the addition of instance matching capabilities in 2016 marking the greatest upgrade, as shown in Table 4.

With respect to the quality of its results, AML had a median rank by F-measure across all OAEI tasks it participated in between 1 and 2 since 2014. Concerning efficiency, it had a median rank by run time between 3 and 4 in all OAEI editions. These results highlight that AML is an all-purpose ontology matching system, capable of tackling a variety of tasks efficiently and with high quality. Among OAEI participants, only LogMap[20] has a comparable global performance.

Table 3
Recurring OAEI tracks AML participated in, classified by type and domain

| Track | Type | Domain | Participations |
|---|---|---|---|
| Anatomy | Schema Matching | Biomedical | 2013-2021 (9) |
| Large Biomedical Ontologies | Schema Matching | Biomedical | 2013-2021 (9) |
| Disease & Phenotype | Schema Matching | Biomedical | 2016-2021 (6) |
| Biodiversity & Ecology | Schema Matching | Biomedical | 2018-2021 (4) |
| Conference | Schema Matching | Conferences | 2013-2021 (9) |
| Interactive Matching | Interactive Schema Matching | Multiple | 2013-2021 (9) |
| Multifarm | Multilingual Schema Matching | Conferences | 2013-2021 (9) |
| Library | Multilingual Thesauri Matching | Libraries | 2013-2014 (2) |
| Benchmark | Schema Matching | Multiple | 2013-2016 (4) |
| Process Model | Instance Matching | Process Models | 2016-2017 (2) |
| DOREMUS | Instance Matching | Music | 2016-2017 (2) |
| SPIMBENCH | Instance Matching | Creative works | 2017-2021 (5) |
| Link Discovery | Link Discovery & Spatial Matching | Path coordinates | 2017-2021 (5) |
| Knowldge Graph | Schema & Instance Matching | Fandom wikis | 2018-2021 (4) |

Table 4
AML's OAEI participation and results across the years

| | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 |
|---|---|---|---|---|---|---|---|---|---|
| Total tracks | 7/8 | 7/8 | 6/7 | 9/9 | 9/9 | 11/11 | 10/11 | 10/12 | 10/13 |
| Total matching tasks | 12 | 14 | 12 | 25 | 20 | 22 | 21 | 21 | 19 |
| New tasks | - | 2 | 1 | 14 | 4 | 6 | 0 | 2 | 0 |
| Median F-measure | 0.73 | 0.75 | 0.71 | 0.74 | 0.83 | 0.76 | 0.73 | 0.78 | 0.78 |
| Median Rank by F-measure | 4 | 1 | 1 | 2 | 1 | 1.5 | 1 | 1.5 | 1 |
| Median Rank by run time | 3 | 3.5 | 3.5 | 3 | 3 | 4 | 4 | 4 | 4 |

## 5. Impact and Use

AML has also been one of the ontology matching systems with the greatest impact in research and beyond, as evidenced by the number of citations of its previous publications as well as by the several applications for which it was used.

We classified the 263 unique publications that cite one or more of AML's previous publications[3] in three dimensions: by type of publication, by type of citation, and by application domain. With respect to type of publication, we count 208 research articles, 40 theses, 7 review articles, 7 OAEI system papers and 1 book chapter. Regarding the type of citation, 123 publications merely mention AML, usually as part of the state of the art in ontology matching, 102 publications use AML as a benchmark against which a proposed algorithm is compared, 4 publications propose extensions to AML (external to the AML team) and 34 publications apply AML in real-world ontology matching problems. While 141 publications are agnostic, 122 focus on a specific application domain, with the distribution displayed in Figure 3.

Many of the application domains under which AML is cited or applied reflect its successes in the OAEI, including the two most frequent domains, the life & health sciences and translation & cross-lingual matching, but also Business Processes and the Geospatial domain. Others, however, are domains in which we had never tested AML, including the Internet of Things (IoT) & Smart Cities, Humanities & Society, Information Technology (IT) & Electronics, and Air Traffic Management (ATM) & Aeronautics.

[3]Citations were collected through Google Scholar on April 14th 2022, with self-citations excluded

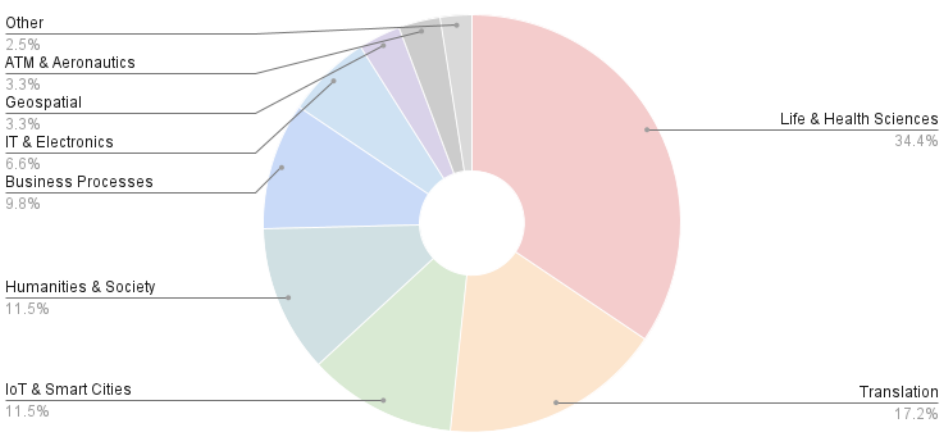


Fig. 3. Domain distribution of publications using AML in real-world applications

Among real-world problems to which AML was applied, we highlight the integration of agricultural thesauri under the Global Agricultural Concept Space for the Food and Agricultural Organization of the United Nations (FAO) [21], the creation of a knowledge graph for ecotoxicology risk assessment [22], and the alignment of ATM ontologies from the Single European Sky ATM Research (SESAR) and the National Aeronautics and Space Administration (NASA) [23].

Last but not least, further evidence of the impact and use of AML comes from the statistics of its GitHub repository, which was forked 31 times and starred 38 times[4].

## 6. Conclusions and Future Work

AML is one of the most successful ontology matching systems, both in terms of its results in the OAEI and in terms of its impact and real-world applications. It evolved from a matching system focusing mainly on the biomedical domain to an all-purpose system with wide applicability, thanks to both a continuous development effort and a core architecture designed with extensibility in mind.

The future development of AML will contemplate new challenges arising in ontology matching, as we endeavor to keep it on the forefront of the field.

As the Semantic Web evolves, we are witnessing the proliferation of huge knowledge graphs supporting online information systems as well as AI applications, which will increase demand for ontology matching, both to support the construction of knowledge graphs from existing data and ontologies, and to enable interoperability between information systems relying on related knowledge graphs. These applications place additional emphasis on scalability, and may require taking ontology matching beyond the pairwise paradigm, and into holistic matching.

But by far the greatest standing challenge in ontology matching is complex matching, which requires identifying complex semantic relations between ontology entities, involving expressions such as property restrictions. Such mappings offer a solution to the problem of reconciling logical coherence with alignment completeness in ontology matching, yet detecting them automatically with reasonable accuracy is still beyond the state of the art, as our preliminary efforts have revealed [24].

## 7. Acknowledgements

We dedicate this article to the memory of Prof. Isabel Cruz, without whom AML would not have existed. We also acknowledge the many students and collaborators who contributed to AML throughout the years.

[4]Statistics collected on April 14th 2022

This work was supported by: FCT through the LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020) and through project DeST: Deep Semantic Tagger (PTDC/CCI-BIO/28685/2017); the KATY project, funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 101017453; NSF award III-1618126; and NIGMS-NIH award R01GM125943.

## References

[1] L. Ehrlinger and W. Wöß, Towards a definition of knowledge graphs., *SEMANTiCS (Posters, Demos, SuCCESS)* **48**(1–4) (2016), 2.
[2] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3**(1) (2016), 1–9.
[3] J. Euzenat and P. Shvaiko, *Ontology matching*, 2nd edn, Springer-Verlag, Heidelberg (DE), 2013.
[4] N. Pour, A. Algergawy, R. Amini, D. Faria, I. Fundulaki, I. Harrow, S. Hertling, E. Jiménez-Ruiz, C. Jonquet, N. Karam et al., Results of the ontology alignment evaluation initiative 2020, in: *Proceedings of the 15th International Workshop on Ontology Matching (OM 2020)*, Vol. 2788, CEUR-WS, 2020, pp. 92–138.
[5] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I.F. Cruz and F.M. Couto, The AgreementMakerLight ontology matching system, in: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, Springer, 2013, pp. 527–541.
[6] F. Scharffe, Correspondence patterns representation, PhD thesis, University of Innsbruck, 2009.
[7] D. Ritze, C. Meilicke, O. Sváb-Zamazal and H. Stuckenschmidt, A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences, in: *OM*, 2009.
[8] I. Megdiche, O. Teste and C. Trojahn, An extensible linear approach for holistic ontology matching, in: *International Semantic Web Conference*, Springer, 2016, pp. 393–410.
[9] C. Meilicke, Alignment incoherence in ontology matching, Universität Mannheim, 2011.
[10] E. Jiménez-Ruiz, B.C. Grau, I. Horrocks and R. Berlanga, Logic-based assessment of the compatibility of UMLS ontology sources, *Journal of biomedical semantics* **2**(1) (2011), 1–16.
[11] D. Faria, C. Pesquita, I. Mott, C. Martins, F.M. Couto and I.F. Cruz, Tackling the challenges of matching biomedical ontologies, *Journal of biomedical semantics* **9**(1) (2018), 1–19.
[12] C. Pesquita, D. Faria, C. Stroe, E. Santos, I.F. Cruz and F.M. Couto, What's in a "nym"? Synonyms in Biomedical Ontology Matching, in: *International Semantic Web Conference (ISWC)*, Lecture Notes in Computer Science, Vol. 8218, Springer, Berlin, Heidelberg, Germany, 2013, pp. 526–541.
[13] G.O. Consortium, The Gene Ontology (GO) database and informatics resource, *Nucleic acids research* **32**(suppl_1) (2004), D258–D261.
[14] M. Horridge and S. Bechhofer, The OWL API: A Java API for OWL ontologies, *Semantic Web* **2**(1) (2011), 11–21.
[15] F. Bancilhon, Naive evaluation of recursively defined relations, in: *On Knowledge Base Management Systems*, Springer, 1986, pp. 165–178.
[16] G. Stoilos, G. Stamou and S. Kollias, A String Metric for Ontology Alignment, in: *International Semantic Web Conference (ISWC)*, Lecture Notes in Computer Science, Vol. 3729, Springer, Berlin, Heidelberg, Germany, 2005, pp. 624–637.
[17] D. Faria, C. Pesquita, E. Santos, I.F. Cruz and F.M. Couto, Automatic Background Knowledge Selection for Matching Biomedical Ontologies, *PLoS One* **9**(11) (2014), e111226.
[18] E. Santos, D. Faria, C. Pesquita and F.M. Couto, Ontology alignment repair through modularization and confidence-based heuristics, *PLoS ONE* **10**(12) (2015), e0144807.
[19] C. Pesquita, D. Faria, E. Santos, J.-M. Neefs and F.M. Couto, Towards visualizing the alignment of large biomedical ontologies, in: *International Conference on Data Integration in the Life Sciences*, Springer, 2014, pp. 104–111.
[20] E. Jiménez-Ruiz and B. Cuenca Grau, Logmap: Logic-based and scalable ontology matching, in: *International Semantic Web Conference*, Springer, 2011, pp. 273–288.
[21] T. Baker, B. Whitehead, R. Musker and J. Keizer, Global agricultural concept space: lightweight semantics for pragmatic interoperability, *NPJ science of food* **3**(1) (2019), 1–8.
[22] E.B. Myklebust, E. Jiménez-Ruiz, J. Chen, R. Wolf and K.E. Tollefsen, Prediction of Adverse Biological Effects of Chemicals Using Knowledge Graph Embeddings, *arXiv preprint arXiv:2112.04605* (2021).
[23] E. Gringinger, R.M. Keller, A. Vennesland, C.G. Schuetz and B. Neumayr, A comparative study of two complex ontologies in air traffic management, in: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, IEEE, 2019, pp. 1–9.
[24] D. Faria, B. Lima, M.C. Silva, F.M. Couto and C. Pesquita, AML and AMLC results for OAEI 2021 (2021).
[25] G.V. Gkoutos, C. Mungall, S. Dolken, M. Ashburner, S. Lewis, J. Hancock, P. Schofield, S. Kohler and P.N. Robinson, Entity/quality-based logical definitions for the human skeletal phenome using PATO, in: *2009 annual international conference of the IEEE engineering in medicine and biology society*, IEEE, 2009, pp. 7069–7072.
[26] B.S. Balasubramani, O. Belingheri, E.S. Boria, I.F. Cruz, S. Derrible and M.D. Siciliano, GUIDES: Geospatial urban infrastructure data engineering solutions, in: *Proceedings of the 25th ACM sigspatial international conference on advances in geographic information systems*, 2017, pp. 1–4.
[27] A. Vennesland, R.M. Keller, C.G. Schuetz, E. Gringinger and B. Neumayr, Matching Ontologies for Air Traffic Management: a Comparison and Reference Alignment of the AIRM and NASA ATM Ontologies., in: *OM@ ISWC*, 2019, pp. 1–12.