The AgreementMakerLight Ontology Matching System

Daniel Faria[1], Catia Pesquita[1], Emanuel Santos[1], Matteo Palmonari[2], Isabel F. Cruz[3], and Francisco M. Couto[1]

[1] LASIGE, Department of Informatics, Faculty of Sciences of the University of Lisbon, Portugal
[2] Department of Informatics, Systems and Communication, University of Milan Bicocca, Italy
[3] ADVIS Lab, Department of Computer Science, University of Illinois at Chicago, USA

Abstract. AgreementMaker is one of the leading ontology matching systems, thanks to its combination of a flexible and extensible framework with a comprehensive user interface. In many domains, such as the biomedical, ontologies are becoming increasingly large thus presenting new challenges. We have developed a new core framework, AgreementMakerLight, focused on computational efficiency and designed to handle very large ontologies, while preserving most of the flexibility and extensibility of the original AgreementMaker framework. We evaluated the efficiency of AgreementMakerLight in two OAEI tracks: Anatomy and Large Biomedical Ontologies, obtaining excellent run time results. In addition, for the Anatomy track, AgreementMakerLight is now the best system as measured in terms of F-measure. Also in terms of F-measure, AgreementMakerLight is competitive with the best OAEI performers in two of the three tasks of the Large Biomedical Ontologies track that match whole ontologies.

1 Introduction

Ontology matching is essential for a full realization of the Semantic Web vision, by providing a means to link concepts from different ontologies. The ontology matching process takes as input two ontologies and outputs a set of correspondences between semantically related ontology concepts, also called an alignment [13]. Ontologies are fast becoming an integral part of many domains, such as biomedicine and geography, and in the last few years they have become increasingly large and complex. As a consequence the more recent ontology matching systems incorporate more elaborate approaches including scaling strategies [14,15,18], ontology repair techniques to ensure the coherence of the alignments [15], and the use of external resources and ontologies to increase the amount of available knowledge to support matching [14].

AgreementMaker has been one of the leading systems in the field of ontology and schema matching since the beginning of its development in 2001 [3,7].

It combines a powerful, flexible and extensible framework with a comprehensive user interface that enables alignment visualization and manual editing. However, AgreementMaker was not originally designed to match ontologies with more than a few thousand concepts. It relies on memory-intensive complex ontology representations and similarity matrices that store the correspondence scores between all concepts in both ontologies. These structures hinder the scalability of AgreementMaker's matching process. Furthermore, AgreementMaker is optimized for visualization, which is critical to support user interaction for semi-automatic ontology matching, but computationally intensive.

The Ontology Alignment Evaluation Initiative (OAEI) is an annual competition that provides a benchmark evaluation for ontology alignment systems [12]. In recent years the AgreementMaker system ranked among the top systems, and in particular was the top system on the Anatomy track in 2010 and 2011, with the best results ever for this track [9,10]. In 2012, reflecting the needs of the biomedical domain, the OAEI competition introduced the Large Biomedical Ontologies track. This track consists on matching ontologies with tens of thousands of terms, which poses new challenges to ontology matching systems. While AgreementMaker did not participate in OAEI 2012, it would have struggled to match ontologies of this size.

Given the increasing importance of matching very large ontologies, particularly in the biomedical domain, we have developed a novel ontology matching framework, derived from AgreementMaker and focused on the efficient matching of very large ontologies — the AgreementMakerLight (AML) framework. Unlike AgreementMaker it is not designed to support user interaction, but it maintains the flexibility and extensibility of the original framework. In this paper, we present the AML framework and its evaluation on the Anatomy and Large Biomedical Ontologies tracks of the OAEI 2012.

This paper is organized as follows: Section 2 presents some basic concepts in ontology matching, Section 3 presents the AML framework in depth, Section 4 shows the details of its evaluation, Section 5 presents and discusses the results of the evaluation, and Section 6 presents the main conclusions of our work.

2 Ontology Matching Concepts

In this section, we introduce some of the nomenclature commonly used in ontology matching, which we will use throughout this paper.

The process of *matching* or *aligning* two input ontologies (one *source* ontology and one *target* ontology) consists in finding semantic relationships between the classes of the *source* ontology and the classes of the *target* ontology. In the context of this paper, these semantic relationships are restricted to equivalence relationships, and are called *mappings*. The set of mappings between two ontologies is called an *alignment* [3].

Ontology matching systems use matching algorithms, called *matchers*, which assign a numerical value to each mapping. This numerical value reflects the semantic similarity between terms. These matchers can function at different levels, including the element level and the structural level [13].

Element-level matchers analyze concepts or their instances in isolation, ignoring their relations with other concepts or instances. These matchers can use internal knowledge only, that is, information contained in the ontology itself, or incorporate external knowledge in the form of reusable alignments, upper or domain ontologies, and other linguistic resources. A popular internal-knowledge element-level matching technique is based on the lexical matching of the labels associated with ontology concepts.

Structure-level techniques compare ontology concepts or their instances based on their relationships with other concepts or instances. They can also use external knowledge, such as instances that are not part of the ontology or previous alignments.

Most ontology systems aggregate several distinct matchers:

- sequential composition, where the results of one matcher are fed to the next
- parallel composition, where distinct matchers are run independently, and their results are combined following specific criteria,
    - homogeneous, in which the different kinds of data are processed by appropriate matchers
    - heterogeneous, in which the same input is used by distinct matchers

Furthermore, the similarity between two ontology concepts may involve the ontologies as a whole, so that the final similarity between two concepts may ultimately depend on all of them. Several approaches use this notion to propagate similarities throughout the ontology [16,11,8].

After the similarities between ontology concepts have been computed, it is necessary to use a global strategy to arrive at a final optimized alignment. These techniques can include trimming, which applies thresholds to ensure only the best matches are considered; or maximal weight matching (or weaker variants like stable marriage), which optimize the global similarity [4].

Recently, ontology matching systems have begun to include approaches to ensure the ontological quality of their outputs, such as the application of rules to prune out illogical mappings [14] or the use of full-fledged repair approaches that strive to ensure the coherence of the final alignment, that is, that all classes are satisfiable [15].

3 AgreementMakerLight Framework

3.1 Overview

The AML framework was programmed in Java and developed in Eclipse. The core framework includes two modules: the ontology loading module and the ontology matching module. The ontology loading module is responsible for loading the input ontology files and constructing ontology objects (Figure 1). It is also responsible for loading external ontologies used as background knowledge. The ontology matching module is responsible for aligning ontology objects by combining one or more matching algorithms and one or more selection steps (Figure 2).

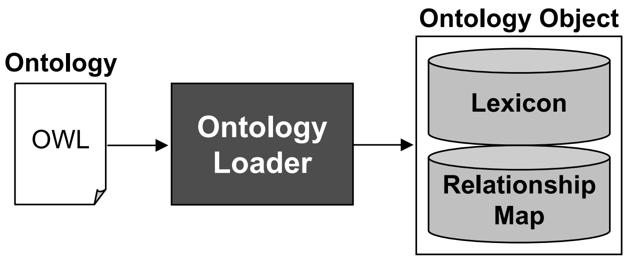

Fig. 1. Schema of the AgreementMakerLight Ontology Loading Module

Like AgreementMaker, the AML ontology matching module was designed with flexibility and extensibility in mind, and thus allows for the inclusion of virtually any matching algorithm.

The AML framework also includes three key data structures: the *Lexicon*, the *RelationshipMap* and the *Alignment*. The first two data structures are the main components of Ontology Objects, i.e., representations of ontologies used in AML to support the matching process. As their names suggest, the *Lexicon* stores the lexical information of an ontology (i.e., the labels and synonyms of each term) and the *RelationshipMap* stores the structural information of an ontology (i.e., the relationships between all terms). The *Alignment* stores the set of mappings between two ontologies produced by one or more matching algorithms.

3.2 Ontology Loading Module

As is the case in AgreementMaker, the AML ontology loading module is currently based on the Jena2 ontology API (Jena), and the first step in the loading process is to read the ontology file into memory as a Jena OntModel. The difference between the two systems is that AML stores in internal data structures all the information from the OntModel that is necessary for ontology matching, whereas AgreementMaker keeps the OntModel in memory throughout the matching process.

After using Jena to read an input ontology file, the AML ontology loading module extracts from the OntModel the following information, which is linked together under an Ontology Object:

- The URI or the ontology.
- A list of URI of all named classes in the ontology that belong to the ontology's namespace.
- A list of local names of all listed classes.
- A list of the synonym properties used in the ontology (e.g., *hasExactSynonym, hasRelatedSynonym*).

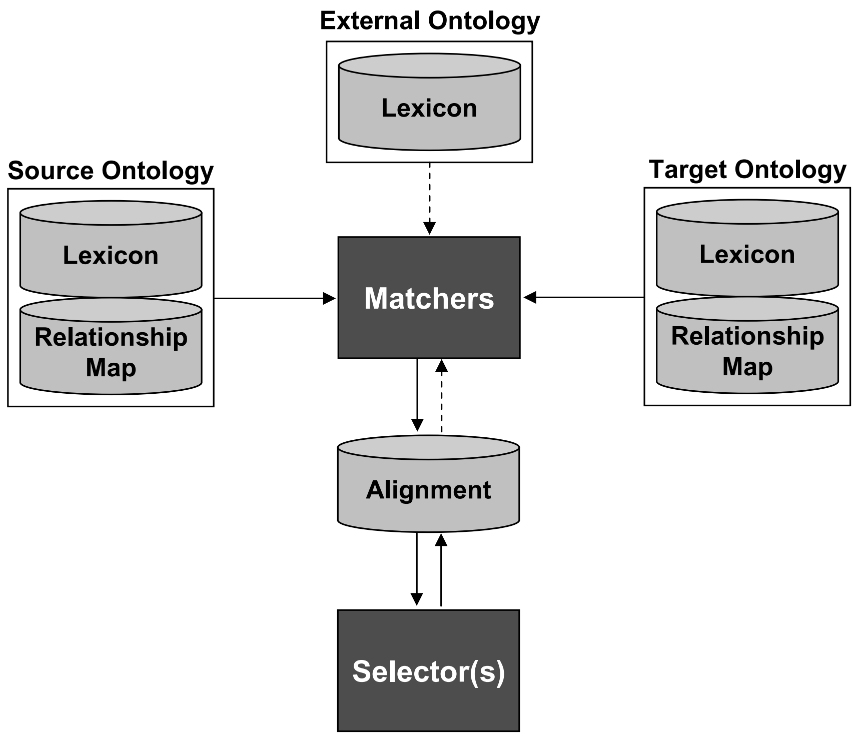

Fig. 2. Schema of the AgreementMakerLight Ontology Matching Module

– A *Lexicon* that contains the local names of all listed classes (when they are not alphanumeric codes), their labels, and all their synonyms (as declared in synonym property statements).
– A *RelationshipMap* that contains the *is a* and *part of* relationships between all listed classes (with transitive closure) plus all disjoint clauses between the listed classes (without transitive closure).

Building the *RelationshipMap* is optional, since the relationships are unnecessary for many matching algorithms, and this step takes approximately 60% of the total ontology loading time. In particular, the *RelationshipMap* is not built when loading external ontologies to use as background knowledge, as only their *Lexicon* is necessary for this purpose.

Even when the *RelationshipMap* is built, the whole ontology loading process is fairly quick: in our 4 core CPU server with 16 GB total RAM, small ontologies (under 10,000 classes) are loaded in 1 or 2 seconds and even very large ontologies (approximately 120,000 classes) are loaded in under 150 seconds.

3.3 Data Structures

A key difference between AML and AgreementMaker is that in the former ontologies are represented exclusively by internal data structures, whereas in the latter internal data structures are used in addition to the Jena OntModel. While building the internal data structures from the OntModel takes time, if those structures are designed with the efficiency of the matching process in mind, they will reduce the total processing time considerably. Furthermore, the internal data structures take up less memory than the OntModel, so in not keeping the latter in memory, we effectively increase the available memory for the matching process. Last but not least, this setup means that AML's ontology matching module is not tied to Jena or any specific ontology-reading API. Thus AML can work with any ontology-reading API by simply changing the ontology loading module.

Lexicon is a data structure that links each class in an ontology with its "names" (i.e., local names, labels, and synonyms) and the provenance of those names (i.e., whether they come from a local name or label, or from which type of synonym property statement). While an equivalent data structure already existed in AgreementMaker, it was built after the ontology loading process and only used by some matching algorithms. In AML, the *Lexicon* is a primary data structure used by all matching algorithms that require lexical information.

A novel aspect that was incorporated in the AML *Lexicon* was a system of weights to reflect the reliability of each provenance. For instance, synonyms obtained from *hasExactSynonym* statements are in principle more reliable than synonyms obtained from *hasRelatedSynonym* statements, as they should be closer in meaning to the concept described by a class. Thus, local names were given a weight of 1.0, labels a weight of 0.95, exact synonyms a weight of 0.9 and other synonyms a weight of 0.85. These weights may be used by any matching algorithm that uses the *Lexicon*.

The internal structure of the *Lexicon* consists on two MultiMaps (which are HashMaps of HashMaps) containing classes, names and provenances, with one having the class as key and the other having the name as key. Thus, the *Lexicon* can be queried by both class and name at virtually no computational cost.

RelationshipMap is a data structure that links each class to the classes related to it through *is a* or *part of* relationships or disjoint clauses. It complements the *Lexicon*, and is a very efficient alternative to the node-based tree structure used in AgreementMaker to represent each ontology.

The *RelationshipMap* stores all *is a* and *part of* paths in an ontology with transitive closure, and includes the distance of each path in number of edges. It also stores all direct disjoint clauses in an ontology (without transitive closure). Like the *Lexicon*, the *RelationshipMap* is based on MultiMaps. It includes two MultiMaps for relationships which contain ancestors, descendents and relationship (i.e., type and distance), with one having the ancestor as key and the other one having the descendent as key. It also includes a HashMap of Sets for disjoint clauses, linking each class to all classes that are disjoint with it. Thus the RelationshipMap can be queried to obtain all descendents of a class, all ancestors of a class, and all classes disjoint with a class at virtually no computational cost.

Alignment is a data structure used by the ontology matching module to store mappings between the input ontologies. This structure was already used by AgreementMaker, and was ported directly from it. However, in AgreementMaker, *Alignment* was used only to store the final output of a matching algorithm or combination of algorithms. During the matching procedure, the primary data structure used by AgreementMaker is a matrix that stores the similarities between all concepts of the source ontology against all concepts of the target ontology, which we abbreviate in the rest of the paper as a *all-against-all* strategy. The problem with this structure is that it takes $O(m \times n)$ memory (where m and n are the number of concepts of the source and target ontologies, respectively) and therefore does not scale for large ontologies. For instance, the matrix that results from matching two ontologies with 50,000 classes would occupy 18.6 GB of memory, which is beyond the capacity of our server. Since the number of mappings in a matching problem with cardinality one-to-one is $O(min(m, n))$, the vast majority of the values in the similarity matrix is very small or zero, thus making their storage unnecessary. In AML we opted for storing similarities directly in the *Alignment* and discarding similarities that are below a given threshold.

The internal structure of *Alignment* in AML is identical to that of AgreementMaker. It includes two MultiMaps that contain the source class, target class and similarity, with one having the source class as key and the other one having the target class as key. This enables efficient querying of mappings by class, and means that *Alignment* corresponds to a sparse matrix. In addition, *Alignment* also includes a list structure that enables sorting and thus facilitates selection.

3.4 Ontology Matching Module

The AML ontology matching module contains three components: *Matchers* (i.e., matching algorithms), *Selectors* (i.e., selection algorithms), and the previously described *Alignment* data structure (see Figure 2).

Matchers are algorithms that compare two ontologies and return an *Alignment* between them. Like in AgreementMaker, any kind of matching algorithm can be implemented as an AML *Matcher*, requiring only the implementation of a matching algorithm (which receives as input two ontology objects and returns an *Alignment* between them) and an extension method (which receives as input two ontology objects and an *Alignment* between them, and extends that *Alignment* by mapping only unmapped classes). However, AML divides *Matchers* into *Primary Matchers* and *Secondary Matchers* according to their efficiency.

Primary Matchers are matching algorithms that rely on HashMap cross-searches (i.e., searches where a key in a HashMap is used to query another HashMap directly) and thus take $O(n)$ time. *Secondary Matchers* are matching algorithms that make non-literal comparisons between terms and thus require explicitly comparing each term in an ontology with each term in the other ontology, which takes $O(n^2)$ time. This means that *Secondary Matchers* cannot be used efficiently to match large ontologies. Taking this into account, we stipulated that the extension method implemented by *Secondary Matchers* should be less extensive than the one implemented by *Primary Matchers*. Thus, in *Secondary Matchers* the extension method only tries to map classes in the vicinity of previously mapped ones (e.g., children, parents, siblings), whereas in *Primary Matchers* the extension methods tries to map all unmapped classes. In general, *Primary Matchers* in AML are used in match mode whereas *Secondary Matchers* are used in extension mode.

One of the unique features of AgreementMaker is its strategy for combining the output of multiple *Matchers* to obtain a better final alignment, the linear weighted combination [3]. However, this strategy relies on the all-against-all similarity matrix produced by each *Matcher*, and thus is not directly portable to AML. Currently AML combines the output of multiple *Matchers* by simply joining the alignments and keeping the highest similarity in case of repeated mappings, which was a strategy previously used in AgreementMaker [5,6].

Selectors are algorithms used to trim an *Alignment* by excluding mappings below a given similarity threshold and excluding competing mappings (i.e., multiple mappings that include the same class) to obtain the desired cardinality. As already mentioned, the desired cardinality in ontology matching is typically one-to-one. The problem of trimming a many-to-many *Alignment* to obtain a one-to-one *Alignment* corresponds to a bipartite mapping problem. The selection algorithm implemented in AgreementMaker finds the maximum weighted bipartite mapping, thus maximizing the sum of the similarities of the selected mappings [4]. However, this algorithm relies on the all-against-all similarity matrix, and thus is not directly portable to AML. Nevertheless, it is not clear that the maximum mapping is the "optimal" mapping for an ontology alignment problem, since the goal of the problem is to maximize the number of correct mappings while minimizing the number of incorrect mappings. Thus, assuming that the similarity value given by the matching algorithms is correlated with the probability that they are correct, then selecting a high-quality alignment (90% similarity) might be better than selecting two competing medium-quality alignments (60% similarity), but the maximum mapping would include the latter.

Taking the above considerations into account, we developed a greedy *Ranked Selector* algorithm for AML, that selects mappings based on their similarity. This simple algorithm starts by sorting the mappings in the *Alignment* in descending order of their similarity values, then selects mappings in that order, as long as they do not include classes that were present in previously selected mappings. This ensures that each class appears in at most one mapping, but favors stronger individual mappings (in terms of similarity) instead of trying to maximize the total similarity of the selected mappings.

3.5 Implemented Matchers

In porting matchers from AgreementMaker, a paramount idea was that, whenever possible, we would eliminate the need for an all-against-all comparison and instead convert them into *Primary Matchers* whenever possible. We implemented four matchers: the *Lexical Matcher*, the *Mediating Matcher*, the *Word Matcher* and the *Parametric String Matcher*. The first three were implemented as *Primary Matchers* and the last one was implemented as a *Secondary Matcher*.

Lexical Matcher is one of the simplest and most efficient matching algorithms, which looks for literal name matches in the *Lexicons* of the input ontologies. It is derived from the *Lexical Synonym Weighted Matcher* of AgreementMaker, but uses a more streamlined weight system. For two input ontologies *source* and *target*, the *Lexical Matcher* algorithm is as follows:

```
set A = empty alignment
set list = names(source)
for each name in list
  if target contains name
    set sourceList = sourceClasses(name)
    set targetList = targetClasses(name)
    for each sourceClass in sourceList
      set weightSource = weight(name,sourceClass)
      for each targetClass in targetList
        set sim = weightSource * weight(name,targetClass)
        add (sourceClass,targetClass,sim) to A
end
```

Note that, regardless of the order in which the input ontologies are given, the *Lexical Matcher* always iterates through the *Lexicon* of the ontology that has the smallest number of names, so as to minimize the number of iterations.

Mediating Matcher is also a simple and efficient matching algorithm based on literal name matches, but it uses a third (external) ontology as a mediator between the input ontologies. It uses the *Lexical Matcher* to compute "bridge" alignments between the input ontologies and the mediating ontology, then cross-searches the bridge alignments to map the input ontologies. For two input ontologies *source* and *target* and a mediating ontology *med*, the *Mediating Matcher* algorithm is as follows:

```
set A = empty alignment
compute Bs = LexicalMatcher(med,source)
compute Bt = LexicalMatcher(med,target)
for each medClass in Bs
  if Bt contains medClass
    for each sourceClass in Bs(medClass)
      set simSource = similarity(medClass,sourceClass)
      for each targetClass in Bt(medClass)
        set sim = simSource * similarity(medClass,targetClass)
        add (sourceClass,targetClass,sim) to A
end
```

Word Matcher is a word-based string similarity algorithm that measures the similarity between two classes through a weighted Jaccard index between the words present in their names. It is derived from the *Vector-based Multi-word Matcher* of AgreementMaker, but is based on a lexical cross-search whereas the latter requires an all-against-all comparison.

The first step in the *Word Matcher* is the derivation of a *Word Lexicon* from the *Lexicon* of each ontology and computing the frequency and evidence content (EC) of each word. The EC of each word is given by the inverse logarithm of its frequency [2]. After computing the *Word Lexicons*, the *Word Matcher* algorithm is as follows:

```
set A = empty alignment
set list = words(source)
for each word in list
  if target contains word
    set sourceList = sourceClasses(word)
    set targetList = targetClasses(word)
    for each sourceClass in sourceList
      set weightS = sourceEC(word) * weight(word,sourceClass)
      for each targetClass in targetList
        set weightT = targetEC(word) * weight(word,targetClass)
        set sim = sqrt(weightS * weightT)
        if A contains (sourceClass,targetClass)
          set A(sourceClass,targetClass) += sim
        else
          add (sourceClass,targetClass,similarity) to A
for each (sourceClass,targetClass) in A
  set itr = similarity(sourceClass,targetClass)
  set uni = weight(sourceClass) + weight(targetClass) - itr
  set A(sourceClass,targetClass) = itr/uni
end
```

Despite also being based on a lexical cross-search, the *Word Matcher* has a steeper memory requirement than the previous two matchers. The reason for this is that it needs to store temporary mappings between all classes that share at least one word, since it can only filter out low similarity mappings after iterating through all words. In the worst case, its memory requirement will be $O(n^2)$ for very verbose ontologies. Thus, despite being a primary matcher, the *Word Matcher* requires that very large ontologies be partitioned due to memory constraints.

Parametric String Matcher is a string similarity algorithm that implements a variety of similarity metrics and was directly ported from AgreementMaker. Since it makes non-literal string comparisons, it requires an all-against-all comparison of the ontologies and therefore is a secondary matcher in AML.

4 Evaluation

We evaluated AML on the Anatomy and Large Biomedical Ontologies tracks of the OAEI 2012 [1].

The Anatomy track consists on a medium-size ontology matching task between the Adult Mouse Anatomy Dictionary (with 2737 classes) and part of the NCI thesaurus describing human anatomy (with 3298 classes). It is evaluated using a manually created reference alignment that has been extensively tested. We were interested in evaluating AML in this track given that AgreementMaker was the top system in it, both in 2010 and 2011. Thus, we wanted to assess whether the improvement in efficiency of AML came at the cost of performance in terms of precision, recall, and F-measure. As such, we selected a configuration for AML that is as close as possible to the configuration used by AgreementMaker in OAEI 2011. It combines the *Lexical Matcher*, the *Mediating Matcher* using UBERON [17] as the background ontology, the *Word Matcher* and the *Parametric String Matcher* in extension mode. All *Matchers* were used with a minimum similarity threshold of 0.6 except the *Parametric String Matcher*, which was used with a higher threshold of 0.7.

The Large Biomedical Ontologies track includes a total of 9 ontology matching tasks between three ontologies: the Foundational Model of Anatomy (FMA), the NCI thesaurus and the SNOMED Clinical Terms. For each pairwise combination of these ontologies, there is a task for matching small overlapping fragments, a task for matching extended fragments, and a task for matching the whole ontologies. The reference alignments used in OAEI 2012 to evaluate these tasks were obtained automatically from UMLS, and then repaired with ALCOMO and/or LogMap. We were interested in evaluating AML on the three whole ontologies tasks given that AgreementMaker was not able to handle them. Thus, we wanted to assess whether the improvement in efficiency of AML was sufficient for it to compete with the top systems in OAEI in terms of run time, and what was its baseline performance. Thus we selected a simple configuration for AML that combined only the *Lexical Matcher* (with a threshold of 0.7) with the *Parametric String Matcher* in extension mode (with a threshold of 0.7). We evaluated AML using the reference alignments repaired by ALCOMO for the

FMA-NCI and FMA-SNOMED tasks, and the reference alignment repaired by LogMap for the SNOMED-NCI task.

AML and AgreementMaker run times were measured in our local server, with 4 core CPU and 16 GB total RAM. They are not directly comparable to the run times reported for the OAEI 2012 Anatomy track, which were measured on a less powerful server (2 core CPU with 3 GB allocated RAM), but can be compared to the run times for the Large Biomedical Ontologies track, which were measured on a more powerful server (16 core CPU with 15 GB allocated RAM).

5 Results and Discussion

The AML results for the Anatomy track are presented in Table 1 together with those of AgreementMaker and the top three systems in OAEI 2012 [1]. Surprisingly, the performance of AML in terms of F-measure was slightly better than that of AgreementMaker, and also slightly better than the top system in OAEI 2012. The comparison with AgreementMaker is interesting because the configuration of the systems was very similar, but AgreementMaker has a more exaustive matching process (including the *Parametric String Matcher* used for global matching and the LWC combination strategy [4]) and therefore we were not expecting AML to perform better than AgreementMaker. While AgreementMaker did have a higher recall than AML, this was compensated by the higher precision of the latter. The main reason behind this increase in precision is likely the new weighting system used in the *Lexicon*, as this is the only substantial change made over AgreementMaker. In any case, it is clear that the features of AgreementMaker that were excluded from AML for the sake of efficiency did not affect AML's performance. They did, however, have a significant effect on efficiency, as the run time of AML was only 5% of the run time of AgreementMaker on the same server and conditions. While we cannot compare our run time directly with those reported for the Anatomy track in OAEI 2012 (since our server is more powerful than the server used in this track) a run time of 10 seconds is nevertheless a clear indication of the efficiency of AML.

Regarding the three tasks of the Large Biomedical Ontologies track, AML was able to match all ontologies in considerably less time than the best systems in OAEI 2012 (see Table 2). Furthermore, the OAEI 2012 run times were obtained

Table 1. Evaluation of the AgreementMakerLight system in the OAEI 2012 Anatomy track

System	Precision	Recall	F-Measure	Run Time (s)
AgreementMakerLight	96.1%	88.9%	92.4%	10*
GOMMA-bk	91.7%	92.8%	92.3%	15
AgreementMaker	95.2%	89.4%	92.2%	200*
YAM++	94.3%	85.8%	89.8%	69
CODI	96.6%	82.7%	89.1%	880

* Run times obtained using a more powerful server than the one used for OAEI 2012.

Table 2. Evaluation of the AgreementMakerLight system on the OAEI 2012 Large Biomedical Ontologies Track

System	Precision	Recall	F-Measure	Run Time (s)
FMA-NCI				
YAM++	86.2%	83.8%	85.0%	1304
GOMMA	82.9%	83.6%	83.3%	217
ServOMapL	84.4%	80.8%	82.6%	251
AgreementMakerLight	84.7%	71.8%	78.0%	89*
FMA-SNOMED				
ServOMapL	85.1%	69.1%	76.3%	517
ServOMap	84.2%	65.5%	73.7%	517
YAM++	79.1%	68.5%	73.4%	23900
AgreementMakerLight	88.0%	18.8%	40.7%	224*
SNOMED-NCI				
YAM++	78.5%	60.4%	68.3%	30155
ServOMapL	78.5%	59.8%	67.9%	738
LogMap	81.2%	57.7%	67.4%	955
AgreementMakerLight	91.8%	49.1%	67.1%	231*

* Run times obtained using a less powerful server than the one used for OAEI 2012.

in a more powerful server, so the difference between AML and these systems should be even greater than revealed by the results. In terms of quality (as measured by the F-measure), the results of AML are behind those of the top systems in OAEI 2012. This was expected, since we used only a simple configuration with two matching algorithms, as our main goal was assessing the efficiency of AML. Nevertheless, the results obtained in the FMA-NCI and SNOMED-NCI tasks are competitive, with a high precision (higher than the top OAEI 2012 systems, in the case of the SNOMED-NCI task) and a reasonable recall. We expect to improve these baseline results significantly once we focus on quality, namely by introducing adequate external information, which is a common feature of all the top OAEI 2012 systems. Given AML's advantage in terms of run time, we expect to obtain results comparable to those of the top OAEI 2012 systems in these tasks, while taking less time.

The results for the FMA-SNOMED task were excellent in terms of run time and precision, but not in terms of recall. However, we identified the problem behind the low recall, which can be overcome without difficulty: we noticed that SNOMED includes the word "structure" in the labels of most anatomical structures (e.g., "structure of hair of trunk", "spinal nerve structure") whereas FMA names the same structures directly (e.g., "hair of trunk", "spinal nerve"). Since our matching configuration for this task was based on the *Lexical Matcher*, which is a literal name matcher, any such mappings will not be found (they may be found by the *Parametric String Matcher*, but only if they lie in the neighborhood of mappings found by the *Lexical Matcher*). One simple strategy to circumvent this problem is to use a word matching algorithm such as our *Word Matcher*. Thus, we expect to be able to obtain a significantly higher recall once we optimize the *Word Matcher*.

6 Conclusions

Overall, the AML framework has fulfilled the goals for which it was designed. On the one hand, it was able to match very large ontologies efficiently, in a competitive time when compared with top ontology matching systems. On the other hand, it was able to produce high quality results in the Anatomy track, that surpassed even those of AgreementMaker. In comparison with AgreementMaker, AML represents a substantial improvement in terms of efficiency without sacrificing performance, as measured in terms of precision, recall, and F-measure. Furthermore, AML shares the focus of AgreementMaker on flexibility and extensibility, which are part of its design strengths [3].

In the Anatomy track, the results obtained by AML place it above the systems that competed in OAEI 2012. We believe that AML can improve even further upon these results, considering that scalability to large and very large ontologies has been, until now, the main focus of the work behind AML. Regarding the Large Biomedical Ontologies track, AML obtained excellent run times, but it is clear that performance can be improved—especially recall—once we enlarge our focus. In fact, even without focusing on performance, the results obtained by AML on the FMA-NCI and SNOMED-NCI tasks were competitive. Thus, we expect AML to place among the very best systems in these tasks, with the inclusion of suitable external data and a more complete matching configuration. As for the FMA-SNOMED task, we identified that the cause for the low recall obtained by AML was the unusual naming convention of the anatomical structures in SNOMED. We expect to circumvent this problem by using a word-based matching algorithm, which together with the inclusion of external data, will enable AML to obtain competitive results in this task while remaining among the fastest systems.

Further improvements to the AML framework will include innovative strategies for using external resources, repairing alignments, and using semantic similarity in the context of structural matching [19].

Acknowledgments. DF, CP, ES and FMC were funded by the Portuguese FCT through the SOMER project (PTDC/EIA-EIA/119119/2010) and the multiannual funding program to LASIGE. The research of IFC was partially supported by NSF Awards 0812258, 1143926, and 1213013, and by UIC Area of Excellence and Civic Engagement Research Awards.

References

1. Aguirre, J.L., Eckert, K., Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Sváb-Zamazal, O., dos Santos, C.T., Jiménez-Ruiz, E., Grau, B.C., Zapilko, B.: Results of the Ontology Alignment Evaluation Initiative 2012. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 946, pp. 73–115 (2012)
2. Couto, F., Silva, M., Coutinho, P.: Finding genomic ontology terms in text using evidence content. BMC Bioinformatics 6(suppl. 1), S21 (2005)

3. Cruz, I.F., Palandri Antonelli, F., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB 2(2), 1586–1589 (2009)
4. Cruz, I.F., Palandri Antonelli, F., Stroe, C.: Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 551, pp. 49–60 (2009)
5. Cruz, I.F., Palmonari, M., Caimi, F., Stroe, C.: Towards "On the Go" Matching of Linked Open Data Ontologies. In: IJCAI Workshop Discovering Meaning on the Go in Large & Heterogeneous Data (LHD), pp. 37–42 (2011)
6. Cruz, I.F., Palmonari, M., Caimi, F., Stroe, C.: Building Linked Ontologies with High Precision Using Subclass Mapping Discovery. Artificial Intelligence Review (2012) (to appear)
7. Cruz, I.F., Stroe, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F.M., Palmonari, M.: Using AgreementMaker to Align Ontologies for OAEI 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 114–121 (2011)
8. Cruz, I.F., Sunna, W.: Structural Alignment Methods with Applications to Geospatial Ontologies. Transactions in GIS 12(6), 683–711 (2008)
9. Euzenat, J., Ferrara, A., Meilicke, C., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 85–117 (2010)
10. Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 85–113 (2011)
11. Euzenat, J., Loup, D., Touzani, M., Valtchev, P.: Ontology alignment with OLA. In: Proceedings of the 3rd International Workshop on Evaluation of Ontology based Tools (EON), Hiroshima, Japan (2004)
12. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: Six years of experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
13. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag New York Inc. (2007)
14. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA results for OAEI 2012. In: Ontology Matching Workshop, International Semantic Web Conference 2012 (2012)
15. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y.: LogMap 2.0: towards logic-based, scalable and interactive ontology matching. In: Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences, pp. 45–46 (2011)
16. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, pp. 117–128 (2001)
17. Mungall, C.J., Torniai, C., Gkoutos, G.V., Lewis, S., Haendel, M.A.: Uberon, an Integrative Multi-species Anatomy Ontology. Genome Biology 13(1), R5 (2012)
18. Ngo, D., Bellahsene, Z.: Yam++: A multi-strategy based approach for ontology matching task. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 421–425. Springer, Heidelberg (2012)
19. Pesquita, C., Stroe, C., Cruz, I.F., Couto, F.: BLOOMS on AgreementMaker: Results for OAEI 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 135–141 (2010)