Large-scale Interactive Ontology Matching: Algorithms and Implementation

Ernesto Jiménez-Ruiz and **Bernardo Cuenca Grau** and **Yujiao Zhou** and **Ian Horrocks**[1]

Abstract. In this paper we present the ontology matching system LogMap 2, a much improved version of its predecessor LogMap. LogMap 2 supports user interaction during the matching process, which is essential for use cases requiring very accurate mappings. Interactivity, however, imposes very strict scalability requirements; we are able to satisfy these requirements by providing real-time user response even for large-scale ontologies. Finally, LogMap 2 implements scalable reasoning and diagnosis algorithms, which minimise any logical inconsistencies introduced by the matching process.

1 MOTIVATION

Ontologies are extensively used in domains such as biomedicine. The most widely used ontology modelling language is the Web Ontology Language (OWL) and its revision OWL 2 [7]. Effective ontology reuse and integration techniques are often needed when developing OWL-based biomedical applications. In particular, reuse of *reference biomedical ontologies*, such as SNOMED CT, the National Cancer Institute Thesaurus (NCI) and the Foundational Model of Anatomy (FMA), facilitates interoperability between applications, and is known to significantly reduce ontology development costs. Furthermore, applications sometimes rely on different reference ontologies, so integrating the relevant reference ontologies is a prerequisite for these applications to interoperate.

A first step in the reuse and integration of independently developed ontologies is to establish suitable *mappings* between their vocabularies. Although the problem of *ontology matching* has received significant attention in the last few years [8, 25], applications are still faced with many important practical challenges.

To make the discussion concrete, we next focus on two use cases as running examples, but the core technical problems are shared with many commercial and academic biomedical applications.

The LUCA use case. The Systems Engineering Group at Oxford University is developing LUCA—a medium-sized OWL ontology that describes the domain of Lung Cancer according to the specifications of the National Health Service (NHS) [24]. To facilitate interoperability with other applications within the NHS, LUCA needs to be integrated with SNOMED CT, which is the reference ontology of choice across NHS's information systems. To this end, LUCA's developers would like to *(i)* identify the concepts in SNOMED CT related to those in LUCA and establish suitable *mappings* between them; and *(ii)* import a small fragment of SNOMED CT (i.e., a *module*) that captures the meaning of such relevant concepts.

SNOMED CT, however, is a complex ontology describing more than 300,000 concepts, and computing mappings with LUCA is unfeasible without suitable tool support. To perform task *(i)* automatically, one could use an *ontology matching system*; however, although existing systems can deal with small ontologies, large-scale ontologies such as SNOMED CT are often beyond their reach. For example, the ontologies in the largest test case of the OAEI 2011 ontology matching initiative contain only 2,000–3,000 concepts, yet only 6 out of 16 systems succeeded in matching them [8]. Most importantly, even if LUCA and SNOMED CT could be matched "offline" (e.g., by running a system overnight on a high performance server), mapping computation heavily relies on heuristics such as string similarity measures, and is thus inevitably error prone; as a result, ontology developers still need to undertake an expensive a-posteriori manual curation of the mappings. Many mapping errors could be avoided if the user is *interactively involved* in the matching process [9]; however, most systems are fully automated, and do not allow the user to intervene. Finally, to perform task *(ii)*, one could use logic-based ontology modularisation techniques [6, 17]; however, these techniques require the relevant vocabulary to be given as input, and hence depend on the successful completion of task *(i)*.

The UMLS-META use case. UMLS Metathesaurus (UMLS-META) contains more than one hundred ontologies and thesauri together with mappings between them [4]. UMLS-META is used in a wide range of applications such as *PubMed* (a search engine for biomedical articles) and *ClinicalTrials.gov* (a worldwide registry of clinical trials). The integration of new ontologies in UMLS-META is a complex process that heavily relies on expert assessment and sophisticated auditing protocols. Such curation and auditing processes, however, are not only extremely costly, but also error-prone. For example, the ontology obtained from the union of FMA, SNOMED CT and the UMLS-META mappings between them contains over 6,000 unsatisfiable concepts [15]. Although several tools can assist the user in mapping curation [14, 19], they do not scale to large ontologies and mapping sets. Thus, as UMLS-META grows in both size and complexity, it becomes imperative to minimise manual curation, while at the same time reducing the number of errors.

To meet the requirements of these use cases, the availability of ontology matching tools with the following features seems essential.

1. *Interactivity.* Most systems rely on heuristics to prune candidate mappings. These heuristics, however, often lead to wrong choices, which can have a "cascade effect" and lead to further errors. Thus, expensive a-posteriori curation of the output mappings is often unavoidable. User intervention during the matching process becomes essential for use cases requiring very accurate mappings, such as LUCA and UMLS-META [9]. Indeed, allowing domain experts to interactively contribute to the matching process when critical choices need to be made could significantly improve matching results while keeping user intervention to a minimum.

[1] Department of Computer Science, University of Oxford, UK, email: {ernesto,berg,yujiao.zhou,ian.horrocks}@cs.ox.ac.uk

2. *Scalability.* Ontology matching is seen by many as an offline process, and systems are often not designed with scalability in mind. Most systems run out of memory with input ontologies containing a few thousand concepts, and hence they cannot deal with our use cases. Furthermore, interactivity imposes very strict scalability requirements, as users will expect reasonable response times.
3. *Reasoning-based error diagnosis.* OWL ontologies have well-defined semantics [7], and mappings are commonly represented as OWL axioms [14]. Many systems, however, disregard the semantics of the input ontologies, and are thus unable to detect and repair inconsistencies that follow from the union of the input ontologies and the mappings. Although there is a growing interest in reasoning techniques for ontology matching [11, 22, 10, 14, 15, 12, 18], reasoning is known to exacerbate the scalability problem [13].

The problem of ontology matching has been extensively studied in the last ten years (see [8, 25]). To the best of our knowledge, however, there is no system with all the above mentioned features.

- Very few systems support user interaction [9]. Examples include COGZ (a visual plugin for PROMPT [21]), COMA++ [2], and Muse [1]. These systems, however, do not scale to ontologies containing thousands of concepts, and they do not provide reasoning-based diagnosis features. Although several tools (e.g. [14, 19]) support interactive mapping diagnosis, they can only be used for a-posteriori mapping curation (i.e., not *during* the matching process), and they do not scale to large ontologies and mapping sets.
- The systems ASMOV [11], KOSIMap [22], CODI [10], and LogMap [12] implement reasoning-based diagnosis techniques. Among these, however, only LogMap can process large-scale ontologies, and none of them supports user interaction.
- Very few systems could process the largest test cases in the OAEI 2011 competition, and only GOMMA [16] and LogMap have shown to successfully scale to larger ontologies. None of these systems, however, supports user interaction. Furthermore, GOMMA is based on lexical matching techniques and does not support reasoning-based diagnosis.

To bridge this gap, we present LogMap 2—a new system that can meet the interactivity, scalability, and diagnosis requirements imposed by biomedical applications such as LUCA and UMLS-META.

2 The Design of a Scalable, Logic-based and Interactive Ontology Matching System

LogMap [12] has partly addressed the problem of efficiently matching large-scale ontologies while performing "on the fly" reasoning-based diagnosis. LogMap, however, provides no support for user interaction, and the underlying algorithm cannot be easily extended with interactive features. Furthermore, despite being highly optimised, LogMap's performance on large-scale ontologies is unsatisfactory for user interaction, where real time response is required. Thus, although certain elements in LogMap can be used, a redesign of the matching algorithm was needed. Section 2.1 discusses our initial design choices; in particular, we discuss which specific techniques implemented in LogMap 2 have been borrowed from LogMap. Section 2.2 discusses the new architecture of LogMap 2 and the novel techniques that we have developed.

2.1 The Starting Point

The keys to LogMap's favourable scalability behaviour can be summarised as given next. LogMap 2 borrows from these ideas and improves on each of them.

```
Input: $\mathcal{O}_1$, $\mathcal{O}_2$: input ontologies; $Interact$: Boolean value
Output: $\mathcal{M}$: mappings.
1: $\langle LI_1, LI_2\rangle := \mathsf{LexicalIndexes}(\mathcal{O}_1, \mathcal{O}_2)$
2: $\mathcal{M}_? := \mathsf{CandidateMappings}(LI_1, LI_2)$
3: $\langle \mathcal{O}_1', \mathcal{O}_2'\rangle := \mathsf{Module}(\mathcal{O}_1, \mathcal{O}_2, \mathcal{M}_?)$    ▷ End of Stage 1
4: $\mathcal{M} := \mathsf{ReliableMappings}(\mathcal{M}_?)$
5: $\mathcal{M}_? := \mathcal{M}_? \setminus \mathcal{M}$
6: $\langle \mathcal{P}_1', \mathcal{P}_2'\rangle := \mathsf{PropEncoding}(\mathcal{O}_1', \mathcal{O}_2')$
7: $\mathcal{M} := \mathsf{Diagnosis}(\mathcal{P}_1', \mathcal{P}_2', \mathcal{M}, \emptyset)$
8: $SI := \mathsf{SemanticIndex}(\mathcal{P}_1', \mathcal{P}_2', \mathcal{M})$
9: $\mathcal{M}_? := \mathcal{M}_? \setminus \mathsf{Discarded}(LI_1, LI_2, SI, \mathcal{M}_?)$
10: if ($Interact = \mathsf{true}$) then
11:     $\mathcal{M}_{user} := \mathsf{InteractiveProcess}(SI, \mathcal{M}_?)$    ▷ See Figure 3
12:     $\mathcal{M} := \mathsf{Diagnosis}(\mathcal{P}_1', \mathcal{P}_2', \mathcal{M} \cup \mathcal{M}_{user}, \mathcal{M}_{user})$
13: else
14:     $\mathcal{M} := \mathsf{Diagnosis}(\mathcal{P}_1', \mathcal{P}_2', \mathcal{M} \cup \mathcal{M}_?, \mathcal{M})$
15: end if
16: return $\mathcal{M}$
```

Figure 1: High level description of LogMap 2

- *Lexical indexation.* An inverted index is used to store information from entity annotations. This allows for efficient computation of an initial set of mappings that are almost "lexically exact".
- *Hierarchy indexation.* The concept hierarchy and the "told" disjointness relationships between concepts in each input ontology are efficiently stored using an interval labelling schema —an optimised data structure for storing directed acyclic graphs that significantly reduces the cost of answering typical taxonomic queries.
- *Propositional reasoning and "greedy" diagnosis.* Input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ are encoded in Horn propositional representations $\mathcal{P}_1$ and $\mathcal{P}_2$. Even if incomplete, such encoding allows LogMap to detect and repair unsatisfiable concepts both soundly and efficiently. Furthermore, incompleteness levels have been shown to be very low in practice. Satisfiability of concept C (seen as a propositional variable) is determined by checking satisfiability of $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{M} \cup \{\mathsf{true} \rightarrow C\}$, with $\mathcal{M}$ the mappings computed thus far (seen as propositional implications). *Diagnosis* is the task of computing a maximal subset $\mathcal{M}'$ of $\mathcal{M}$ such that $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{M}' \cup \{\mathsf{true} \rightarrow C\}$ is satisfiable for each C in $\mathcal{O}_1 \cup \mathcal{O}_2$. LogMap implements a greedy algorithm that deletes as few mappings as possible from $\mathcal{M}$.

2.2 The Architecture of LogMap 2

LogMap 2 accepts as input OWL 2 ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, and returns a set $\mathcal{M}$ of mappings of the form $A_1 \sqsubseteq A_2$, with A_1 and A_2 atomic concepts in $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively.

LogMap 2 can work in two modes: *interactive mode*, where the user will be asked a (hopefully small) number of questions to assist LogMap 2 in "critical" decisions, and *automatic mode*, where the whole matching process is completed automatically. In both modes, LogMap 2's algorithm can be divided in two main stages:

1. *Computation of candidate mappings.* To be both efficient and comprehensive, LogMap 2 computes a (typically large) set $\mathcal{M}_?$ of candidate mappings using lexical techniques only. The goal is to *maximise recall*; that is, to efficiently compute an initial set of candidates containing as many valid mappings as possible. No further mappings are computed after this stage; thus, the recall obtained with $\mathcal{M}_?$ is an upper-bound on the recall for the final output $\mathcal{M}$.

Table 1: Inverted Indexes for FMA and NCI

Inverted Index for FMA		**Ids for FMA concept URIs**	
Index entry	***Concept ids***	***Concept id***	***Concept URI (namespace omitted)***
{ acinus }	6953,7661,8171	6953	Mixed_acinus
{ common,branch,artery }	1170,7842	7661	Serous_acinus
		8171	Hepatic_acinus
		1170	Branch_of_common_cochlear_artery
		7842	Branch_of_common_interosseous_artery
Inverted Index for NCI		**Ids for NCI concept URIs**	
Index entry	***Concept ids***	***Concept id***	***Concept URI (namespace omitted)***
{ acinus }	18081	18081	Liver_acinus
{ common,branch,artery }	1204,8087,27727	1204	Common_carotid_artery_branch
		8087	Common_iliac_artery_branch
		27727	Common_femoral_artery_branch

Logic-based module extraction techniques [6] are used to compute ontology fragments $\mathcal{O}'_1 \subseteq \mathcal{O}_1$ and $\mathcal{O}'_2 \subseteq \mathcal{O}_2$ that "capture the meaning" in each input ontology of the concepts mapped by $\mathcal{M}_?$; these fragments will be used instead of the input ontologies when performing reasoning-based diagnosis tasks.

2. *Assessment of candidate mappings.* The goal of this stage is to *maximise precision* without harming recall. This is achieved by progressively discarding mappings that are "very likely" to be incorrect, as well as by identifying those that are "very likely" to be correct. Lexical, structural and reasoning-based techniques are all used in this process. In the interactive mode, cases that are not "clear-cut" are referred to the user; in the automatic mode, such unclear cases are resolved heuristically.

We next discuss each of the main steps performed by LogMap 2, which are schematically represented in Figure 1.

2.2.1 *Maximising recall: computing candidate mappings*

Lexical Indexation. Concept labels and URIs in each input ontology are stored using an inverted lexical index. Table 1 illustrates the structure of the index for FMA and NCI. An entry is a set of words occurring together in the label or URI of a concept; the index maps each entry with integer ids, each of which uniquely corresponds to a concept URI. For example, entry {common, branch, artery} is mapped to the internal id 1170, which corresponds to the URI FMA:Branch_of_common_cochlear_artery; thus, this concept contains all three words in the index entry in either its URI, or in one of its labels (e.g., synonyms). To consider lexical variations, additional index entries are included using stemming techniques.

As already mentioned, LogMap 2 largely borrows its indexation scheme from LogMap. However, the index in LogMap 2 is much larger than the one in its predecessor, as it serves a different purpose.

Computing candidate mappings. The set $\mathcal{M}_?$ of *candidate mappings* is computed from the inverted indexes LI_1 and LI_2 of $\mathcal{O}_1$ and $\mathcal{O}_2$ (Step 2 in Figure 1). LogMap 2 considers all entries e_1 in LI_1 and e_2 in LI_2 such that $e_1 = e_2$ and creates mappings $A_1 \sqsubseteq A_2$ and $A_2 \sqsubseteq A_1$ for each concept A_1 corresponding to e_1 and A_2 corresponding to e_2. For example, the entry {acinus} occurs in both indexes from Table 1; this entry corresponds to concepts FMA:Mixed_acinus, FMA:Serous_acinus, and FMA:Hepatic_acinus in FMA's index and to concept NCI:Liver_acinus in NCI's index. Thus, our tool considers as candidate mappings all axioms of the form NCI:Liver_acinus $\sqsubseteq X$ and $X \sqsubseteq$ NCI:Liver_acinus, for $X =$ FMA:Mixed_acinus, or $X =$ FMA:Serous_acinus, or $X =$ FMA:Hepatic_acinus.

Most candidate mappings will turn out to be incorrect (e.g., precision for FMA-NCI at this stage is 0.14); the goal, however, is to maximise recall (e.g., 0.93 recall for FMA-NCI), while keeping the number of candidates manageable (e.g., 19,151 for FMA-NCI, which contain 78,989 and 66,724 concepts respectively).

Logic-based module extraction. LogMap 2 implements sound algorithms for detecting unsatisfiable concepts, and for fixing the source of unsatisfiability. The practical feasibility of these techniques, however, critically depends on the size of the input ontologies. LogMap 2 exploits ontology modularisation techniques. More precisely, given $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{M}_?$, it computes fragments $\mathcal{O}'_1 \subseteq \mathcal{O}_1$ and $\mathcal{O}'_2 \subseteq \mathcal{O}_2$ (Step 3 in Figure 1) with the following properties:

- If $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}_? \models A \sqsubseteq B$ with A and B atomic, $\top$, or $\bot$, and either A or B occurs in $\mathcal{M}_?$, then $\{A, B\} \subseteq \mathsf{sig}(\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}_?)$.
- If $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}_? \models \alpha$ with α an arbitrary OWL axiom, and $\mathsf{sig}(\alpha) \subseteq \mathsf{sig}(\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}_?)$, then $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}_? \models \alpha$.

The first condition ensures that $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}_?$ contains in its signature all super-concepts of a concept in a candidate mapping; the second one ensures that the module $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}_?$ is logically indistinguishable from $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}_?$ w.r.t. its own signature. Such modules can be efficiently computed [6], and are typically much smaller than the original ontologies (e.g., the module computed for FMA-NCI contains 37% of the concepts in FMA, and 38% of those in NCI).

2.2.2 *Maximising precision: assessing candidate mappings*

Reliable mappings are those candidate mappings that LogMap 2 considers very likely to be correct; these mappings are provisionally taken as part of the final output (see Step 4 in Figure 1). In order for a mapping to be seen as "reliable", it must satisfy two requirements.

- *High similarity value.* Our tool assigns a similarity value to each candidate mapping using an off-the-shelf string matching tool. The threshold required to reliable mappings is very high; hence, reliable mappings involve concepts that are almost lexically identical (e.g., the mapping FMA:CarpalBone $\sqsubseteq$ NCI:Carpal_Bone).
- *Matching contexts.* Not all mappings with high similarity are correct. For example, in FMA:Trapezoid $\sqsubseteq$ NCI:Trapezoid, concept FMA:Trapezoid refers to a bone, whereas NCI:Trapezoid refers to a polygon. When parsing, LogMap 2 keeps basic information about the neighbours of each concept in the asserted hierarchy of $\mathcal{O}_1$ and $\mathcal{O}_2$. A mapping between A and B is considered reliable only if there is also a candidate mapping relating a neighbour of A to a neighbour of B.

Reasoning-based diagnosis. Reliable mappings typically have a high precision w.r.t. the Gold Standard (e.g. 2,281 reliable mappings for FMA-NCI with a precision of 0.9). Even if all atomic concepts in the union of the input ontologies and the Gold Standard mappings are satisfiable, a few incorrect reliable mappings can cause many concepts to become unsatisfiable. For example, the union of FMA, NCI and the set of reliable mappings computed by LogMap 2 (which have a precision of 0.9) entails more than 600 unsatisfiable concepts.

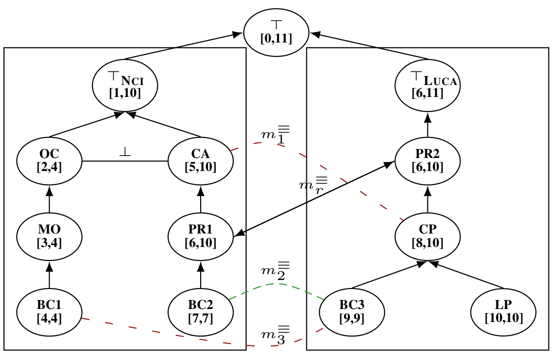

Figure 2: Semantic index for NCI-LUCA. Each node in the index is labelled with one or more intervals. Each interval characterises a subgraph with its respective minimum and maximum preorder numbers. Disjointness between intervals is also stored. *MO* = Medical_occupation, *OC* = Occupation, *BC1* = Bronchoscopist, *CA* = Clinical_activity, *PR1* and *PR2* = Procedure, *BC2* and *BC3* = Bronchoscopy, *CP* = Clinical_procedure, *LP* = Laboratory_procedure.

LogMap 2 exploits the propositional encoding and reasoning algorithms already used by its predecessor to detect and repair these unsatisfiable concepts (Steps 6 and 7 in Figure 1). As a result, a small number of mappings are discarded and a "clean" set of reliable mappings is efficiently computed. In the diagnosis process, $\mathcal{P}'_1$, $\mathcal{P}'_2$, and a subset of the mappings (see last parameter of the Diagnosis procedure) are considered to be "fixed" (i.e., they cannot be discarded during diagnosis). In contrast to LogMap, modules are used instead of the input ontologies, which improves reasoning performance.

Semantic indexation. The entailments of $\mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{M}$, with $\mathcal{M}$ the reliable mappings after diagnosis, are efficiently encoded in a *semantic index* (Step 8 in Figure 1) using an interval labelling schema [5]. This index reduces the cost of computing "semantic" queries over $\mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{M}$; an example is given in Figure 2. The interval encoding of propositional variables allows us to efficiently reduce to integer operations complex queries such as "Is NCI:Bronchoscopist disjoint with LUCA:Procedure w.r.t. $\mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{M}$?". Similar interval encodings were used in LogMap for indexing the concept hierarchy of each input ontology; in contrast to the indexing scheme in LogMap, however, the semantic index in LogMap 2 allows us to efficiently answer entailment queries over the integration of the input ontologies and the mappings computed thus far.

Discarding candidate mappings. LogMap 2 exploits the lexical and semantic indexes to discard most of the remaining "non-reliable" mappings (Step 9 in Figure 1). The process consists of two stages:

1. *Conflicts with the semantic index.* LogMap 2 discards the mappings in $\mathcal{M}_?$ that, when added to $\mathcal{M}$, make a concept unsatisfiable (e.g., the equivalence mapping $m_3^{\equiv}$ in Figure 2); this information is obtained from the semantic index; e.g., NCI:Bronchoscopist and LUCA:Bronchoscopy are disjoint w.r.t. $\mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{M}$:
   - NCI:Bronchoscopist is subsumed by NCI:Occupation ($[4,4]$ contained in $[2,4]$); LUCA:Bronchoscopy is subsumed by NCI:Clinical_activity ($[9,9]$ contained in $[5,10]$); and
   - NCI:Occupation is disjoint with NCI:Clinical_activity.

   Our tool also discards mappings such as $m_1^{\equiv}$ in Figure 2, which, given a reliable $m_r^{\equiv}$ (NCI:Procedure $\equiv$ LUCA:Procedure), map an ancestor of a concept in $m_r^{\equiv}$ (NCI:Clinical_activity) to a descendant of the other one (LUCA:Clinical_procedure).

```
Procedure InteractiveProcess
Input: $SI$: semantic index; $\mathcal{M}_?$: mappings to revise;
Output: $\mathcal{M}_{user}$: user-selected mappings;
 1: (✓) $FilterAmb$ := Filter by Ambiguity?
 2: while $Interact$ = true and $\mathcal{M}_? \neq \emptyset$ do
 3:     $\mathcal{M}_? :=$ ComputePartialOrder($\mathcal{M}_?$)
 4:     (✓) $\langle m, action\rangle :=$ AssessMappingFromList($\mathcal{M}_?$)
 5:     $\mathcal{M}_? := \mathcal{M}_? \setminus \{m\}$
 6:     if ($FilterAmb$ = true) then
 7:         Aux := Ambiguous($m, \mathcal{M}_?$) and $\mathcal{M}_? := \mathcal{M}_? \setminus$ Aux
 8:     end if
 9:     if (action = addition) then
10:         $\mathcal{M}_{user} := \mathcal{M}_{user} \cup \{m\}$
11:         $\mathcal{M}_? := \mathcal{M}_? \setminus$ Conflict($SI \cup \{m\}, \mathcal{M}_?$)
12:     else if ($FilterAmb$ = true) then
13:         $\mathcal{M}_{user} := \mathcal{M}_{user} \cup$ Aux
14:     end if
15:     (✓) $Interact$ := Stop interaction?
16: end while
17: if $\mathcal{M}_? \neq \emptyset$ then $\mathcal{M}_{user} := \mathcal{M}_{user} \cup$ applyHeuristics($\mathcal{M}_?$)
18: return $\mathcal{M}_{user}$
```

Figure 3: LogMap 2 user interaction

2. *Revision of similarity values.* LogMap 2 uses the lexical and semantic indexes to revise the similarity values assigned to candidate mappings (Step 4 in Figure 1). The value for a mapping $A \equiv B$ is refined by taking into account the following:
   - Word co-occurrence in the lexical entries for A and B; e.g., if A is Hepatic_acinus in FMA and B is Liver_acinus in NCI, a string matcher computes a low similarity; however, "Liver" and "Hepatic" frequently co-occur in FMA and NCI concepts (e.g., Hepatic_stem_cell and Liver_stem_cell are synonyms in FMA); based on such analysis, similarity between A and B can be increased. Co-occurrence has been successfully used in information retrieval, as well as for ontology matching (e.g. [23]).
   - Mappings involving superconcepts and subconcepts of A and B; this information can be drawn from the semantic index.

User interaction Many candidate mappings are discarded automatically in the previous step (more than 16,000 for FMA-NCI). The remaining ones, however, are not "clear cut" cases and user feedback would be highly beneficial. The number of such mappings can still be significant (852 such mappings for FMA-NCI); hence, it is crucial to reduce the number of questions to the human expert, on the one hand, and the delay between successive questions, on the other hand.

As discussed in the evaluation section, automatic decisions based on users' feedback can significantly reduce the number of questions in practice. This idea has also been successfully applied to knowledge base revision in ontology engineering (e.g. [3, 20]).

The interaction between LogMap 2 and an expert user is specified in Figure 3, where a (✓) indicates the steps where user intervention is possible. Mappings in $\mathcal{M}_?$ are arranged in a partial order using their (revised) similarity value (Step 3). Questions are asked to users following this partial order, and users can choose to accept or reject the given mapping (Step 4). Users can decide to stop the process (Step 15) so that remaining cases are decided heuristically (Step 17).

Automatic decisions based on a particular user decision to accept or reject a mapping are made according to the following criteria:

- *Ambiguity.* $m = (A \equiv B)$ is ambiguous with $m' = (C \equiv D)$ if $C = A$ or $D = B$ (e.g., FMA:Ethmoid $\equiv$ NCI:Ethmoid_bone is ambiguous with FMA:Ethmoid $\equiv$ NCI:Ethmoid_artery). Ambiguity is a major source of errors in ontology matching [15]; e.g., Ethmoid is either a bone or an artery, but not both. If the user accepts m, and there is only one candidate mapping m' in $\mathcal{M}_?$ that is ambiguous with m, then m' is automatically rejected (Step 7); in contrast, if the user rejects m, then m' is accepted (Step 13). Users have the choice not to use ambiguity criteria for automatic decisions (Step 1), as it may have a negative impact on recall.

- *Conflicts with semantic index.* If a user decides to accept m, then those mappings that are in conflict with the semantic index updated with m (as described in Point 1 in the preceding section) can be safely rejected automatically (Step 11).

Final diagnosis A final reasoning and diagnosis step is performed before returning the output (Steps 12 and 14 in Figure 1). When working in interactive mode, priority is given to human decisions; thus, the mappings $\mathcal{M}_{user}$ selected by the human expert are considered to be fixed, which can result in the deletion of some automatically computed *reliable mappings*. In contrast, when working in non-interactive mode, reliable mappings are considered to be fixed, and thus take precedence over the remaining candidate mappings.

3 EVALUATION

LogMap 2 has been evaluated in both *automatic mode* and *interactive mode* on a laptop with $4Gb$ of RAM. Since LogMap 2 does not yet provide a graphical user interface, we have “simulated” the human expert using Gold Standard mappings to return the correct answer with a given probability. The system is available for download and can also be used directly through a Web interface.[2]

For each operation mode, we have conducted experiments corresponding to the LUCA and UMLS-META use cases. First, we matched LUCA to SNOMED CT (Jan. 2009) and NCI (v. 08.05d). These ontologies contain 395, 306,591 and 66,724 concepts respectively. Precision and recall were evaluated using a curated Gold Standard. We then matched SNOMED-NCI, FMA-SNOMED, and FMA-NCI, where FMA refers to its 2.0 version (78,989 concepts). A clean version of UMLS-META (v. 2009 AA) was used as a Gold Standard [12].

3.1 Automatic Mode

Computation times are summarised in Table 2. LUCA can be matched to NCI and SNOMED CT in one and four minutes, respectively, with parsing (using OWL API) being the most expensive step. SNOMED-NCI was the most challenging case (about 39 minutes) due to the number of unsatisfiable concepts, which made diagnosis expensive; since memory usage was the main issue, we repeated the experiment on a $10Gb$ machine and obtained the output in 15 min.

As already mentioned, only LogMap and GOMMA have also shown to scale with large ontologies. LogMap reports times in line with (but slower than LogMap 2) for FMA-NCI and SNOMED-NCI.[3] Times for FMA-SNOMED are not comparable since LogMap computes a much smaller set of mappings, and thus diagnosis is less expensive. Finally, GOMMA is able to match FMA-NCI in 48 minutes, however it fails to cope with FMA-SNOMED and SNOMED-NCI.

Precision and recall Table 3 summarises our results. In the table, $|\mathcal{M}_{GS}|$ is the number of mappings in the Gold Standard. Precision (**P**), recall (**R**) and F-score (**F**) are given for intermediate steps.

In Step 2, LogMap 2 computes all candidate mappings using the lexical index. As expected, recall values are high, but precision is low. In Step 7, LogMap 2 has computed a clean set of reliable mappings. Precision is high, but recall is now low. LogMap 2 returns in Step 14; precision has decreased slightly in comparison to Step 7, but recall has significantly increased and thus also the F-score.

Finally, the last column of Table 3 indicates the number of unsatisfiable concepts when classifying $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}$ using a DL reasoner, where $\mathcal{M}$ is the output mappings computed by LogMap 2. Although the reasoning algorithms in LogMap 2 are incomplete, LogMap 2 is able to produce a “clean” set of output mappings in many cases. Note that no OWL 2 DL reasoner could cope with SNOMED-NCI [13].

When compared with LogMap, LogMap 2 obtains a better precision and F-score in all cases. Furthermore, recall is dramatically improved for FMA-SNOMED. GOMMA could only be compared for FMA-NCI, where it produces a good F-score (0.81), however, after the integration, more than 5,000 concepts became unsatisfiable.[4]

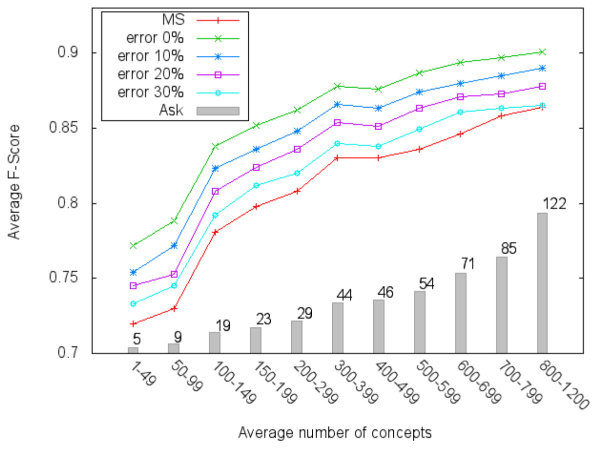

Figure 4: Synthetic experiments for FMA and modules of NCI. Bars represent average number of questions; colored lines represent F-score values for simulated users with variable error rates.

3.2 Interactive Mode

Results are summarised in Table 4 and organised according to whether ambiguity criteria are used to reduce the number of questions. We have simulated users with variable error rate, and observed that results in the non-interactive mode are similar to those obtained with 30% user error rate. Thus, interaction seems to be beneficial if the human expert answers correctly more than 70% of questions.

The LUCA use case The number of questions was surprisingly low for the LUCA use case, which is a most common scenario in practice. Waiting time between questions is negligible, thus making the the interactive process real time. These results suggest the practical feasibility of our approach for many scenarios.

To obtain further empirical evidence, we have also matched more than 1,100 medium-sized modules of NCI to (the whole of) FMA and analysed the number of user questions and the average F-score. Results are given in Figure 4. The number of questions is manageable even for modules with over 1,000 concepts. F-score values obtained in interactive mode for simulated users with up to 30% error rate are better than those obtained in non-interactive mode.

The UMLS-META use case Although the number of questions in this use case was larger (e.g., 2,310 for SNOMED-NCI), it is still rather low given the huge number of candidate mappings (e.g., 102,512 for

[2] `http://www.cs.ox.ac.uk/isg/tools/LogMap/`
[3] Note that times in [12] do not include loading and preprocessing
[4] `http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/`

Table 2: LogMap 2 computation times

Time (s) / Ontologies	Candidate Computation			Candidate Assessment			Total
	Parsing	Lex. Index	Modules	Sem. Index	Conf. Values	Diagnosis	
LUCA-NCI	22.0	20.8	4.9	8.6	6.1	7.9	70.3
LUCA-SNOMED CT	80.6	42.0	19.2	28.2	17.8	60.2	248.0
FMA-NCI	24.5	45.4	8.4	12.1	41.5	114.6	246.5
FMA-SNOMED CT	83.8	96.9	18.9	78.7	409.0	466.5	1155.8
SNOMED CT-NCI	94.4	100.0	17.4	276.5	548.0	1300.7	2338.0

Table 3: Precision and recall w.r.t. Gold Standard ($\mathcal{M}_{GS}$). Steps refer to Figure 1.

Ontologies	$\lvert\mathcal{M}_{GS}\rvert$	$\mathcal{M}_?$ (Step 2)				$\mathcal{M}$ (Step 7)				Output $\mathcal{M}$ (Step 14)				
		$\lvert\mathcal{M}_?\rvert$	**P**	**R**	**F**	$\lvert\mathcal{M}\rvert$	**P**	**R**	**F**	$\lvert\mathcal{M}\rvert$	**P**	**R**	**F**	Unsat.
LUCA-NCI	89	563	0.14	1.00	0.25	63	0.87	0.62	0.72	77	0.82	0.71	0.76	0
LUCA-SNOMED CT	337	783	0.34	0.79	0.47	159	0.98	0.46	0.63	275	0.89	0.72	0.78	0
FMA-NCI	2,898	19,151	0.14	0.93	0.24	2,256	0.91	0.71	0.79	2,658	0.87	0.80	0.83	2
FMA-SNOMED CT	8,111	67,592	0.09	0.74	0.16	4,929	0.84	0.51	0.64	6,313	0.80	0.62	0.70	0
SNOMED CT-NCI	18,322	102,514	0.13	0.75	0.22	10,598	0.86	0.50	0.63	12,978	0.81	0.58	0.67	*

Table 4: Results for interactive mode. "Ask" denotes the total number of questions asked to user, and t denotes the average waiting time between questions (in seconds). Users are simulated at different error rates, with "Perfect" the error-free user.

Ontologies	$FilterAmb$ = true					$FilterAmb$ = false													
	Perfect					**Perfect**				**Error 10%**			**Error 20%**			**Error 30%**			
	Ask	t	**P**	**R**	**F**	Ask	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	
LUCA-NCI	20	0.1	0.86	0.71	0.78	35	0.89	0.75	0.82	0.89	0.75	0.82	0.88	0.74	0.80	0.86	0.71	0.78	
LUCA-SNOMED CT	103	0.2	0.96	0.72	0.82	126	0.99	0.74	0.84	0.98	0.71	0.82	0.97	0.69	0.81	0.96	0.66	0.78	
FMA-NCI	585	<0.1	0.91	0.83	0.86	852	0.92	0.84	0.88	0.90	0.83	0.86	0.87	0.81	0.85	0.88	0.80	0.84	
FMA-SNOMED CT	946	0.4	0.85	0.59	0.70	1,595	0.87	0.63	0.73	0.86	0.61	0.72	0.84	0.60	0.70	0.83	0.59	0.69	
SNOMED CT-NCI	2,310	0.9	0.87	0.57	0.69	3,306	0.88	0.58	0.70	0.87	0.58	0.69	0.85	0.57	0.68	0.84	0.56	0.67	

SNOMED-NCI). Furthermore, waiting time between questions was also negligible. F-score values in the automatic mode are in line to those obtained in the interactive mode for users with 30% error rate; improvements are observed especially in precision.

4 CONCLUSION

We have presented a matching system with unique interactivity, scalability and reasoning-based diagnosis features. Empirical results on large-scale ontologies for two typical application scenarios suggest the practical feasibility of our approach. The main challenge remains the design of suitable interfaces to support user interaction and their evaluation via a pilot user study.

Acknowledgements. Funded by The Royal Society, the FP7 project SEALS and the EPSRC projects ConDOR, ExODA and LogMap.

REFERENCES

[1] B. Alexe, L. Chiticariu, R. J. Miller, and W. C. Tan, 'Muse: Mapping understanding and design by example', in *Proc. of ICDE*, (2008).

[2] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm, 'Schema and ontology matching with COMA++', in *SIGMOD Conference*, (2005).

[3] F. Baader, B. Ganter, B. Sertkaya, and U. Sattler, 'Completing description logic knowledge bases using formal concept analysis', in *Proc. of IJCAI*, pp. 230–235. AAAI Press, (2007).

[4] O. Bodenreider, 'The Unified Medical Language System (UMLS): integrating biomedical terminology.', *Nuc. acids res.*, **32**, (2004).

[5] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis, 'On labeling schemes for the Semantic Web', in *Proc. of WWW*, (2003).

[6] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler, 'Just the right amount: extracting modules from ontologies', in *WWW*, (2007).

[7] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler, 'OWL 2: The next step for OWL', *J. Web Sem.*, (2008).

[8] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, and C. Trojahn, 'Ontology Alignment Evaluation Initiative: six years of experience', *J Data Semantics*, (2011). http://oaei.ontologymatching.org/.

[9] S. M. Falconer and N. F. Noy, 'Interactive techniques to support ontology matching', in *Schema Matching and Mapping*, (2011).

[10] J. Huber, T. Sztyler, J. Noessner, and C. Meilicke, 'CODI: Combinatorial optimization for data integration.', in *Proc. of OM-OAEI*, (2011).

[11] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka, 'Ontology matching with semantic verification', *J. Web Sem.*, (2009).

[12] E. Jiménez-Ruiz and B. Cuenca Grau, 'LogMap: Logic-based and scalable ontology matching', in *Proc. of ISWC*, (2011).

[13] E. Jiménez-Ruiz, B. Cuenca Grau, and I. Horrocks, 'On the feasibility of using OWL 2 DL reasoners for ontology matching problems', in *OWL Reasoner Evaluation (ORE) Workshop*, (2012).

[14] E. Jimenez-Ruiz, B. Cuenca Grau, I. Horrocks, and R. Berlanga, 'Ontology integration using mappings: Towards getting the right logical consequences', in *Proc. of ESWC*, (2009).

[15] E. Jiménez-Ruiz, B. Cuenca Grau, I. Horrocks, and R. Berlanga, 'Logic-based assessment of the compatibility of UMLS ontology sources', *J. Biomed. Sem.*, **2**, (2011).

[16] T. Kirsten, A. Gross, M. Hartung, and E. Rahm, 'GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution', *J. Biomed. Sem.*, **2**, 6, (2011).

[17] B. Konev, C. Lutz, D. Walther, and F. Wolter, 'Semantic modularity and module extraction in description logics', in *Proc. of ECAI*, (2008).

[18] C. Meilicke, *Alignment Incoherence in Ontology Matching*, Ph.D. dissertation, University of Mannheim, 2011.

[19] C. Meilicke, H. Stuckenschmidt, and O. Sváb-Zamazal, 'A reasoning-based support tool for ontology mapping evaluation', in *ESWC*, (2009).

[20] N. Nikitina, S. Rudolph, and B. Glimm, 'Interactive ontology revision', *J. Web Sem.*, (2012).

[21] N. F. Noy and M. A. Musen, 'PROMPT: Algorithm and tool for automated ontology merging and alignment', in *Proc. of AAAI*, (2000).

[22] Q. Reul and J. Z. Pan, 'KOSIMap: Use of description logic reasoning to align heterogeneous ontologies', in *Proc. of DL*, (2010).

[23] M. A. Rodríguez and M. J. Egenhofer, 'Determining semantic similarity among entity classes from different ontologies', *IEEE Trans. Knowl. Data Eng.*, **15**, 442–456, (2003).

[24] B. Sesen, R. Banares-Alcantara, J. Fox, T. Kadir, and J. M. Brady, 'Lung Cancer Assistant: An ontology-driven, online decision support prototype for lung cancer treatment selection', in *OWL-ED*, (2012).

[25] P. Shvaiko and J. Euzenat, 'Ontology matching: State of the art and future challenges', *IEEE Trans. Knowl. Data Eng.*, **99**, (2011).