

Infrastructure for Dynamic Knowledge Integration – Automated Biomedical Ontology Extension Using Textual Resources

Vít Nováček, Loredana Laera, Siegfried Handschuh, Brian Davis^{a,b,a,a}

^a*Digital Enterprise Research Institute
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland*

^b*Department of Computer Science
University of Liverpool, UK*

Abstract

We present a novel ontology integration technique that explicitly takes the dynamics and data-intensiveness of e-health and biomedicine application domains into account. Changing and growing knowledge, possibly contained in unstructured natural language resources, is handled by application of cutting-edge Semantic Web technologies. In particular, semi-automatic integration of ontology learning results into a manually developed ontology is employed. This integration bases on automatic negotiation of agreed alignments, inconsistency resolution and natural language generation methods. Their novel combination alleviates the end-user effort in the incorporation of new knowledge to large extent. This allows for efficient application in many practical use cases, as we show in the paper.

Key words: dynamic ontology integration, ontology evolution, ontology alignment and negotiation, ontology learning, biomedical ontologies, knowledge acquisition, lifecycle

1 Introduction

Ontologies (formal knowledge bases) on the Semantic Web are often very likely subject to change given the dynamic nature of domain knowledge. Knowledge

Email address: vit.novacek@deri.org, lori@csc.liv.ac.uk, siegfried.handschuh@deri.org, brian.davis@deri.org (Vít Nováček, Loredana Laera, Siegfried Handschuh, Brian Davis).

changes and evolves over time as experience accumulates – it is revised and augmented in the light of deeper understanding; new facts are becoming known while some of the older ones need to be revised and/or retracted at the same time. This holds especially for scientific domains, however, even virtually any industrial domain is dynamic – changes typically occur in product portfolios, personnel structure or industrial processes, which can all be reflected by an ontology in a knowledge management policy.

The domains of e-health and biomedicine are both scientific (biomedical research) and industrial (clinical practice, pharmaceuticals). The need for ontologies in biomedicine knowledge and data management has already been recognised by the community. Ontologies can serve as structured repositories giving a shared meaning to data and thus making it possible to process and query them in a more efficient and expressive manner. The shared meaning provided by ontologies also results in facilitation of integration between different medical data formats once they are bound to an ontology. Moreover, the state of the art ontology-based techniques (like alignment or reasoning as described in [39]) can help to integrate the data even if they adhere to different ontologies.

In the biomedical domain, ontology construction is usually a result of a collaboration involving ontology engineers and domain experts, where the knowledge is being extracted and modelled manually. However, it is not always feasible to process all the relevant data and extract the knowledge manually from domain resources, since we might not have a sufficiently large committee of ontology engineers and/or dedicated experts at hand in order to process new data anytime it arrives. This implies a need for automation of knowledge extraction and maintenance processes in dynamic and data-intensive medical environments. If the knowledge is available in textual resources, ontology learning (see [33]) can help in this task. Therefore, a lifecycle of an ontology development process apt for universal application in the medicine domain should also support appropriate mechanisms for the incorporation of dynamically extracted knowledge. In this paper, we introduce such a lifecycle scenario and a novel solution to the dynamic knowledge integration task.

Our efforts have several particular **motivations**. While there has been a great deal of work on ontology learning for ontology construction, e.g. in [7], as well as on manual or collaborative ontology development in [41], relatively little attention has been paid to the user-friendly integration of both approaches within an ontology lifecycle scenario. By user-friendly we mean especially accessible to users who are not experts in ontology engineering (i.e. biomedicine researchers or practitioners). In this paper, we introduce our framework for practical handling of dynamic and large data-sets in an ontology lifecycle, focusing particularly on dynamic integration of learned knowledge into manually maintained ontologies. However, the introduced integration mechanism is not

restricted only to learned ontologies – arbitrary “external” ontology can be integrated into the primary ontology in question by the very same process.

The dynamic nature of knowledge is one of the most challenging problems not only in biomedicine, but in the whole current Semantic Web research. Here we provide a solution for dealing with these dynamics on a large scale, based on the properly developed connection of ontology learning and dynamic manual development. We do not concentrate on formal specification of respective ontology integration operators, we focus rather on implementation of them, following certain practical requirements:

- (1) the ability to process new knowledge (resources) automatically whenever it appears and when it is inappropriate for human users to incorporate it
- (2) the ability to automatically compare the new knowledge with a “master” ontology that is manually maintained and select the new knowledge accordingly
- (3) the ability to resolve possible major inconsistencies between the new and current knowledge, possibly favouring the assertions from presumably more complex and precise master ontology against the learned ones
- (4) the ability to automatically sort the new knowledge according to user-defined preferences and present it to them in a very simple and accessible way, thus further alleviating human effort in the task of knowledge integration

On one hand, using the automatic methods, we are able to deal with large amounts of changing data. On the other hand, the final incorporation of new knowledge is to be decided by the expert human users, repairing possible errors and inappropriate findings of the automatic techniques. The key to success and applicability is to let machines do most of the tedious and time-consuming work and provide people with concise and simple suggestions on ontology integration.

The main **contribution** of the presented work is two-fold:

- proposal and implementation of a generic algorithm for dynamic integration of knowledge automatically extracted from various unstructured resources (e.g., natural language articles or web pages) into manually maintained formal ontologies (described in Sections 4 and 5)
- presentation of an example application of the implemented algorithm in a task of biomedical ontology extension by integrating knowledge automatically learned from textual domain resources, showing usability of the approach in the context of the presented use cases (Section 6)

The rest of the paper is organized as follows: Section 2 gives basic overview of the essential notions and background of the paper, together with respective relevant references. Section 3 discusses the related work. Section 4 gives an

overview of our ontology lifecycle scenario and framework, whereas Section 5 presents the integration of manually designed and automatically learned ontologies in more detail. In Section 6, we describe an example practical application of our integration technique, using real world input data (from the biomedicine research domain). Preliminary evaluation and its discussion is also provided. Section 7 outlines relevant real-world settings, challenges and contributions our framework can bring in these contexts. A related user feedback analysis is provided in Section 7, too. Section 8 summarises the paper and future directions of the presented research.

2 Key Notions

In the following list we give a brief description of the essential notions that are relevant for the presented content and describe how they relate to the field of bioinformatics:

- **Semantic Web** – the Semantic Web initiative is generally about giving a formal shared meaning to the data present on the normal world wide web in order to make them fully accessible and “comprehensible” by machines, not only by humans (see [5]). However, the technologies that have been developed within Semantic Web research are applicable to many other fields. In the case of bioinformatics, biomedical data management and decision support, we can exploit for instance Semantic Web methods of intelligent and efficient knowledge representation, reasoning, data integration or knowledge management.
- **ontology** – according to a popular definition in [24], ontology is a representation of shared conceptualisation. As such, ontologies are used for formal representation of knowledge in particular domains, i.e. various subfields of biomedicine.
- **ontology integration** – the process of merging, consolidating and respective analysis and modification of two or more ontologies into one (integrated) ontology (see [38]). The process can be either manual or (semi)automatical.
- **ontology learning** – acquisition of an ontology from unstructured or semi-structured natural language text, typically resources relevant for a particular domain (e.g. web pages, articles or other types of documents). Natural Language Processing and Machine Learning methods are mostly used as a base for ontology learning algorithms (see [33]).
- **ontology alignment** – ontology alignment establishes mappings between concepts and other entities (e.g. relations or instances) in two or more ontologies. Either manually designed mappings (created on the fly or contained in appropriate alignment repositories), or automatically generated ones can be used to align the ontologies (see [17, 16]).

- **ontology evolution** – development and maintenance of ontologies in dynamic environments (see [40, 37, 25]), where the knowledge needs to be updated on regular basis and changes in the domain conceptualisation occur often (e.g. science or business domains, where frequent introduction of new concepts or revision of the old ones is essential).
- **ontology lifecycle** – a methodology or scenario, that describes how the particular phases of the ontology development, maintenance and possibly also exploitation are mutually connected and dependent (see [23, 35]).

3 Related Work

Within the Semantic Web research, several approaches and methodologies have been defined and implemented in the context of ontology lifecycle and integration. Recent overviews of the state-of-the-art in ontologies and related methodologies can be found in [39] and [23]. However, none of them offers a direct solution to the requirements specified in Section 1.

The *Methontology* methodology by [19] was developed in the *Esperanto* EU project. It defines the process of designing ontologies and extends it towards evolving ontologies. It is provided with an ontology lifecycle based on evolving prototypes (see [20]) and defines stages from specification and knowledge acquisition to configuration management. The particular stages and their requirements are characterised, but rather in a general manner. The automatic ontology acquisition methods are considered in *Methontology*, however, their concrete incorporation into the whole lifecycle is not covered. The ODESeW and WebODE (see [10]) projects base on *Methontology* and provide an infrastructure and tools for semantic application development/management, which is in the process of being extended for networked and evolving ontologies. However, they focus rather on the application development part of the problem than on the ontology evolution and dynamic ontology integration parts.

The methods and tools referenced above lack concrete mechanisms that would efficiently deal with the dynamics of realistic domains (so characteristic for instance for e-health and biomedicine). Moreover, the need for automatic methods of ontology acquisition in data-intensive environments is acknowledged, but the role and application of the automatic techniques is usually not clearly studied and implemented. Our approach described in [35] offers a complex picture of how to deal with the dynamics in the general lifecycle scenario. The work we present here implements the fundamental semi-automatic dynamic integration component of the scenario.

There are more specific approaches similar to the one presented by our lifecycle framework. [14] incorporates automatic ontology extraction from a medical

database and its consequent population by linguistic processing of corpus data. However, the mechanism is rather task-specific – the ontology is represented in RDF(S) format (see [6]) that is less expressive than the OWL language (see [4]), which we use. The extraction is oriented primarily at taxonomies and does not take the dynamics directly into account. Therefore the approach can hardly be applied in universal settings, which is one of our aims.

Protégé (see [22]) and related PROMPT (see [36]) tools are designed for manual ontology development and semi-automatic ontology merging, respectively. PROMPT provides heuristic methods for identification of similarities between ontologies. The similarities are offered to the users for further processing. However, the direct connection to ontology learning, which we find important for dynamic and data-intensive domains like e-health and biomedicine, is missing.

There are several works addressing directly the topic of ontology integration. [1] and [8] describe two approaches inspired mainly by database techniques of data mediation and query rewriting in order to provide integrated (global) view on several (local) ontologies. [28] present web ontology integration method using SHOE, a web-based knowledge representation language, and semi-automatically generated alignments. [12] implement a dynamic and automatic ontology integration technique in multi-agent environments, based on relatively simple graph ontology model inclusions and other operations. Again, none of the approaches tackles the requirements we specify in Section 1. Even though the methods propose solutions to the integration problem in general, there is no direct way how to integrate knowledge from unstructured resources, minimising human intervention. Furthermore, there is no emphasis on accessibility of the ontology integration to the laymen users. Our approach is distinguished by the fact that it pays special attention to these features, which we find essential for the application in e-health and/or bioinformatics.

4 DINO – A Dynamic Ontology Lifecycle Scenario

Our integration platform is a part of a broader lifecycle scenario (see [35]). We refer to both lifecycle and integration platform by the DINO abbreviation, evoking multiple features of our solution: it reflects three key elements of the lifecycle scenario – *Dynamics*, *INtegration* and *Ontology*; however, the first two parts can also be *Data* and *INtensive*; finally, DINO can be read as *Dynamic INtegration* of *Ontologies*, too. All these features express the primary aim of our efforts – to make the knowledge (integration) efficiently and reasonably manageable in data-intensive and dynamic domains.

Figure 1 depicts the scheme of the proposed dynamic and application-oriented ontology lifecycle that deals with the problems mentioned as a part of our mo-

tivations. Our ontology lifecycle builds on four basic phases of an ontology life-

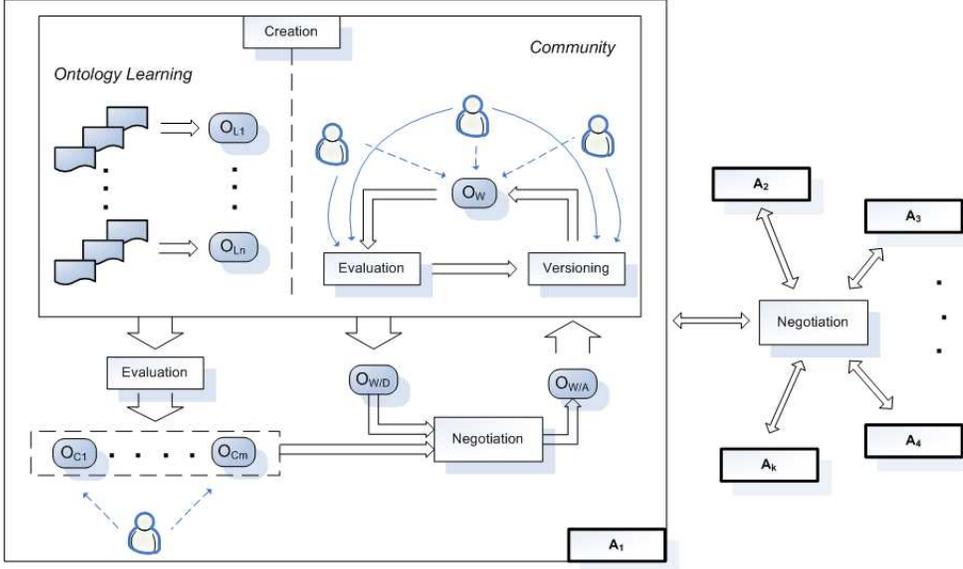


Fig. 1. Dynamic ontology lifecycle scheme

cycle: *creation* (comprises both manual and automatic ontology development and update approaches), *versioning*, *evaluation* and *negotiation* (comprises ontology alignment and merging as well as negotiation among different possible alignments). The four main phases are indicated by the boxes annotated by respective names. Ontologies or their snapshots in time are represented by circles, with arrows expressing the information flow and transitions between them. The boxes labelled A_i present actors (institutions, companies, research teams etc.) involved in ontology development, where A_1 is zoomed-in in order to show the lifecycle's components in detail.

The general dynamics of the lifecycle goes as follows: (1), the community experts and/or ontology engineers develop a relatively precise and complex domain ontology (the *Community* part of the *Creation* component); (2), the experts use means for continuous ontology *evaluation* and *versioning* to maintain high quality and manage changes during the development process, respectively; (3), if the amount of data suitable for knowledge extraction (e.g. domain resources in natural language) is too large to be managed by the community, *ontology learning* takes its place; (4), the ontology learning results are *evaluated* by human experts and eventually integrated (using the *negotiation* component) into the more precise reference community ontology, if the respective extensions have been found appropriate.

The integration in the scenario is based on alignment and merging covered by the *negotiation* component. Its proposal, implementation principles and application in selected e-health use case form the key contribution of this paper (see Sections 5 and 6 for details). The *negotiation* component takes its place

also when interchanging or sharing the knowledge with other independent actors in the field. All the phases support ontologies in the standard OWL format. In the following we will concentrate on the integration mechanism. More information on other parts of the lifecycle can be found in [35].

5 Dynamic Integration of Automatically Learned Knowledge

The key novelty of the presented lifecycle scenario is its support for incorporation of changing knowledge in data-intensive domains, especially when unstructured data (i.e. natural language) is involved. This is achieved by implementation of a specific integration mechanism introduced in this section. The scheme of the integration process is depicted in Figure 2.

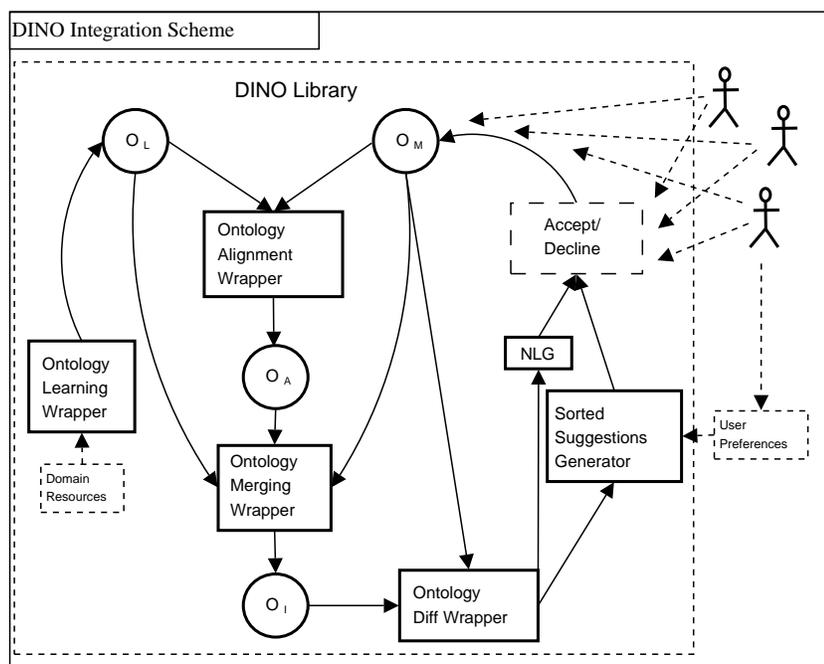


Fig. 2. Dynamic ontology integration scheme

The integration scheme details the combination of several generic lifecycle components—mainly the (automatic) *creation* and *negotiation*—in the process of incorporation of learned ontologies into a collaboratively developed one. The latter ontology serves as a master, presumably precise model in the process of learned knowledge integration.

The master ontology – O_M circle in Figure 2 – is supposed to be developed within a dedicated external application such as Protégé¹. The DINO inte-

¹ See <http://protege.stanford.edu/>.

gration platform itself is implemented as a respective API library and GUI interface. Simple research prototypes of these applications and user documentation can be downloaded at <http://smile.deri.ie/tools/dino>.

O_M in Figure 2 presents a reference for integration with the O_L ontology resulting from the learning process. Ontology Alignment Wrapper produces an alignment ontology O_A that encodes mappings between O_M and O_L . All these ontologies are passed to the Ontology Merging Wrapper that resolves possible inconsistencies and produces integrated ontology O_I . Ontology Diff Wrapper compares O_I with the former master ontology O_M and passes the respective additional statements (not present in O_M) to the NLG and Sorted Suggestions Generator component. NLG (Natural Language Generator) produces a comprehensive natural language representation of all the addition statements. The Sorted Suggestions Generator component outputs the final product of the integration process – particular natural language suggestions on the master ontology extension, sorted according to the user preferences. The suggestions agreed by human users form a base of a next version of the O_M ontology created after the integration. Note that during all phases of integration, we use the former O_M base namespace for all the other ontologies involved. The integration phases outlined in Figure 2 are described in detail in the sections below.

5.1 *Ontology Learning Wrapper*

In this phase, machine learning and NLP methods are used for the processing of relevant resources and extracting knowledge from them (ontology learning). The ontology learning is realised using the Text2Onto framework (see [9]) that is able to extract an ontology from an arbitrary set of textual documents. Due to space restrictions, we cannot properly comment on the methods used for ontology extraction and post-processing in Text2Onto, however, they are described in detail in [32, 9]. Note that this component does not tackle selection of the documents the ontology is to be learned from – this task needs to be performed manually by the system users.

In the current implementation, only a restricted subset of possible OWL (DL) constructs is being extracted: `rdfs:subClassOf` axioms, class instances, named class assertions, `owl:disjointWith` axioms and `owl:ObjectProperty` assertions with `rdfs:domain` and `rdfs:range` properties specified. `owl:equivalentClass` relations can be inferred from mutual `rdfs:subClassOf` axioms between particular classes. `owl:equivalentClass` and `owl:sameAs` constructs can also be extracted using the Text2Onto concept/instance similarity determination algorithms, however, their performance, precision and coverage was not found to be sufficient enough, therefore they are not included in the current

version of the DINO framework.

We have not performed a rigorous evaluation of the ontology learning step as such, however, the informal precision rate of ontology extraction was about 70% for the sample application described in Section 6 (given by ratio of meaningful axioms to all extracted axioms). Note that even an arbitrary external ontology can be integrated instead of the learned one, however, the integration results are not necessarily complete in the case of more complex ontologies (e.g., containing complex restrictions and anonymous classes). This is due to the fact that the current implementation is tailored specifically to the rather simple learned ontologies.

5.2 *Ontology Alignment Wrapper*

When the learned ontology O_L has been created, it has to be reconciled with master ontology O_M since they cover the same domain, but might be structured differently. The reconciliation of these ontologies depends on the ability to reach an agreement on the semantics of the terms used. The agreement takes the form of an alignment between the ontologies, that is, a set of correspondences (or mappings) between the concepts, properties, and relationships in the ontologies. However, the ontologies are developed in different contexts and under different conditions and thus they might represent different perspectives over similar knowledge, so the process by which to come to an agreement will necessarily only come through a negotiation process. The negotiation process is performed using argumentation-based negotiation that uses preferences over the types of correspondences in order to choose the mappings that will be used to finally merge the ontologies (see Section 5.3). The preferences depend on the context and situation. A major feature of this context is the ontology, and the structural features thereof, such as the depth of the subclass hierarchy and branching factor, ratio of properties to concepts, etc. The analysis of the components of the ontology is aligned with the approach to ontology evaluation, demonstrated in [13], and can be formalized in terms of feature metrics. Thus the preferences can be determined on the characteristics of the ontology. For example, we can select a preference for terminological mapping if the ontology is lacking in structure, or prefer extensional mapping if the ontology is rich in instances.

Thus, the alignment/negotiation wrapper interfaces two tools – one for the ontology alignment discovery and one for negotiation of agreed alignment. We call these tools *AKit* and *NKit*, respectively, within this section. For the former, we use the ontology alignment API (see [16]) developed by INRIA Rhone-Alpes². For the negotiation we use the framework described in [30].

² See <http://alignapi.gforge.inria.fr/> for up-to-date information on the API.

Both tools are used by the wrapper in order to produce O_A – an ontology consisting of axioms³ merging classes, individuals and properties in the O_L and O_M ontologies. It is used in consequent factual merging and refinement in the ontology reasoning and management wrapper (see Section 5.3 for details).

The wrapper itself works according to the meta-code in Algorithm 1. The on-

Algorithm 1 Meta-algorithm of the alignment and negotiation

Require: O_L, O_M — ontologies in OWL format

Require: $AKit, NKit$ — ontology alignment and alignment negotiation tools, respectively

Require: $ALMSET$ — a set of the alignment methods to be used

Require: $PREFSET$ — a set of alignment formal preferences corresponding to the O_L, O_M ontologies (to be used in N-kit)

```

1:  $S_A \leftarrow \emptyset$ 
2: for  $method \in ALMSET$  do
3:    $S_A \leftarrow S_A \cup AKit.getAlignment(O_L, O_M, method)$ 
4: end for
5:  $A_{agreed} \leftarrow NKit.negotiateAlignment(S_A, PREFSET)$ 
6:  $O_A \leftarrow AKit.produceBridgeAxioms(A_{agreed})$ 
7: return  $O_A$ 

```

tology alignment API offers several possibilities of actual alignment methods, which range from trivial lexical equality detection through more sophisticated string and edit-distance based algorithms to an iterative structural alignment by the OLA algorithm (see [18]). The ontology alignment API has recently been extended by a method for the calculation of a similarity metric between ontology entities, an adaptation of the SRMetric used in [43]. We also consider a set of justifications, that explain why the mappings have been generated. This information forms the basis for the negotiation framework that dynamically generates arguments, supplies the reasons for the mapping choices and negotiates an agreed alignment for both ontologies O_L and O_M .

5.3 Ontology Merging Wrapper

This wrapper is used for merging of the O_L and O_M ontologies according to the statements in O_A (each of the ontologies technically represented as a respective Jena ontology model). Moreover, the wrapper resolves possible inconsistencies caused by the merging – favouring the assertions in the O_M ontology, which are supposed to be more relevant. The resulting ontology O_I is passed to the ontology diff wrapper to be compared with the former O_M master ontology. The respective addition model forms a basis for the natural language suggestions that are produced as a final product of the integration (see Sections 5.4 and 5.5 for details).

³ Using constructs like *owl:equivalentClass*, *owl:sameAs*, *owl:equivalentProperty*, *rdfs:subClassOf* or *rdfs:subPropertyOf*.

Algorithm 2 describes the meta-code of the process arranged by the ontology merging and reasoning wrapper. We currently employ no reasoning in

Algorithm 2 Meta-algorithm of the merging and inconsistency resolution

Require: O_L, O_M, O_A — ontologies in OWL format

Require: $merge()$ — a function that merges the axioms from input ontologies, possibly implementing reasoning routines according to the ontology model used

Require: C — set of implemented consistency restrictions; each element $r \in C$ can execute two functions $r.detect()$ and $r.resolve()$ that detect (and return) and resolve an inconsistency in the input ontology, respectively

```

1:  $O_I \leftarrow merge(O_M, O_L, O_A)$ 
2:  $inconsistencies \leftarrow \emptyset$ 
3: for  $r \in C$  do
4:    $inconsistencies \leftarrow inconsistencies \cup r.detect(O_I)$ 
5:    $O_I \leftarrow r.resolve(O_I)$ 
6: end for
7: return  $O_I, inconsistencies$ 

```

the $merge()$ function. However, sub-class subsumption (as implemented by the Jena framework) is used when detecting and resolving inconsistencies. The inconsistencies are constituted by user-defined restrictions. These restrictions are implemented as extensions of a generic inconsistency detector and resolver in the ontology merging wrapper. Thus we can implement either logical (in terms of Description Logics, see [2]) inconsistencies, or custom-defined inconsistencies (i.e. cyclic definitions) according to requirements of particular practical applications.

The automatic inconsistency resolution itself is somewhat tricky. However, we can apply a sort of “greedy” heuristic, considering the assertions in the master O_M ontology to be more valid. Therefore we can discard axioms from O_L or O_A that are inconsistent with axioms in O_M – we call such axioms *candidate* in the text below. If there are more such axioms, we discard them one by one randomly until the inconsistency is resolved⁴. If all the conflicting axioms originated in O_M , we just report them without resolution.

We currently implement and resolve the following inconsistencies:

- **sub-class hierarchy cycles:** these are resolved by cutting the cycle, i.e. removing a candidate *rdfs:subClassOf* statement;
- **disjointness-subsumption** conflicts: if classes are said to be disjoint and a sub-class relationship holds between them at the same time, a candidate conflicting assertion is removed;
- **disjointness-superclass** conflicts: if a class is said to be a sub-class of classes that are disjoint, a candidate conflicting assertion is removed;

⁴ This is the currently implemented way, however, we plan to improve the selection of candidate axioms according to confidence ranking produced by the Text2Onto tool – similarly to the technique described in [26]. This is scheduled for the next version of the DINO integration library.

- **disjointness-instantiation** conflicts (specialisation of the above): if an individual is said to be an instance of classes that are disjoint, a candidate conflicting assertion is removed.

The first one is non-logical inconsistency, whereas the remaining free are examples of logical inconsistencies. More on the types and nature of logical (DL) inconsistencies can be found for instance in [21]. Since most logical inconsistencies are introduced by negative constructs like `owl:disjointWith`, `owl:complementOf` or `owl:differentFrom`, we can easily adapt the above techniques related to disjointness in order to support additional inconsistency types.

A transparent and flexible support of arbitrary non-logical consistency constraints is a part of our future work. We plan to implement this feature on the top of user-defined rules (expressing facts like “if X is a male mammal, then it does not have an ovary”). DINO will not include the learned statements that are in a conflict with the respective rule-based constraints into the merged ontology. Certain more subtle issues related to the ontology design (such as possibly unwelcome multiple inheritance) cannot, however, be generally handled even by the rule-based inconsistency resolution, therefore the more sophisticated refinement of the integrated ontology is deliberately left for the user.

Note that each element of the set of inconsistencies returned by Algorithm 2 (besides the integrated ontology itself) is associated with respective simple natural language description. The descriptions are presented for further examinations by human users in the DINO user interface.

5.4 *Ontology Diff Wrapper*

Possible extension of a master ontology O_M by elements contained in the merged and refined ontology O_I naturally corresponds to the differences between them. In particular, the possible extensions are equal to the additions O_I brings into O_M . The additions can be computed in several ways. Ontology diff wrapper in DINO offers a way how to uniformly interface the particular methods of addition computation. No matter which underlying method is employed, a respective Jena ontology model containing the respective additions is returned. Currently, the following methods are implemented within the wrapper:

- (1) SemVersion-based diff computation – additions at the RDF (triple) level computed using the SemVersion library (see [44])
- (2) addition model computation by set operations on the underlying Jena RDF models

- (3) addition model computation by direct iterative querying of the former master ontology model, integrated model and alignment model for reference purposes (see Algorithm 3 for details on implementation)

For the practical experiments with ontologies, we have used the third method – mainly due to the fact that it computes the additions directly at the ontology level and not at the lower triple level (which means subsequent processing load when getting back to the ontology model again).

Algorithm 3 Meta-algorithm of the addition model computation (by direct model querying)

Require: O_M, O_I, O_A — former master, integrated and alignment ontologies, respectively
Require: $copyResource()$ — a function that returns a copy of an ontology resource (e.g. class or property) including all relevant features that are bound to it (e.g. subclasses, superclasses, instances for a class or domain and range for a property)

```

1:  $O_{added} \leftarrow \emptyset$ 
2: for  $c \in O_I.getNamedOntologyClasses()$  do
3:   if not  $O_M.contains(c)$  or  $O_A.contains(c)$  then
4:      $O_{added} \leftarrow copyResource(c)$ 
5:   end if
6: end for
7: for  $p \in O_I.getOntologyProperties()$  do
8:   if not  $O_M.contains(p)$  or  $O_A.contains(p)$  then
9:      $O_{added} \leftarrow copyResource(p)$ 
10:  end if
11: end for
12: return  $O_{added}$ 

```

Note that the algorithm does not compute all differences between arbitrary ontologies in general. However, this is no drawback for the current implementation of DINO integration. We deal with learned ontology extending the master one. The extensions originating in automatically learned knowledge do not cover the whole range of possible OWL constructs, thus we do not need to tackle e.g. anonymous classes and restrictions in the addition model computation. Therefore the employed custom addition computation can be safely applied without any loss of information. The computed addition ontology model is passed to the suggestion sorter then (see Section 5.5 for details).

5.5 Sorted Suggestions Generator

The addition ontology passed to this component forms a base for the eventual extension suggestions for the domain experts. In order to reduce the effort in the final reviewing of the master ontology extensions, we create respective simple natural language suggestions that are associated with corresponding facts in the addition ontology model. The natural language suggestions are then presented to users – when a suggestion is accepted by the users, the associated fact is included into the master ontology model. Table 1 shows a scheme of the natural language (NL) suggestion generation. The r variable represents possible relations between classes or properties (e.g. `rdfs:subClassOf`,

Table 1
Scheme of suggestion generation

Axiom pattern	NL suggestion scheme	Example
class c_1 is related by relation r to class c_2	The class $c_1.label()$ $f(r)$ the class $c_2.label()$.	The class "difference_c" is disjoint with the class "inclusion_c".
individual i is a member of class c	The class $c.label()$ has the $i.label()$ instance.	The class "the_cytoskeleton_organiser_c" has the "centrosome_i" instance.
property p_1 with features x is related to property p_2 by relation r	There is a $p_1.label()$ $g(x)$ property. It is $f(r)$ $p_2.label()$.	There is a "contain_r" object property. Its range is the "organ_c" class.
property p_1 with features x has domain/range class c	There is a $p_1.label()$ $g(x)$ property. Its domain/range is the $c.label()$ class.	There is a "contain_r" object property. It has the "has_part_r" superproperty.

`rdfs:subPropertyOf` or `owl:disjointWith`), mapped by the function $f()$ to a respective natural language representation (e.g. *is a sub-class of*, *is a sub-property of* or *is disjoint with*). The x variable represents possible features of a property (e.g. `owl:ObjectProperty` or `owl:FunctionalProperty`, mapped by the function $g()$ to a respective natural language representation (e.g. *object* or *functional*).

In general, the number of suggestions originating from the addition ontology model can be quite large, so an ordering that takes a relevance measure of possible suggestions into account is needed. Thus we can for example eliminate suggestions with low relevance level when presenting the final set to the users (without overwhelming them with a large number of possibly irrelevant suggestions). As a possible solution to this task, we have proposed and implemented a method based on string subsumption and a specific distance measure (see [31]). These two measures are used within relevance computation by comparing the lexical labels occurring in a suggestion with respect to two sets S_p, S_n of words, provided by users. The S_p and S_n sets contain preferred and unwanted words respectively, concerning the lexical level of optimal extensions. The suggestions T are sorted according to the respective $rel(T, S_p) - rel(T, S_n)$ values, where $rel(T, S)$ is a function measuring the relevance of the suggestion triple T with respect to the words in the set S . The higher the value, the more relevant the suggestion triple is. We develop the relevance function in detail in Algorithm 4.

The function naturally measures the "closeness" of the labels occurring in the suggestion to the set of terms in S . The value of 1 is achieved when the label is a direct substring of or equal to any word in S or vice versa. When the Levenshtein distance between the label and a word in S is lower than or equal to the defined threshold t , the relevance decreases from 1 by a value proportional to the fraction of the distance and t . If this is not the case (i.e. the label's distance is greater than t for each word in S), a similar principle is applied for possible word-parts of the label and the relevance is further

Algorithm 4 The relevance function

Require: S_t — a set of (possibly multiword) lexical terms occurring in the suggestion

Require: S — set of words

Require: $\rho \in (0, 1)$ influences the absolute value of relevance measure

Require: t — integer constant; maximal allowed distance

Require: $levDist(s_1, s_2)$ — Lev. distance implementation

```
1: for  $elem \in S_t$  do
2:    $R_{elem} \leftarrow 0$ 
3: end for
4: for  $elem \in S_t$  do
5:   if  $elem$  is a substring of or equals to any word in  $S$  or vice versa then
6:      $R_{elem} \leftarrow 1$ 
7:   else
8:      $d \leftarrow \infty$ 
9:     for  $v \in S$  do
10:      if  $levDist(elem, v) < d$  then
11:         $d \leftarrow levDist(elem, v)$ 
12:      end if
13:    end for
14:    if  $d \leq t$  then
15:       $R_{elem} \leftarrow (1 - \frac{d}{t+1})$ 
16:    else if  $elem$  is a multiword term then
17:       $L \leftarrow$  set of single terms in the  $elem$  label expression
18:       $EXP \leftarrow 0$ 
19:      for  $u \in L$  do
20:        if  $u$  is a substring of or equals to any word in  $S$  or vice versa then
21:           $EXP \leftarrow EXP + 1$ 
22:        else
23:           $d \leftarrow \infty$ 
24:          for  $v \in S$  do
25:            if  $levDist(u, v) < d$  then
26:               $d \leftarrow levDist(u, v)$ 
27:            end if
28:          end for
29:          if  $d \leq t$  then
30:             $EXP \leftarrow EXP + (1 - \frac{d}{t+1})$ 
31:          end if
32:        end if
33:      end for
34:      if  $EXP = 0$  then
35:         $R_{elem} \leftarrow 0$ 
36:      else
37:         $R_{elem} \leftarrow \rho \frac{1}{EXP}$ 
38:      end if
39:    end if
40:  end if
41: end for
42: return  $\frac{\sum_{elem \in S_t} R_{elem}}{|S_t|}$ 
```

proportionally decreased (the minimal possible value being 0).

Note that the complexity of the sorting itself mostly contributes to the overall complexity of the relevance-based sorting of suggestions. As can be found out from Algorithm 4, the complexity is in $O(cmn l^2 + m \log m)$ (c – maximal number of terms occurring in a suggestion, thus a constant; m – number of suggestions; n – number of words in the preference sets; l – maximal length of a word in suggestion terms, basically a constant), which gives $O(m(n + \log m))$. As the size of the sets of user preferences can be practically treated as constant, we obtain the $O(m \log m)$ complexity class with respect to the number of

suggestions, which is feasible.

5.6 Natural Language Generation (NLG) Component

The DINO framework is supposed to be used primarily by users who are not experts in ontology engineering. Therefore the suggestions are produced in a form of very simple natural language statements, as seen in the previous section. Moreover, we automatically create a natural language representation of the whole addition model, interfacing the framework described in [42]. This is meant to further support laymen users by readable representation of the whole addition model in order to give them an overall impression of the changes.

The single suggestions are still bound to the underlying statement in the addition ontology model. Therefore a user can very easily add the appropriate OWL axioms into the new version of the O_M master ontology without actually dealing with the intricate OWL syntax itself. Concrete examples of both suggestions and continuous natural language representation of the addition model are given in Section 6.

6 Example Application and Results of DINO Integration

We applied the integration technique described in Section 5 in the context of data typical for biomedical research. However, the way of exploiting the DINO integration technique reported in this section is rather general, since it aims at cost-efficient extension or population of a master ontology by knowledge learned from empirical data. Thus, a similar deployment of the integration can actually help to tackle needs of many other possible use cases.

Real world data for the master ontology and ontology learning sources were used. More specifically, we employed resources from CO-ODE biomedicine ontology fragment repository⁵ and data from relevant Wikipedia topics, respectively.

Rigorous evaluation of the whole process of integration is a complex task involving lot of open problems as its sub-problems (for instance, there is no standard ontology evaluation process applicable in general – see [27, 13]). Moreover, there is an emphasis on the human-readable and laymen oriented form of the integration process results. This dimension forms a primary axis of the evaluation, however, its realisation involves logistically demanding participation of a broader (biomedicine) expert community.

⁵ See <http://www.co-ode.org/ontologies>.

Accomplishing the above tasks properly is a part of our future work. Nonetheless, there are several aspects that can be assessed and reported even without devising an optimal ontology evaluation method (which may be impossible anyway) and/or getting involved large representative sample of domain experts:

- features of the learned ontology (e.g. size or complexity)
- mappings established by alignment
- basic assessment of the quality and correctness of suggestions and their sorting according to defined preferences

These factors of integration are analysed and discussed within an experimental application described in Section 6.1.

The negotiation component has recently been evaluated separately as a stand-alone module, using the Ontology Alignment Evaluation Initiative test suite⁶ and experiments on the impact that the argumentation approach has over a set of mappings. A comparison wrt. current alignment tools is presented in [29]. The preliminary results of these experiments are promising and suggest that the argumentation approach can be beneficial and an effective solution to the problem of dynamically aligning heterogeneous ontologies. This justifies also the application of the implemented technique in the ontology integration task.

6.1 Experimental Integration of Biomedical Research Knowledge – Extension of (Blood) Cells Ontology Fragment

In order to show the basic features of our novel integration technique in practice, we tested the implementation using knowledge resources from biomedicine domain⁷. In particular, we combined fragments of GO cellular component description and eukaryotic cell description⁸ to form the master ontology. In the example scenario, we wanted to extend this master ontology using content of Wikipedia entries on `Cells_(biology)` and `Red_blood_cell`. These resources were passed to the ontology learning DINO component and respective ontology was learned. Both master and learned ontology samples are displayed in Figure 3 (on the left-hand and right-hand side, respectively). Note

⁶ See <http://oaei.ontologymatching.org/>.

⁷ Should the reader be interested, all relevant resources used and/or created during the described experiment are available at http://smile.deri.ie/resources/2007/08/31/dino_exp_data.zip

⁸ Samples downloaded from the CO-ODE repository, see http://www.co-ode.org/ontologies/bio-tutorial/sources/GO_CELLULAR_COMPONENT_EXTRACT.owl and <http://www.co-ode.org/ontologies/eukariotic/2005/06/01/eukariotic.owl>, respectively.

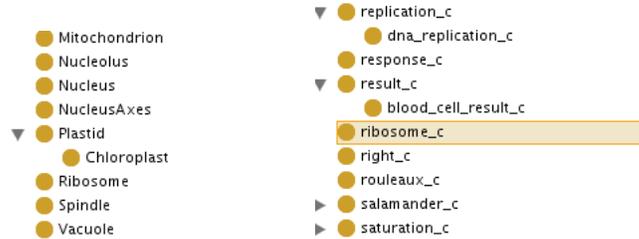


Fig. 3. Sample from master and learned ontology

that these master and learned ontologies correspond to the O_M, O_L ontologies displayed in Figure 2, Section 5. The names in learned ontology have specific suffixes (i.e. “_c”). This is due to naming conventions of the ontology learning algorithm we use. We keep the suffixes in suggestions, since they help to easily discriminate what comes from empirical data and what from the master ontology. However, we filter them out when generating the text representing the whole extension model (see below for examples).

Table 2 compares metric properties of the master and learned ontologies, as computed by the Protégé tool. The particular metrics are expanded as follows:

Table 2

Metrics of master and learned ontologies

Metric/ Ontology	M_1	M_2	M_3	M_4	M_5	M_6
Learned	391 / 379 / 12	3 / 1 / 5	7 / 1 / 16	0	13 / 13 / 0	$\mathcal{ALC}(D)$
Master	40 / 36 / 4	2 / 1 / 2	5 / 1 / 15	16 (restr.)	1 / 1 / 0	\mathcal{ALCN}

M_1 – number of named classes (all/primitive/defined); M_2 – number of parents per class (mean/median/maximum); M_3 – number of siblings per class (mean/median/maximum); M_4 – number of anonymous classes (restrictions); M_5 – number of properties (all/object/datatype); M_6 – Description Logics expressivity.

The learned ontology has higher ratio of primitive classes, moreover, it contains no restriction on class definitions. There are some simple object properties with both domains and ranges defined. Its DL expressivity allows concept intersection, full universal and existential quantification, atomic and complex negation and datatypes. The expressivity of the master ontology does not involve datatypes, however, it contains numeric restrictions. Summing up, the master ontology contains several complicated constructs not present in the learned ontology, however, the ontology learned only from two simple and relatively small resources is much larger.

When computing the negotiated alignment (the O_A ontology as given in Figure 2, Section 5) between master and learned ontology, 207 mappings were produced and among them, 16 were accepted. A sample from the alignment ontology is displayed in Figure 4.

```

<owl:Class rdf:about="#Chromosome">
  <owl:equivalentClass rdf:resource="#chromosome_c"/>
</owl:Class>

<owl:Class rdf:about="#Chloroplast">
  <owl:equivalentClass rdf:resource="#chloroplast_c"/>
</owl:Class>

<owl:Class rdf:about="#Ribosome">
  <owl:equivalentClass rdf:resource="#ribosome_c"/>
</owl:Class>

<owl:Class rdf:about="#Ribosome">
  <owl:equivalentClass rdf:resource="#the_ribosome_c"/>
</owl:Class>

<owl:Class rdf:about="#Nucleus">
  <owl:equivalentClass rdf:resource="#nucleus_c"/>
</owl:Class>

<owl:Class rdf:about="#Mitochondrion">
  <owl:equivalentClass rdf:resource="#mitochondrium_c"/>
</owl:Class>

```

Fig. 4. Sample alignment

Merging of the learned and master ontologies according to the computed alignments results in several inconsistencies – the report generated by DINO is displayed in Figure 5. Two of these three inconsistencies are resolved cor-

```

Inconsistency:
The following classes are disjoint and in mutual sub-class relationship at the same time:
"organelle_c" and "nucleus_c"

Inconsistency:
The following classes are disjoint and in mutual sub-class relationship at the same time:
"cell_c" and "blood_cell_c"

Inconsistency:
The following classes are disjoint and in mutual sub-class relationship at the same time:
"cell_wall_c" and "membrane_c"

```

Fig. 5. Report on inconsistencies

rectly (according to human intuition) by the algorithm, forming an integrated ontology O_I , as displayed in Figure 2, Section 5.

After resolving the inconsistencies (3 inconsistencies per an integrated resource were resolved in average within our experiment) and generating the addition model, natural language suggestions (associated with respective OWL axioms) are produced. Sample suggestions associated with respective relevance measures are displayed in Figure 6. A portion of the continuous text generated by the NLG component that is corresponding to the addition model is displayed in Figure 7. Similar “pretty” texts are to be presented to users in the extended DINO interface (the current interface offers only raw text, however, necessary parsing, filtering and highlighting of the ontology terms is under construction). It provides users with additional source of lookup when deciding which suggestions to accept into the next version of the master ontology.

The suggestions are the ultimate output of the integration algorithm. Their

```

...
-----
Relevance: 0.75
Suggestion : The class "cell_nucleus_c" is disjoint with the class "compartment_c".
-----
Relevance: 0.083333336
Suggestion : The class "Nucleus" is equivalent to the class "nucleus_c"
-----
Relevance: 0.0
Suggestion : The class "organelle_c" has the "mitochondrion_c" subclass.
-----
Relevance: 0.0
Suggestion : The class "Mitochondrion" is equivalent to the class "mitochondrion_c".
-----
Relevance: -0.8333333
Suggestion : The class "chromosome_c" has the "Organelle" superclass.
-----
Relevance: -0.9166666
Suggestion : The class "Chromosome" is equivalent to the class "chromosome_c".
-----
...

```

Fig. 6. Sample suggestions

```

...
There are "Cells", "Nucleuss", "bacteriums", and "genetic diseases".
There are "red blood cells", "absorptions", "additional functions", "advantages", and "archaeons".
There are "autoimmunediseseas", "aplasiums", "appendages", "areas", and "atoms".
There are "bacterias", "bacteriums", "beacons", "bilayers", and "blockages".
There are "cannots", "capacitys", "capsules", "cells", and "changes".
There are "chloroplasts", "chromosomals", "ciliums", "coagulations", and "comparisons".
...

```

Fig. 7. Sample from the generated continuous text

main purpose is to facilitate laymen effort in incorporation of new knowledge from unstructured resources into an ontology. Therefore we performed basic evaluation of several parameters that influence actual applicability of the suggestions. We ran the integration algorithm on the same data with four different suggestion-preference sets, simulating four generic trends in the preference definition:

- specification of rather small number of preferred terms, no unwanted terms
- specification of rather small number of preferred and unwanted terms
- specification of larger number of preferred terms, no unwanted terms
- specification of larger number of preferred and unwanted terms

Table 3 gives an overview of the four iterations, the particular preferred and unwanted terms and distribution of suggestions into relevance classes. The terms were set by a human user arbitrarily, reflecting general interest in clinical aspects of the experimental domain knowledge. The terms in preference sets reflect possible topics to be covered by the automatic extension of the current ontology. S_+ , S_0 and S_- are classes of suggestions with relevance greater, equal and lower than zero, respectively ($S = S_+ \cup S_0 \cup S_-$).

For each of the relevance classes induced by one iteration, we randomly selected 20 suggestions and computed two values on this sample:

Table 3

Iterations – the preference sets and sizes of the resulting suggestion classes

Iteration	Preferred	Unwanted	$ S_+ $	$ S_0 $	$ S_- $	$ S $
I_1	cell; autoimmune disease; transport; drug; gene; DNA	\emptyset	310	429	0	739
I_2	cell; autoimmune disease; transport; drug; gene; DNA	bacteria; prokaryotic; organelle; wall; chromosome; creation	250	344	145	739
I_3	cell; autoimmune disease; transport; drug; gene; DNA eukaryotic; organ; function; part; protein; disease; treatment; cell part immunosuppression; production	\emptyset	485	254	0	739
I_4	cell; autoimmune disease; transport; drug; gene; DNA eukaryotic; organ; function; part; protein; disease; treatment; cell part immunosuppression; production	bilayer; bacteria; prokaryotic; additional function; organelle; macromolecule; archaeon; vessel; wall; volume; body; cell nucleus; chromosome; erythrocyte; creation	314	292	133	739

- $P_x, x \in \{+, 0, -\}$ – ratio of suggestions correctly placed by the sorting algorithm into an order defined by a human user for the same set (according to the interest defined by the particular preferences)
- $A_x, x \in \{+, 0, -\}$ – ratio of suggestions that are considered appropriate by a human user according to his or her knowledge of the domain (among all the suggestions in the sample)

The results are summed up in Table 4. More details on interpretation of all the experimental findings are given in consequent Section 6.2.

Table 4

Evaluation of random suggestion samples per class

Iteration	P_+	A_+	P_0	A_0	P_-	A_-
I_1	0.45	0.75	0.90	0.60	-	-
I_2	0.45	0.75	1.00	0.80	0.60	0.70
I_3	0.70	0.80	0.95	0.75	-	-
I_4	0.55	0.75	0.70	0.85	0.50	0.85

6.2 Discussion of the Experiment Results

The DINO integration library allows users to submit the resources containing knowledge they would like to reflect in their current ontology. The only thing that is needed is to specify preferences on the knowledge to be included using the sets of preferred and unwanted terms. After this, sorted suggestions on possible ontology extensions (after resolution or reporting of possible inconsistencies) can be produced and processed in minutes, whereas the purely manual development and integration of respective ontology would take hours even for relatively simple natural language resources. Moreover, it would require a certain experience with knowledge engineering, which is uncommon among biomedicine domain experts.

In Section 6.1 we described the application of our integration technique to an extension of biomedical research ontology fragment. The analysed results show that the suggestions produced are mostly correct (even though rather simple and sometimes obvious) with respect to the domain in question, ranging from 50% to 85% among the algorithm iterations. The relevance-based sorting according to preferences is more appropriate in case of irrelevant (zero relevance) suggestions, ranging from 70% to 100% of correctly placed suggestions. Its precision in case of suggestions with positive and negative relevance is lower, ranging from 45% to 70%. More terms in the preference sets cause better sorting performance (the ratio of appropriate suggestions being independent on this fact). Thus, the best discrimination in terms of presenting the most relevant suggestions first is achieved for larger preference sets. However, even the discrimination for smaller sets is fair enough (as seen in Table 3 in the previous section).

The automatically produced natural language suggestions can be very easily browsed and assessed by users who are not familiar with ontology engineering at all. Since the respective axioms are associated to the suggestions, their inclusion into another version of the master ontology is pretty straightforward once a suggestion is followed by a user. The DINO integration technique still needs to be evaluated with a broader domain expert audience involved, however, even the preliminary results presented here are very promising in the scope of the requirements specified in Section 1.

7 Notes on Realistic DINO Deployment

The EU IST 6th Framework project RIDE has identified and analysed several biomedical use case areas in [15] relevant concerning deployment of the Semantic Web technologies (i.e., ontologies and related querying, knowledge

and data management tools). The scope of [15] is rather broad, however, we can track few specific areas with significant needs that can be covered by the DINO ontology lifecycle and integration framework (Section 7.1). Section 7.2 discusses preliminary feedback of our potential users and consequently suggests most appropriate modes of the DINO prototype exploitation.

7.1 Selected Use Case Areas

Longitudinal Electronic Health Record: The main topic here is development of standards and platforms supporting creation and management of long-term electronic health records of particular patients. These records should be able to integrate various sources of data coming from different medical institutions a patient may have been treated in during his whole life. Quite obviously, one needs to integrate different data sources, present very often in unstructured natural language form. Ontologies extracted from the respective patient data resources can very naturally support their integration into longitudinal electronic health records by means of DINO.

Epidemiological Registries: Epidemiology analyses diseases, their reasons, statistical origins and their relation to a selected population sample's socio-economic characteristic. Epidemiological registries should be able to reasonably store and manage data related to population samples and their medical attributes in order to support efficient processing of the respective knowledge by the experts. In this use case area, one has to integrate knowledge from electronic health records in order to create population-wise repositories. Once the ontology-enabled electronic health records are created (DINO can help here as mentioned above), one can integrate them within another version of an "epidemiology" ontology (again, by means of the DINO framework). The resulting model can be employed in order to perform symbolic analysis (using ontology-based symbolic querying and logical inference) of the registry data, complementing the statistical numeric analysis methods.

Public Health Surveillance: Public health surveillance presents ongoing collection, analysis, interpretation and dissemination of health-related data in order to facilitate a public health action reducing mortality and/or improving health. The use case area puts an emphasis on efficient dynamic processing of new data that are mostly in the free natural language text form, which can be directly facilitated by the DINO integration of respective learned ontologies. Ontologies created from and extended by urgent dynamic data can efficiently support expert decisions in risk management tasks. Continuous integration of less urgent data from various sources (either texts or ontologies) can support studies on public health issues in the long term perspective then.

Management of Clinical Trials: Clinical trials are studies of the effects of newly developed drugs on real patient samples. They are essential part of approval of new drugs for normal clinical use and present an important bridge between medical research and practice. Efficient electronic data representation and querying is crucial here. However, even if the data are electronically represented, problems with their heterogeneity and integration occur as there are typically several different institutions involved in a single trial. The presented integration method can help in coping with the data heterogeneity here, especially when some of the data is present in the natural language form.

7.2 Preliminary User Feedback and Lessons Learned

We presented a DINO demo and/or sample knowledge integration results to biomedical domain and ontology engineering experts⁹. We also discussed a sketch of the DINO application in the above practical use cases with them.

Their preliminary feedback can be summarised into the following three points: (1), the framework was considered as a helpful complement to the traditional manual ontology development environments (such as Protégé); (2), the results were found promising concerning the scalable ontology extension by the knowledge in unstructured domain resources, however, certain refinement by ontology engineers was generally considered as a must in order to maintain high quality of the respective master biomedical ontologies; (3), the natural language presentation of the sorted extension suggestions was found to be very useful for the domain experts with no ontology engineering background. The last finding has been further supported by the recent evaluation of the natural language generation framework we use in DINO (see [11] for details).

Following the discussion with the domain and ontology engineering experts, we can distinguish between two practical and reliable DINO application modes with different requirements on the expert user involvement:

- **Instance-only integration:** Ontology learned from the textual resources is semi-automatically integrated into a master ontology, taking only instance-related assertions into account, i.e., the upper ontology is populated with new instances of the present concepts and with relations among the instances. Such an application does not require any extensive expert involvement of ontology engineers, since the instance-related suggestions produced by DINO are relatively reasonable according to our discussions with domain

⁹ These were namely researchers from the REMEDI institute, see <http://www.nuigalway.ie/remedi/>, Prof. Werner Ceusters, M.D. (director of the Ontology Research Group of the New York State Center of Excellence in Bioinformatics and Life Sciences) and ontology engineers from the Knowledge Engineering Group at the University of Economics in Prague.

experts. Severe modelling errors can only be introduced very rarely, therefore only the expert knowledge of the domain is generally enough to decide which DINO suggestions to follow in the master ontology extension.

- **Full-fledged integration:** An unrestricted processing of the DINO suggestions, i.e., taking also the class-related assertions into account, requires more careful expert involvement in order to guarantee high quality of the integration results. Ontology experts are generally still needed when resolving possible modelling bugs (such as multiple class inheritance or redundant disjointness relations) that might be insufficiently tackled by the domain experts when processing the natural language DINO suggestions. State of the art methodologies such as ontology “re-engineering” as introduced in [3] can help when applying DINO this way.

8 Summary and Future Work

We have presented the basic principles of DINO – a novel lifecycle scenario and framework for ontology integration and maintenance in dynamic and data-intensive domains like medicine. As a core contribution of the paper, we have described the mechanism of integration of automatically learned and manually maintained medical knowledge. The presented method covers all the requirements specified in Section 1. The proposed combination of automatic and manual knowledge acquisition principles, integration and inconsistency resolution ensures more scalable production and extension of ontologies in dynamic domains. We presented and analysed results of a preliminary practical application of the DINO integration technique in Section 6. Section 7 outlined possible applications in realistic use case areas that have been recently identified in the biomedicine and e-health fields. The section also summarised preliminary feedback of our potential users. Based on the feedback analysis, two practical DINO application modes were suggested. Note that we have also delivered prototype implementations of a DINO API library and a respective GUI interface (research prototypes of the respective software can be downloaded at <http://smile.deri.ie/tools/dino>).

Since the primary funding project has finished, we are in the process (as of 2008) of securing another funding that could support further improvements of DINO. These improvements consist mainly of an extended support for inconsistency resolution, integration with state of the art ontology editors (primarily Protégé) and extension of the DINO user interface (e.g., providing explicit support for the two application modes given in Section 7.2).

Moreover, we have recently started to work on another project with motivations similar to DINO, however, with much more ambitious goals. [34] presents a preliminary proposal and results of a novel empirical knowledge

representation and reasoning framework. One of the principal applications of the researched framework is a complex empirical inference-based integration of arbitrary emergent knowledge (e.g., learned ontologies) with precise manually designed knowledge bases. We plan to combine the ontology integration powered by the reasoning described in [34] with the results achieved within the DINO implementation in order to allow for more efficient, scalable, user-friendly and robust dynamic maintenance of (partially emergent) ontologies. Last but not least, we are going to continuously evaluate the resulting framework among broader biomedicine expert communities and improve it in line with demands of interested industry partners (possibly, but not only within the presented real-world application domains).

Acknowledgements The article is a significantly extended version of the paper: Vít Nováček, Loredana Laera, Siegfried Handschuh. *Dynamic Integration of Medical Ontologies in Large Scale*. In: Proceedings of the WWW2007 / HCLSDI workshop. ACM Press, 2007. (see http://www2007.org/workshops/paper_141.pdf). The presented work has been kindly supported by the EU IST 6th framework's Network of Excellence 'Knowledge Web' (FP6-507482), by the 'Líon' project funded by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and partially by Academy of Sciences of the Czech Republic, 'Information Society' national research program, the grant number AV 1ET100300419. Moreover, we greatly appreciated the anonymous reviewers' remarks that resulted in significant improvements of the submitted text.

References

- [1] A. Alasoud, V. Haarslev, and N. Shiri. A hybrid approach for ontology integration. In *Proceedings of the 31st VLDB Conference*. Very Large Data Base Endowment, 2005.
- [2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, Cambridge, USA, 2003.
- [3] S. Bechhofer, A. Gangemi, N. Guarino, F. van Harmelen, I. Horrocks, M. Klein, C. Masolo, D. Oberle, S. Staab, H. Stuckenschmidt, and R. Volz. Tackling the ontology acquisition bottleneck: An experiment in ontology re-engineering, 2003. Retrieved at <http://citeseer.ist.psu.edu/bechhofer03tackling.html>, Apr 3 2008.
- [4] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*, 2004. Available at (February 2006): <http://www.w3.org/TR/owl-ref/>.

- [5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 5, 2001.
- [6] D. Brickley and R. V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*, 2004. Available at (February 2006): <http://www.w3.org/TR/rdf-schema/>.
- [7] F. C. C. Brewster and Y. Wilks. User-centred onology learning for knowledge management. In *In Proceedings 7th International Workshop on Applications of Natural Language to Information Systems, Stockholm.*, 2002.
- [8] D. Calvanese, G. D. Giacomo, and M. Lenzerini. A framework for ontology integration. In *In Proc. of the First Semantic Web Working Symposium*. Springer-Verlag, 2001.
- [9] P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the NLDB 2005 Conference*, pages 227–238. Springer-Verlag, 2005.
- [10] O. Corcho, A. Lopez-Cima, and A. Gomez-Perez. The ODESeW 2.0 semantic web application framework. In *Proceedings of WWW 2006*, pages 1049–1050, New York, 2006. ACM Press.
- [11] B. Davis, A. A. Iqbal, A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, and S. Handschuh. Roundtrip ontology authoring. In *Proceedings of ISWC 2008*. Springer-Verlag, 2008. Submitted.
- [12] S. M. Deen and K. Ponnampereuma. Dynamic ontology integration in a multi-agent environment. In *Proceedings of AINA '06*. IEEE Computer Society, 2006.
- [13] K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proceedings of the International Semantic Web Conference. Athens, GA, USA.*, 2006.
- [14] R. Dieng-Kuntz, D. Minier, M. Ruzicka, F. Corby, O. Corby, and L. Alalmarguy. Building and using a medical ontology for knowledge management and cooperative work in a health care network. *Computers in Biology and Medicine*, 36:871–892, 2006.
- [15] M. Eichelberg. Requirements analysis for the ride roadmap. Deliverable D2.1.1, RIDE, 2006.
- [16] J. Euzenat. An API for ontology alignment. In *ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 698–712. Springer-Verlag, 2004.
- [17] J. Euzenat, T. L. Bach, J. Barrasa, P. Bouquet, , J. D. Bo, R. Dieng, M. Ehrig, , M. Hauswirth, M. Jarrar, , R. Lara, , D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. V. Acker, and I. Zaihrayeu. D2.2.3: State of the art on ontology alignment. Technical report, Knowledge Web, 2004.
- [18] J. Euzenat, D. Loup, M. Touzani, and P. Valtchev. Ontology alignment with ola. In *Proceedings of the 3rd International Workshop on Evaluation of Ontology based Tools (EON)*, Hiroshima, Japan, 2004. CEUR-WS.
- [19] M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of*

- the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.
- [20] M. Fernandez-Lopez, A. Gomez-Perez, and M. D. Rojas. Ontologies' crossed life cycles. In *Proceedings of International Conference in Knowledge Engineering and Management*, pages 65–79. Springer-Verlag, 2000.
 - [21] G. Flouris, Z. Huang, J. Z. Pan, D. Plexousakis, and H. Wache. Inconsistencies, negations and changes in ontologies. In *Proceedings of AAAI 2006*. AAAI Press, 2006.
 - [22] J. H. Gennari, M. A. Musen, R. W. Ferguson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003.
 - [23] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer-Verlag, 2004.
 - [24] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
 - [25] P. Haase and Y. Sure. State-of-the-art on ontology evolution. Deliverable 3.1.1.b, SEKT, 2004.
 - [26] P. Haase and J. Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *Proceedings of the URSW2005 Workshop*, pages 45–55, NOV 2005.
 - [27] J. Hartmann, P. Spyns, A. Giboin, D. Maynard, R. Cuel, M. C. Suarez-Figueroa, and Y. Sure. Methods for ontology evaluation (D1.2.3). Deliverable 123, Knowledge Web, 2005.
 - [28] J. Heflin and J. Hendler. Dynamic ontologies on the web. In *Proceedings of AAAI 2000*. AAAI Press, 2000.
 - [29] L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat, and T. Bench-Capon. Argumentation over ontology correspondences in mas. In *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), Honolulu, Hawaii, USA. To Appear*, 2007.
 - [30] L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. R. Payne. Reaching agreement over ontology alignments. In *Proceedings of 5th International Semantic Web Conference (ISWC 2006)*. Springer-Verlag, 2006.
 - [31] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics Control Theory*, 10:707–710, 1966.
 - [32] A. Maedche and S. Staab. Learning ontologies for the semantic web. In *Semantic Web Workshop 2001*, 2001.
 - [33] A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, chapter 9, pages 173–190. Springer-Verlag, 2004.

- [34] V. Nováček. Complex inference for emergent knowledge. Technical Report DERI-TR-2008-04-18, DERI, NUIG, 2008. Available at <http://smile.deri.ie/resources/2008/vit/pubs/aerTR0408.pdf>.
- [35] V. Nováček, S. Handschuh, L. Laera, D. Maynard, M. Völkel, T. Groza, V. Tamma, and S. R. Kruk. Report and prototype of dynamics in the ontology lifecycle (D2.3.8v1). Deliverable 238v1, Knowledge Web, 2006.
- [36] N. Noy and M. Musen. The prompt suite: Interactive tools for ontology merging and mapping, 2002.
- [37] N. F. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, pages 428–440, 2004.
- [38] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proceedings of K-CAP'01*, 2001.
- [39] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer-Verlag, 2004.
- [40] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, 2004.
- [41] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*, Sardinia, 2002. Springer.
- [42] V. Tablan, T. Polajnar, H. Cunningham, and K. Bontcheva. User-friendly ontology authoring using a controlled language. In *Proceedings of LREC 2006 - 5th International Conference on Language Resources and Evaluation*. ELRA/ELDA Paris, 2006.
- [43] B. L. S. V. Tamma, I. Blacoe and M. Wooldridge. Introducing autonomous behaviour in semantic web agents. In *In Proceedings of the Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland, November.*, 2005.
- [44] M. Völkel and T. Groza. SemVersion: RDF-based ontology versioning system. In *Proceedings of the IADIS International Conference WWW/Internet 2006 (ICWI 2006)*, 2006.