Supporting Digital Healthcare Services Using Semantic Web Technologies

Gintaras Barisevičius, Martin Coste, David Geleta, Damir Juric, Mohammad Khodadadi, Giorgos Stoilos(✉), and Ilya Zaihrayeu

Babylon Health, London SW3 3DD, UK
{gintaras.barisevicius,martin.coste,david.geleta,damir.juric, mohammad.khodadadi,giorgos.stoilos,ilya.zaihrayeu}@babylonhealth.com

Abstract. We report on our efforts and faced challenges in using Semantic Web technologies for the purposes of supporting healthcare services provided by Babylon Health. First, we created a large medical Linked Data Graph (LDG) which integrates many publicly available (bio)medical data sources as well as several country specific ones for which we had to build custom RDF-converters. Even for data sources already distributed in RDF format a conversion process had to be applied in order to unify their schemata, simplify their structure and adapt them to the Babylon data model. Another important issue in maintaining and managing the LDG was versioning and updating with new releases of data sources. After creating the LDG, various services were built on top in order to provide an abstraction layer for non-expert end-users like doctors and software engineers which need to interact with it. Finally, we report on one of the key use cases built in Babylon, namely an AI-based chatbot which can be used by users to determine if they are in need of immediate medical treatment or they can follow a conservative treatment at home. To match user text to our internal AI-models an NLP-based knowledge extraction and logic-based reasoning approach was implemented and evaluation provided with encouraging results.

1 Introduction

The use of Semantic Web technologies such as Linked Data have started to be used extensively in many real-world applications [5,8,15]. Especially in the biomedical domain, OWL has been adopted since the early days of the Semantic Web and used to create a large number of medical ontologies [19], prominent examples of which are SNOMED [22], FMA [11], NCI [10], the Disease ontology [20], and many more. Many of these ontologies cover different and complementary topics such as genes, human phenotypes, proteins, and so on, and can be quite heterogeneous making it hard to retrieve information in a uniform way. Linking them under a homogeneous data model over which applications can be built would be beneficial [5,18].

Semantic technologies have also been adopted within Babylon Health.[1] Babylon offers healthcare services through a mobile application. Users can register to

[1] https://www.babylonhealth.com/.

D. Vrandečić et al. (Eds.): ISWC 2018, LNCS 11137, pp. 291–306, 2018.
https://doi.org/10.1007/978-3-030-00668-6_18

the app and have video consultations with doctors and healthcare professionals. The service also allows doctors to prescribe drugs to patients which can receive them from pharmacies of their choice. Moreover, patients can also receive referrals to health specialists or book lab exams with nearby facilities. Besides consultation with doctors, Babylon has been developing an AI-based doctor accessible through a chatbot, which can be used for symptom checking and triaging—that is, determining if the conditions that a user is entering in the chatbot are critical and he/she is in need of immediate medical attention or he/she can follow a conservative treatment at home.

Various services within Babylon generate, exchange, and consume clinical and health data and knowledge. For example, information extraction and text annotation services have been developed in order to process patient entered text and recognise the relevant medical terms that are entered. These terms may need to be compared with symptoms and risk factors in our symptom checking and triaging engines or with past diagnosis stored in the user profiles. Various other services in Babylon like drug prescribing or billing also deal with medical data like drugs, their side effects, contraindications and more.

To support the above services a medical Linked Data Graph (LDG) has been created by converting various medical data sources into RDF. Hence, all data within Babylon (diagnosis, drugs, etc.) are encoded using codes from medical ontologies like SNOMED, NCI, ICD-10, and more. Standards are heavily used in order to represent complex medical conditions and reasoning services have been implemented in order to achieve high degree of interoperability and intercommunication between the services. Some of the challenges faced in realising this use case are the following:

- Biomedical data sources are highly heterogeneous and custom converters had to be implemented in order to unify and harmonise them.
- Although efforts like BioPortal [19] and Bio2RDF [2] already offer a very large number of medical ontologies in RDF, several country specific clinical data sources are missing.
- OWL often exhibits complex structure (e.g., complex and/or nested class expressions) which would be impossible to be interpreted by our non-expert end-users (mostly doctors and software engineers). Consequently, even data sources distributed in OWL had to be converted to our simplified model.
- Updating our LDG with new releases of the data sources is a non-trivial issue since services already operate over the existing schema and changes may alter their behaviour.
- It would not be possible for our end-users to interact with triple-stores and use SPARQL, hence abstraction layers, services, and browsers had to be implemented in order help them use and feel comfortable with Semantic Web technologies and the LDG.
- Comparing OWL classes with existing reasoners is too strict in a real-world setting where one has to deal with language ambiguity and variability.

In the following, we first present our efforts in creating a medical Linked Data Graph and show how we addressed the above challenges. Next, we present several of the services we built around it in order to make the content accessible and easy to use by our non-experts. In more detail, we built a middle-ware service, called ClinicalKnowledge, whose purpose is to provide an abstraction layer to the LDG through a set of REST services. In addition, we have also implemented a web-browsing tool which can be used to search for classes and see their content like relations to other classes, direct super-classes, and more. Although, our LDG does not store complex OWL class expressions, such expressions are used in other components and services within Babylon like the triaging engine or patient profiles where complex medical conditions are formed by combining IRIs from the LDG and using OWL constructors. These expressions need to be compared with each other in order for services to exchange knowledge and interoperate and for these purposes a custom (hybrid) reasoner was implemented. Finally, we report on the triaging use-case built in Babylon and the role of our hybrid reasoner in matching user text to the internal triaging and symptom checking models. Preliminary evaluation of our Semantic Web-based (NLP plus logic) solution provided with encouraging results.

2 Building a Medical Linked Data Graph

The overall architecture of our platform is depicted in Fig. 1. The pipeline currently supports structured (RDF/OWL) and semi-structured (XML, CSV/TSV) data sources. All sources (even those already distributed in RDF) undergo a conversion process in which their schema and structure is processed in order to adapt it to the RDF model used in Babylon and reconcile their differences as much as possible. This conversion process also links the sources to an *Upper Level Ontology* which consists of an abstract medical model via which access is realised. Since data sources often feature overlaps, ontology alignment algorithms [21] are also used in order to establish mappings between the various sources and improve the level of integration. All converted data sources as well as the computed mappings are loaded into GraphDB.[2] The pipeline also supports the integration of information extracted from unstructured (web) data sources via Machine Reading [9] and crawling but the description of this pipeline is out of the scope of this paper. On top of the LDG a set of services is provided for outside clients to interact with the LDG. As it can be seen in the architecture, the LDG is continuously updated with new data sources as these are released. In the following sections we present further technical details about the aforementioned components.

2.1 Data Sources

Today a wealth of medical knowledge and data sources are available on-line. Several of these are already distributed in OWL and/or RDF, prominent examples of which are SNOMED, NCI, FMA, the ontologies in BioPortal, and many more. The UMLS project also consists of a continuous effort towards integrating

[2] https://ontotext.com/products/graphdb/.

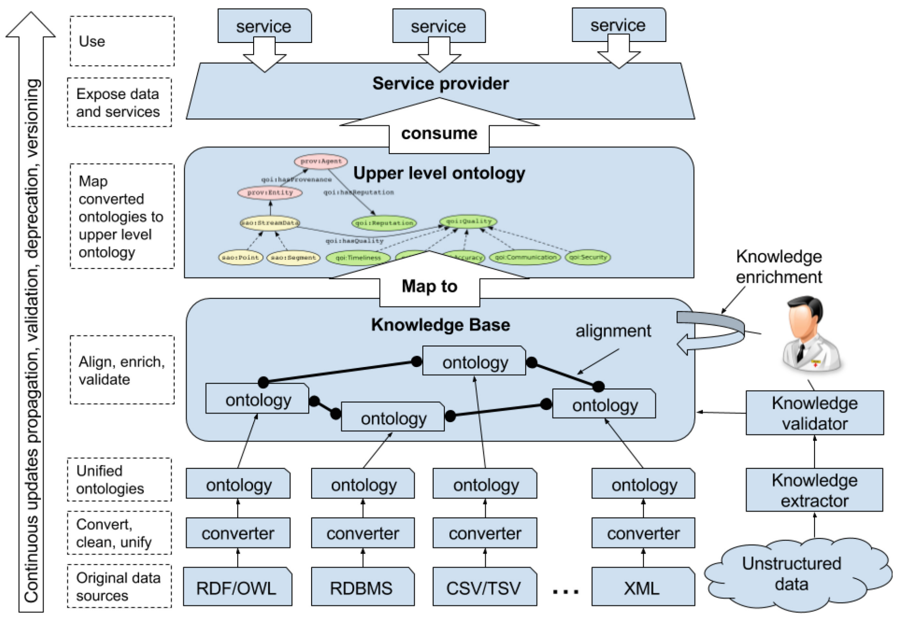


Fig. 1. Architecture of Babylon's linked data graph generation pipeline

and linking biomedical ontologies under a common vocabulary and providing homogeneous access through the UMLS semantic network [17]. The Babylon LDG uses several ontologies from UMLS, however, two issues were identified: (i) UMLS does not contain the most up to date releases of data sources, e.g., new releases of SNOMED are integrated with a six months delay, and (ii) it is missing some country specific data sources like dm+d (which is actually updated weekly), the Canadian Clinical Drug Dataset, and more.

Other prominent efforts in integrating biomedical data sources under a common RDF-based model is the Bio2RDF effort [2,4]. Bio2RDF could be a potential fit for the Babylon use case, however, it mostly focuses on Genes, Proteins, Genomics, and so forth, while Babylon mostly focuses on clinical services like Diseases, Risk Factors, Symptoms, and Drugs, so data sources like SNOMED and NCI which are missing from Bio2RDF are critical. In addition, most importantly the schema and structure of the generated Linked Data Graph had to be under the control of Babylon in order to be easily customisable and adaptable to internal service requirements. For these reasons several custom converters had to be implemented and an LDG was constructed from scratch.

The LDG was built by integrating almost 300 structured as well as semi-structured data sources. We have used most of the UMLS, several ontologies from BioPortal, latest versions of well-known ontologies like FMA, SNOMED, NCI, coding systems like ICD-10(pcs) and ReadCodes, as well as several country specific sources and extensions like dm+d, RxNorm, and more. Table 1 presents

Table 1. List of Important Data sources in our LDG.

Ontology	s/o IRIs	properties	$\sharp\{iri_1\ p\ iri_2\ .\}$	$\sharp\{iri_1\ p\ \mathsf{Lt}\ .\}$	$\sqsubseteq$
General Medical Vocabularies					
SNOMED	326 k	78	926 k	1,1 m	486 k
NCI	133 k	204	298 k	1,8 m	147 k
CHV	57 k	8	0	247 k	0
MeSH	2 m	44	3,6 m	9,6 m	187 k
MedDRA	26 k	8	0	885 k	34 k
Drug Ontologies					
dm+d	309 k	33	444 k	1,6 m	0
SIDER	1 m	28	2,6 m	8,4 m	0
Drugbank	10 m	36	74 m	195 k	0
RxNorm	114 k	41	989 k	1,12 m	200 k
DailyMed	10 k	31	57 k	80 k	0
Coding Systems					
OPCS-4	11 k	6	0	22 k	9 k
ICD-10	11 k	12	11 k	35 k	11 k
ICD-10pcs	190 k	9	0 k	708 k	190 k
CTV3	322 k	97	679 k	868 k	278 k
Read2	89 k	9	0	355 k	89 k

statistics about some of the sources integrated in our LDG; it shows the number of classes and individuals (i.e., IRIs in the subject or object position of triples), the number of properties, the number of triples of the form $s\ p\ o$. where both s and o are IRIs, and the number of such triples where o is an `owl:Literal` and the number of subClassOf axioms ($\sqsubseteq$). For readability we have rounded up numbers and used "k" to indicate thousands and "m" to indicate millions. In total the Babylon medical LDG consists of about 280 million triples which were loaded in GraphDB.

2.2 Schema and Data Model

Since the LDG is stored in a triple-store all data sources are serialised to triples. This creates problems when the original source contains complex class expressions that require the use of blank nodes in order to be serialised. For example, the OWL axiom $\mathsf{Malaria} \sqsubseteq \exists \mathsf{mayHaveFinding}.\mathsf{Fever}$ in the NCI ontology is serialised into the following set of triples, where `_:x` is a blank node:

```
Malaria  rdfs:subClassOf     _:x .
    _:x  rdf:type            owl:Restriction .
    _:x  owl:onProperty      mayHaveFinding .
    _:x  owl:someValuesFrom  Fever .
```

Clearly, it will be cumbersome for services to operate over such structures as well as problematic for non-expert end-users to browse over them. Hence, our converters perform some level of simplification and normalisation whenever this is possible. For example, the above OWL axiom is serialised into the following triple:

`Malaria mayHaveFinding Fever .`

Other examples of performed normalisation are axioms of the form $A \equiv C \sqcap D$ which are normalised into $C \sqcap D \sqsubseteq A, A \sqsubseteq C$, and $A \sqsubseteq D$. From them only those that can be translated to triples without the use of blank nodes are added to our LDG (i.e., the latter two in the above case) while the others are saved in OWL format for potential future use like reasoning.

Another issue is that authors of different data sources choose different ways and names to represent the same information. For example, at least the following properties have been found in various ontologies for encoding synonym labels:

http://snomed.info/field/Description.term.synonym
http://www.geneontology.org/formats/oboInOwl#hasExactSynonym
http://www.w3.org/2004/02/skos/core#altLabel

In order to unify the model, our converters are replacing these labels with the label `skos:altLabel`. Every class has zero or more `skos:altLabel` properties attached to it and exactly one `skos:prefLabel` property per language tag.

Besides schema harmonisation and simplification, the converted ontologies need to also satisfy some logical and structural constraints. First, every converted ontology $\mathcal{O}$ contains exactly one top-level "root" class—that is, a class O_{root} such that for every atomic class $A \in \mathsf{Sig}(\mathcal{O})$ we have $\mathcal{O} \models A \sqsubseteq O_{\mathsf{root}}$ and $O_{\mathsf{root}} \sqsubseteq A \notin \mathcal{O}$. Second, $\mathcal{O}$ must not contain any cluster of equivalent classes—that is, no list of classes $A_1, \dots, A_n$ should exist such that $\{A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_3, \dots, A_n \sqsubseteq A_1\} \subseteq \mathcal{O}$. From a semantic point of view such "loops" are not problematic [1], however, these complicate implementations of graph algorithms like, traversing the subClassOf hierarchy, computing paths between two entities, defining the depth of an ontology, and so one, hence it was decided that loops in imported ontologies would be eliminated. This is done using a depth-first search algorithm which detects them and removes the last subClassOf link.

2.3 Source Updates and Version Management

The medical domain is a very dynamic one and sources are updated very frequently. For example, a new version of SNOMED and UMLS is released every six months while dm+d is released every week. It is critical that updates are imported in the Babylon system as soon as possible since these can provide data on new or retracted drugs, risk factors, diseases, etc. Updates, however, bring the issue of ontology versioning and management. Note that obsolete content cannot be simply removed from the LDG as it may be in use by some services, thus a careful and controlled migration plan is needed. We explored two approaches to ontology versioning:

1. Encode source version in class IRIs. An advantage of this is that the content associated with IRIs never changes and services always receive the same content for the same set of input IRIs. A disadvantage is that, as some services cache IRIs, IRI migration to new versions is required and is a complicated process.
2. Keep IRIs version unaware and manage content updates behind the scenes. An advantage is that, migration is not required, however, when a source is updated services may receive unexpected or different results for the same IRI as content has changed.

From the above we are currently following approach 1. and the main motivation is to ensure that services have a fixed behaviour which does not change with source updates. To implement this approach our converters make IRIs version aware by using the following scheme:

```
bblPrefix:{ontology_id}-{ontology_version}/{resource_id}
```

For example, the class Malaria in NCI versions 17.07e and 17.12 is represented as follows:

http://kb.babylonhealth.com/nci_thesaurus-17.07e/C34797
http://kb.babylonhealth.com/nci_thesaurus-17.12/C34797

This way two different versions of the same data source can co-exist in the LDG. When a new version is integrated, owners of services within Babylon are notified and they start to migrate gradually to the new IRIs. This process can be asynchronous as different services may migrate at different points in time, so the two versions of the same data source may co-exist for even up to six months. After migration is successfully completed the old converted data source is removed from the LDG.

2.4 Upper Level Ontology

Different ontologies and datasets may use different vocabulary and structure in order to represent the same real-world medical concept. For example, to represent medical conditions one ontology may use the class Disease another the class Disorder while another the class ClinicalFinding. For that purposes the use of an upper-level-ontology (ULO) which will provide uniform and source independent access to the underlying LDG has been advocated [7,12]. In the above example, the ULO can contain one class `ulo:Disease` and all the aforementioned classes can be declared to be subclasses of it. The classes and properties in ULO are those entities that are exposed to the services that are using the LDG and are called *Semantic Types*. Every class in the LDG is associated with at least one semantic type.

In Babylon we adopted the UMLS semantic network (SN) [3] as a starting point for our ULO, however, this was subsequently extended or altered when seemed necessary. More precisely, if a data source contained a top-level class that we feel could be interesting to be exposed to the services, then we created a new class in the ULO with the same label and linked the imported ontology class with the new one. For example, UMLS SN does not contain a class for rare diseases whereas the Babylon LDG includes the RareDisease ontology from BioPortal which provide such a grouping of diseases.

Ontologies that we import from UMLS are already associated with entities in ULO (since our ULO originates from the UMLS SN), however, ontologies which we convert using custom converters are not, hence these have to be assigned one. Given an ontology $\mathcal{O}$ and our ULO $\mathcal{O}_u$ the process of assigning Semantic Types to the classes in $\mathcal{O}$ is the following:

1. Identify "top-level" classes in $\mathcal{O}$ that represent a similar real-world notion as some class in $\mathcal{O}_u$. By similar notion we mean that both $\mathcal{O}$ and $\mathcal{O}_u$ contain classes with the same or similar labels; e.g., both contain classes with label "Disease" (same label), or one contains a class with label "Physiological Process" and the other a class with label "Physiological Function" (similar label). In essence this is a manual alignment process.
2. The relations identified in the previous step between a class C in $\mathcal{O}$ and a class D in $\mathcal{O}_u$ are recorded in the converter configuration in the form $\langle C, D\rangle$.
3. The converter for $\mathcal{O}$ reads the configuration and creates for every link $\langle C, D\rangle$ and every $\mathcal{O} \models C' \sqsubseteq C$ the triple C `bbl:hasSTY` D . assigning the Semantic Type D to every "descendant" class of C in $\mathcal{O}$.

A similar approach is followed for properties, however, properties of an imported ontology are linked to properties from ULO using subPropertyOf axioms. For example, "part of" relations in SNOMED, FMA, and the Alzheimer's Disease ontologies are declared to be subPropertyOf the ULO partOf relation while six"has ingredient" relations from different data sources are also linked under the ULO hasIngredient property.

Besides accessing the data through a unique abstract model, the ULO can also be used for checking consistency of the underlying LDG [12]. Some of the top-level classes of the ULO have been declared to be disjoint, e.g., Organism and ManufacturedObject. Then, the following query checks if the LDG contains classes that that are sub-classes of both of them.

```
ask where { ?x rdfs:subClassOf :Organism, :ManufacturedObject . }
```

which should return false.

In total our ULO contains 817 classes, 349 properties, 816 subClassOf axioms and 332 subPropertyOf axioms.

2.5 Source Alignment

An important part of data integration is discovering links (mappings) between the entities of the various different data sources. For example, class Malaria appears in at least 15 different sources in UMLS, as well as in SNOMED, NCI, and more. In each of them complementary information may be described about this condition, e.g., one source may describe its symptoms, another drugs to treat it and so on. It would be beneficial if we can establish mappings between these classes as then, when one queries for information about this disease data from all sources would be returned.

For that purpose a large effort was spent towards pair-wise aligning the imported data sources. Alignments were build by mostly using the mappings included in UMLS (which are used to build the silver standard in the OAEI campaign [13]); e.g., SNOMED Malaria is mapped to the NCI one if they share the same UMLS code. Mappings were initially stored in GraphDB as equivalence axioms, e.g., if class A is mapped to class B then axiom $A \equiv B$ is added. However, it quickly became clear that this approach does not scale as these axioms cause a combinatorial explosion of the inferred statements computed by GraphDB during loading (GraphDB materialises inferences at loading); more precisely, all ancestors and descendants of A (resp. B) become ancestors and descendants of B (resp. A). Our partial solution was a bit unconventional. Instead, we encoded mappings using `owl:sameAs` and used GraphDB's `owl:sameAs` optimisation[3] to significantly reduce the number of inferred statements.

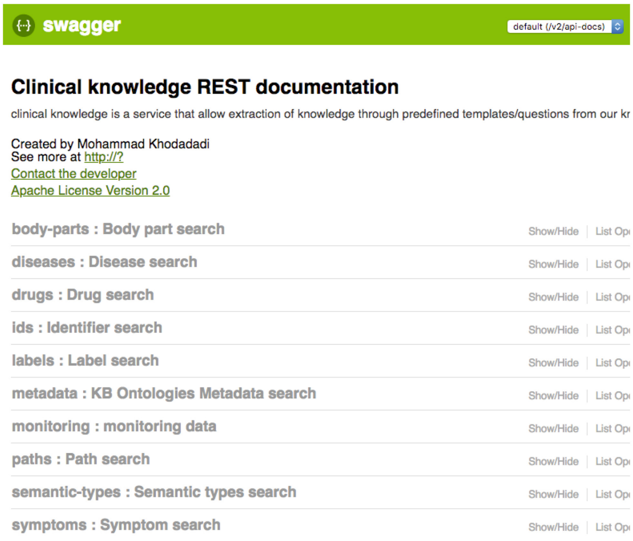
We were able to load about 3 million mappings between entities of the LDG, however, it has become apparent that we have reached the limits of the capabilities of state-of-the-art triple-stores. Loading the LDG with these mappings takes about 36 hours and our attempts to load all computed mappings (about 4.5 million) have failed.[4] Another issue that has been raised is that although mappings help us complement the information that each source contains for each class it also causes duplication and redundancy. If all Malaria classes have a `skos:definition` in all these 15 sources and all of them are linked with `owl:sameAs`, then after reasoning every such class will contain 15 such definitions. Moreover, the ancestors of each class will be the union of the ancestor of each of these classes creating a blow-up in the number of ancestors of a class. To alleviate this issue one needs to built post-processing filtering mechanism on top of the LDG [5,18]. Such mechanisms were implemented, however, initial results show that they do not scale well in practice.

3 Data Usage and Querying

In the current section we provide some details about mid-level services have been built on top of our LDG. These services provide a form of abstraction layer for accessing and browsing the LDG or for comparing classes w.r.t. the knowledge stored in the LDG.

[3] http://graphdb.ontotext.com/documentation/standard/sameas-optimisation.html.

[4] Alternative triple-stores have also been investigated. We have also tried non-materialisation-based systems which although much faster in loading (as they don't perform reasoning) they fail at query time when they perform backward-chaining reasoning.


swagger
default (/v2/api-docs)

Clinical knowledge REST documentation

clinical knowledge is a service that allow extraction of knowledge through predefined templates/questions from our kr

Created by Mohammad Khodadadi
See more at http://?
Contact the developer
Apache License Version 2.0

body-parts : Body part search — Show/Hide | List Op
diseases : Disease search — Show/Hide | List Op
drugs : Drug search — Show/Hide | List Op
ids : Identifier search — Show/Hide | List Op
labels : Label search — Show/Hide | List Op
metadata : KB Ontologies Metadata search — Show/Hide | List Op
monitoring : monitoring data — Show/Hide | List Op
paths : Path search — Show/Hide | List Op
semantic-types : Semantic types search — Show/Hide | List Op
symptoms : Symptom search — Show/Hide | List Op

Fig. 2. Class browsing page.

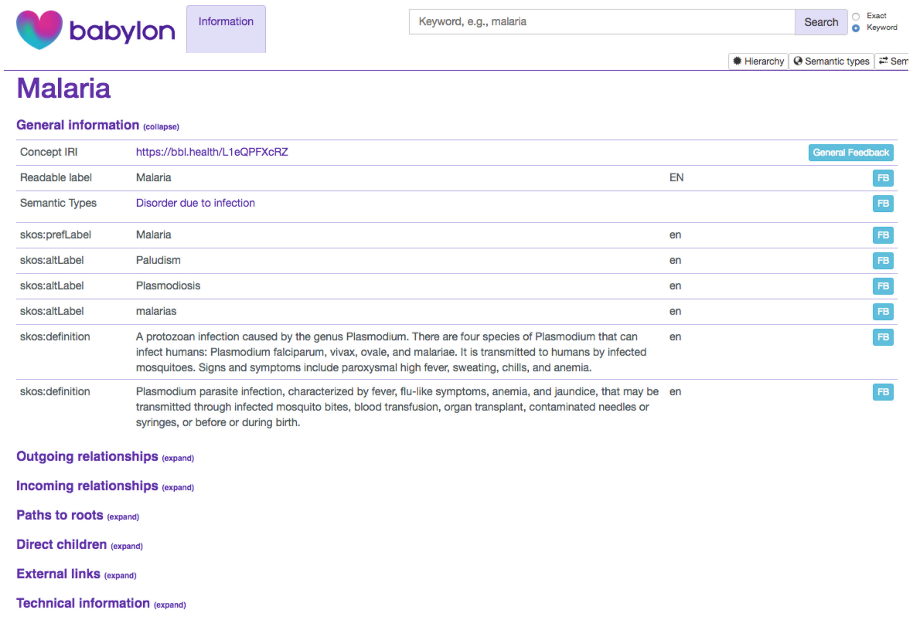
3.1 Querying with Clinical Knowledge

To abstract away from the SPARQL syntax and provide our end-users with the ability to easily retrieve data from the LDG we have built a data access service, called ClinicalKnowledge (CK). CK is using the ULO which, as stated, is the abstract unified model of our medical LDG. CK provides 50 REST services which implement simple or more complex SPARQL queries over our LDG possibly with post-processing on the returned results. Those 50 services are grouped into 10 categories which can be depicted in Fig. 2. The most important services are the following:

- **Disease:** provides services to retrieve information about diseases, e.g., associated diseases, causes of a disease, associated drugs, symptoms, etc.
- **Label:** provides label-based search services, e.g., given some word return a list of IRIs having this word as `skos:prefLabel` or `skos:altLabel`.
- **Semantic Types:** given an IRI, services of this category can be used to retrieve its assigned Semantic Type (i.e., ULO classes), or given a Semantic Type to list all classes annotated with it, and more.
- **Paths:** Given an IRI, services of this group return paths of classes (w.r.t. subClassOf) from that IRI to the top-level or to leaf classes of the LDG.

3.2 Browsing with KB-Explorer

Besides retrieving data in a tabular format using CK, an important part for building services in Babylon or debugging existing ones is to browse through the stored data and understand them. For these purposes an in-house web-browser



Malaria

General information (collapse)

Concept IRI	https://bbl.health/L1eQPFXcRZ		
Readable label	Malaria	EN	FB
Semantic Types	Disorder due to infection		FB
skos:prefLabel	Malaria	en	FB
skos:altLabel	Paludism	en	FB
skos:altLabel	Plasmodiosis	en	FB
skos:altLabel	malarias	en	FB
skos:definition	A protozoan infection caused by the genus Plasmodium. There are four species of Plasmodium that can infect humans: Plasmodium falciparum, vivax, ovale, and malariae. It is transmitted to humans by infected mosquitoes. Signs and symptoms include paroxysmal high fever, sweating, chills, and anemia.	en	FB
skos:definition	Plasmodium parasite infection, characterized by fever, flu-like symptoms, anemia, and jaundice, that may be transmitted through infected mosquito bites, blood transfusion, organ transplant, contaminated needles or syringes, or before or during birth.	en	FB

Outgoing relationships (expand)

Incoming relationships (expand)

Paths to roots (expand)

Direct children (expand)

External links (expand)

Technical information (expand)

Fig. 3. Class browsing page.

for our LDG was implemented, called KB-explorer. A screen-shot of the class page of the KB-explorer for the class Malaria is depicted in Fig. 3.

The following information can be depicted for each class (some of this information is concealed in order not to overwhelm users):

- Preferred and alternative labels of the class with their language tags.
- The semantic types from ULO that have been assigned to this class, in this case DisorderDueToInfection.
- The `skos:definition` associated to the class (in this case there are two definitions coming from different sources).
- The paths in the hierarchy from the current class to top-level classes.
- Direct super/sub-classes of a class as well as its relations with other classes
- External links to DBpedia and/or Wikidata.

Through KB-explorer users can also browse the classes and properties of ULO as well as annotate medical terms in a text using an annotator that has been built in Babylon for experimentation purposes.

3.3 Class Comparison with a Hybrid Lightweight Reasoner

Babylon services produce and consume classes which are constructed using IRIs from the LDG. In several cases, a class expression may have to be formed in order to capture the medical condition of a patient. For example, the LDG contains no pre-defined class for the notion of a "recent injury in left leg" but this can be captured using OWL class expression using SNOMED atomic classes in either of the following ways:

$$C_1 = \mathsf{RecentInjury} \sqcap \exists \mathsf{findingSite}.\mathsf{LeftLeg}$$
$$C_2 = \mathsf{Injury} \sqcap \exists \mathsf{occurred}.\mathsf{Recent} \sqcap \exists \mathsf{findingSite}.\mathsf{LeftLeg}$$

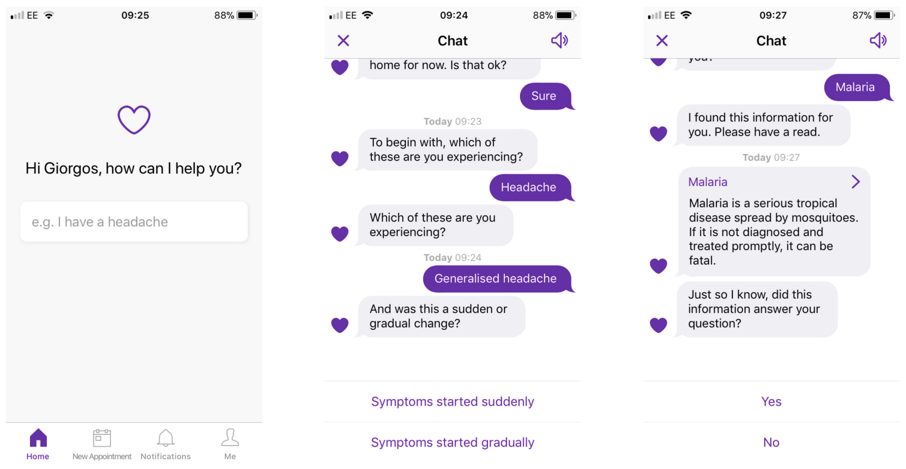
Inevitably, different services within Babylon will use either of the above (and possibly even more) ways to represent this real-world notion. It would be beneficial for interoperability purposes if we can determine that these two classes are equivalent. In theory, this can be done with the help of an OWL reasoner. If our LDG ($\mathcal{O}_{ldg}$) contained the axiom $ax := \mathsf{RecentInjury} \equiv \mathsf{Injury} \sqcap \exists \mathsf{occurred}.\mathsf{Recent}$ then we would indeed have $\mathcal{O}_{ldg} \models C_1 \equiv C_2$. Unfortunately, in a very large number of cases such axioms are missing from $\mathcal{O}_{ldg}$ (in fact, the above axiom does not exist in SNOMED and hence neither in $\mathcal{O}_{ldg}$) and, moreover, the vast size of $\mathcal{O}_{ldg}$ makes it at least challenging to use any of the existing OWL reasoners to perform sub-class reasoning.

For these reasons a custom lightweight approximate reasoner was implemented on top of GraphDB. Since we are mostly dealing with class expressions containing existential quantifiers over which GraphDB is incomplete [6], the reasoner is using some of the consequence-based techniques presented in the literature [14] to improve the inference capabilities of GraphDB. However, it does not implement a complete $\mathcal{EL}$ calculus due to scalability reasons. In addition, in order to tackle the issue with the lack of axioms for classes (like the one mentioned above) it is also using NLP-based knowledge extraction techniques to extract (possibly missing) axioms from class labels. For example, consider class $\mathsf{RecentInjury}$ with preferred label "Recent Injury". Dependency parsing [16] is applied on the label in order to break it into word "Injury" with *modifier* "Recent". Then, Named Entity Disambiguation is applied to associate IRIs from the LDG to each of these words; assume that we successfully pick the IRIs of classes Injury and Recent. Finally, a relation from ULO is selected in an attempt to ideally build the expression $\mathsf{Injury} \sqcap \exists \mathsf{occurred}.\mathsf{Recent}$. This is then used to replace class $\mathsf{RecentInjury}$ in C_1 building C_1' which is essentially the same as C_2 hence being able to determine that the two classes are equivalent.

4 Babylon Chatbot and Triaging

Patient triaging is one of the central automated services offered by Babylon through app's chatbot. Triaging is the process of sorting patients into groups based on their need for immediate medical treatment and can be used in hospital emergency rooms when limited medical resources are only available.

In addition to triaging, Babylon's chatbot also supports general purpose queries, like "Get me info for Malaria" or"What are the symptoms of flu". Figure 4 presents snapshots from Babylon's chatbot. Figure 4a depicts the initial screen prompting the user to enter some text, Fig. 4b depicts a triaging interaction with a user and Fig. 4c an information retrieval one. In order to determine



Hi Giorgos, how can I help you?

e.g. I have a headache

home for now. Is that ok?

Sure

Today 09:23

To begin with, which of these are you experiencing?

Headache

Which of these are you experiencing?

Today 09:24

Generalised headache

And was this a sudden or gradual change?

Symptoms started suddenly

Symptoms started gradually

Malaria

I found this information for you. Please have a read.

Today 09:27

Malaria

Malaria is a serious tropical disease spread by mosquitoes. If it is not diagnosed and treated promptly, it can be fatal.

Just so I know, did this information answer your question?

Yes

No

(a) initial screen (b) triaging interaction (c) general query

Fig. 4. Babylon chatbot

which type of interaction to initiate the original user text goes through a classifier which has been trained in Babylon. In Fig. 4b the user initially entered the phrase "My head hurts" hence initiating a triaging interaction. In contrast, in Fig. 4c the user entered the word "Malaria" and the classifier determined that the user entered a general information retrieval query.

Triaging is implemented with the notion of a *flow*. A flow is a directed graph where every node is a multiple choice question to be asked to the user and the answers determine the subsequent node and question to be asked. Every answer in a node is associated with an OWL class expression which is built using entities from the LDG and possibly also OWL constructors. For example, a node can contain the question "Right, do you have any pain?" with the following potential answers and associated classes:

– "ansId": 1, "txt": "No", None.
– "ansId": 2, "txt": "Yes, in one part of my head", $\mathsf{Headache} \sqcap \exists \mathsf{fSite}.\mathsf{HeadPart}$.
– "ansId": 3, "txt": "Yes, a general headache", GeneralizedHeadache.

Flows are created by in-house doctors using a platform developed in Babylon. OWL constructors that have been used so far are $\neg$, $\sqcap$, $\sqcup$ and $\exists$. For example, the answer "Painful to touch scalp or temples" is associated to the class expression $\mathsf{Tenderness} \sqcap \exists \mathsf{fSite}.(\mathsf{Scalp} \sqcup \mathsf{Temples})$. Doctors have so far created flows for nineteen body parts or medical conditions some of which are, Fever, Chest, Pregnancy, Foot, Mouth, Head, Abdomen, and more. A flow can contain more than 50 nodes and each path in this graph encodes a potential interaction with the user until a conclusion is reached. So far more than 1,000 possible interactions have been encoded in the form of flows. The head-flow used to produce the triaging interaction in Fig. 4b is depicted in Fig. 5.

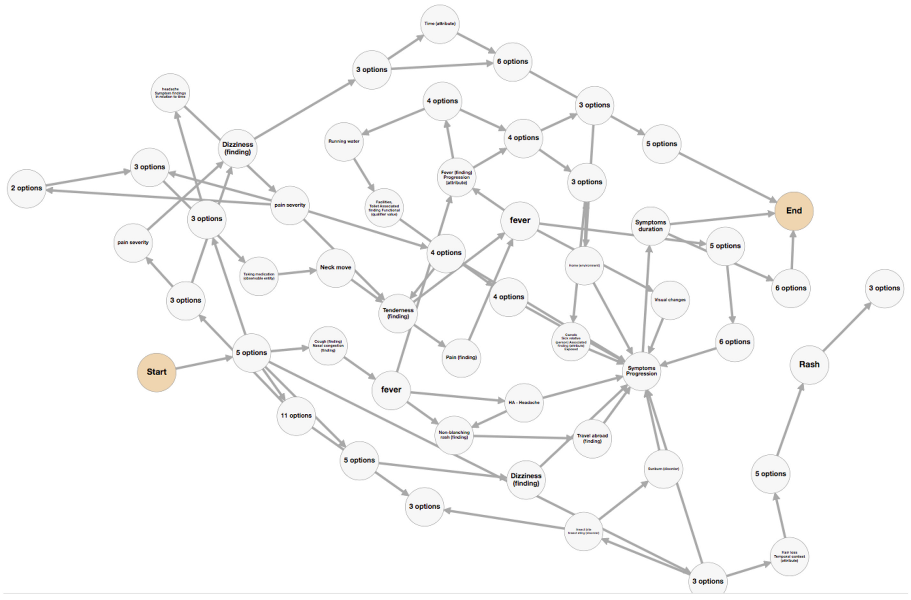


Fig. 5. Head flow

Which flows to activate given some user text is accomplished using our Knowledge-Extraction (KE) and hybrid reasoner methods. From a user text like "I feel a pain around my heart" our KE method extracts the class expression $\mathsf{Pain} \sqcap \exists \mathsf{fSite.Heart}$. Then, the reasoner is used to find all flows which contain nodes that are superclasses or this class; e.g., flows that contain nodes annotated with $\mathsf{ChestPain}$. To evaluate our approach in-house doctors created 680 natural language queries mimicking the text that users would type into the chatbot as well as the list of expected superclasses from the LDG. For example, this test set includes text like"I cut my finger" or "My lower back hurts" while the expected LDG classes are $\mathsf{HurtFinger}$ and $\mathsf{LowerBackPain}$, respectively. We measured 0.967 precision and 0.799 recall. The approach was compared against a Machine Learning one based on sentence embeddings and showed better precision and recall, hence the current setting uses the KE+hybrid reasoner while the sentence embedder is used as a fall-back solution.

So far the Babylon app has been downloaded about 600 K times within the UK. Moreover, 392 in-house or contracted doctors have conducted about 135 K video consultations while there have been about 326 K completed conversations with the chatbot.

5 Lessons Learnt and Conclusions

We have presented our efforts, challenges, design decisions, and solutions to problems faced while trying to use Semantic Web technologies in an industrial-strength healthcare application in Babylon Health. Our main lesson learned is that indeed "Linked Data Is Merely More Data" [12]. Following the best practices described in [12]—that is, an upper-level-ontology, integrity constraints to check validity of data, building abstraction layers on top of SPARQL, etc., does improve usability of the LDG. However, still its vast size and the heterogeneity of the sources makes it hard to maintain a consistent structure, comprehend and work with the data, link them using alignment, and build intelligent applications on top. Even if links are discovered, inferring equivalence and unifying (merging) the linked entities cannot be realised at this scale. Finally, comparing class expressions using traditional reasoning systems is very restrictive and the lack of a complete set of axioms in the background knowledge provides a limited recall.

On the positive side Semantic Technologies help at least in the following aspects. Complex class expressions allow us to dynamically represent almost any medical notion (condition) without the need to pre-define all of them (a task clearly impossible). The use of formal semantics for comparing classes helped us achieve a very high precision while the integration of NLP-based techniques also quite good recall improving on previous purely ML-based approaches. Formal semantics, integrity constraints, and the ULO also allowed us to ensure further data quality and consistency while the release of sources using standards relatively easy to create our LDG. Last but not least, materialisation in triples-stores is helping us infer new knowledge and also execute hierarchy traversal queries in a scalable way.

Regarding future plans we are currently in the process of re-building our LDG from scratch. We will follow a more conservative approach starting with few sources as a seed and enriching them with information from other sources [23]. We are also in the process of further improving our KE and hybrid reasoning approaches as well as enriching medical data sources with new knowledge.

References

1. Baader, F.: Terminological cycles in KL-ONE-based knowledge representation languages. In: Proceedings of the 8th National Conference on Artificial Intelligence, pp. 621–626 (1990)
2. Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. J. Biomed. Inform. **41**(5), 706–716 (2008)
3. Bodenreider, O., McCray, A.T.: Exploring semantic groups through visual approaches. J. Biomed. Inform. **36**(6), 414–432 (2003)
4. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2RDF Release 2: improved coverage, interoperability and provenance of life science linked data. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 200–212. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38288-8_14
5. Chen, B., Dong, X., Jiao, D., Wang, H., Zhu, Q., Ding, Y., Wild, D.J.: Chem2Bio2RDF: a semantic framework for linking and data mining chemogenomic and systems chemical biology data. BMC Bioinform. **11**, 255 (2010)

6. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Completeness guarantees for incomplete ontology reasoners: theory and practice. J. Artif. Intell. Res. **43**, 419–476 (2012)
7. Dumontier, M., Baker, C.J.O., Baran, J., Callahan, A., Chepelev, L.L., Cruz-Toledo, J., Rio, N.R.D., Duck, G., Furlong, L.I., Keath, N., et al.: The semantic-science integrated ontology (SIO) for biomedical research and knowledge discovery. J. Biomed. Semant. **5**, 14 (2014)
8. Elmer, S., Jrad, F., Liebig, T., ul Mehdi, A., Opitz, M., Stauß, T., Weidig, D.: Ontologies and reasoning to capture product complexity in automation industry. In: Proceedings of the 16th International Semantic Web Conference Posters & Demonstrations and Industry Tracks (2017)
9. Etzioni, O., Banko, M., Cafarella, M.J.: Machine reading. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), pp. 1517–1519 (2006)
10. Golbeck, J., Fragoso, G., Hartel, F.W., Hendler, J.A., Oberthaler, J., Parsia, B.: The national cancer institute's thésaurus and ontology. J. Web Semant. **1**(1), 75–80 (2003)
11. Golbreich, C., Grosjean, J., Darmoni, S.J.: The foundational model of anatomy in OWL 2 and its use. Artif. Intell. Med. **57**(2), 119–132 (2013)
12. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked data is merely more data. In: Linked Data Meets Artificial Intelligence, Papers from the 2010 AAAI Spring Symposium, Technical Report SS-10-07, Stanford, California, USA (2010)
13. Jiménez-Ruiz, E., Grau, B.C., Horrocks, I.: Exploiting the UMLS metathesaurus in the ontology alignment evaluation initiative. In: Proceedings of the 2nd International Workshop on Exploiting Large Knowledge Repositories (2012)
14. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), pp. 2040–2045 (2009)
15. Knoblock, C.A., et al.: Lessons learned in building linked data for the American art collaborative. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 263–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_26
16. Kübler, S., McDonald, R.T., Nivre, J.: Dependency Parsing. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, San Rafael (2009)
17. McCray, A., Nelson, S.: The representation of meaning in the UMLS. Meth. Inform. Med. **34**, 193–201 (1995)
18. Nolin, M., Dumontier, M., Belleau, F., Corbeil, J.: Building an HIV data mashup using Bio2RDF. Brief. Bioinform. **13**(1), 98–106 (2012)
19. Salvadores, M., Alexander, P.R., Musen, M.A., Noy, N.F.: Bioportal as a dataset of linked biomedical ontologies and terminologies in RDF. Semant. Web **4**(3), 277–284 (2013)
20. Schriml, L.M., et al.: Disease ontology: a backbone for disease semantic integration. Nucleic Acids Research **40(Database–Issue)**, 940–946 (2012)
21. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1), 158–176 (2013)
22. Spackman, K.A., Campbell, K.E., Côté, R.A.: SNOMED RT: a reference terminology for health care. In: American Medical Informatics Association Annual Symposium, AMIA 1997, Nashville, TN, USA, October 25–29, 1997 (1997)
23. Stoilos, G., Geleta, D., Shamdasani, J., Khodadadi, M.: A novel approach and practical algorithms for ontology integration. In: Proceedings of 17th International Semantic Web Conference (ISWC) (2018)