BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–2

Brain: Biomedical Knowledge Manipulation

Samuel Croset [1,*], Robert Hoehndorf [2], John Overington [1] and Dietrich Rebholz-Schuhmann [1]

[1]European Bioinformatics Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SD UK
[2]Department of Genetics, University of Cambridge, Downing Street, Cambridge, CB2 3EH, UK

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT
Summary: Brain is a library facilitating the creation and manipulation of ontologies and knowledge bases represented in the Web Ontology Language (OWL).
Availability and Implementation: The Java source code and the library are freely available at https://github.com/loopasam/Brain and on the Maven Central repository (GroupId: uk.ac.ebi.brain). The documentation is available at https://github.com/loopasam/Brain/wiki.
Contact: croset@ebi.ac.uk
Supplementary information: Links to additional figures/data available on a web site, or reference to online-only Supplementary data available at the journal's web site.

1 MOTIVATION

Relational databases hold most of the available structured biomedical information. The content of these repositories is often extracted from scientific literature by manual curation with the help of text-mining tools. The transformation from raw text into structured data is most important, as the curated information can then be classified, managed and queried more easily. Databases facilitate the re-use of previous work in a computer-friendly manner and support the biomedical knowledge to scale-up. In order to leverage further more the existing information, the current trend is at data integration and interoperability, with large projects such as ELIXIR leading the way. The underlying idea assumes that increasingly complex biomedical challenges such as finding new treatments for diseases could be better addressed by combining the content of independant repositories over the Internet. Knowledge bases, an idea coming from pure computer science, would be a solution to improve the interoperability and the value of the data yet no framework is available at the time of writting for the biomedical domain. Brain, the library presented in this manuscript, addresses this matter and provides a simplified Java interface dedicated to life science, to handle and query knowledge bases.

2 KNOWLEDGE BASES

Traditional relational databases are an obstacle to realize a large-scale data integration, mostly because of their lack of support for interoperability: The schema structuring the data is indeed very repository-specific which limits the combination of the native content with external data in an efficient and meaningful fashion. In order to address this issue, a series a standard forming the semantic web have been developped. One of them, the Resources Description Framework (RDF) enables the exposition of the underlying structure as part of the data themselves. This representation relies on triples as building block, composed as *subject - relation - object* which serves to describe the data as well as their types. More complicated data structures, such as sub-class or transitive relationships can be further expressed via another standard, the Web Ontology Language (OWL). OWL derives from description logic and has been designed to capture the knowledge of a domain of interest in the form of a structured vocabulary. This feature makes it particularly interesting from the point of view of life science, as a lot of ontologies and classification have been devellopped since the origin of the discipline. OWL is often expressed in combination with RDF in order to reveal the underlying schema of the data, but it can also be used as such, as a computer implementation of description logic. Knowledge bases and ontologies can therefore be built without being necessary expressed as RDF triples while still preserving all the advantages in regards to data integration and interoperability. Brain aims at facilitating the construction and manipulation of such knowledge bases (referred also as ontologies), rather than being oriented towards the consumption of RDF data. We will present first the biomedical motivation for the particular subset of OWL supported by Brain so called OWL 2EL. Then will be discussed the main features implemented by the library in regards to this profile.

3 SCALABILITY AND COMPLEXITY

Knowledge representation in the biomedical domain differs from other discipline, because of the type and abundance of information that can possibly be interconnected. Brain appreciates this aspect and focuses on a particular profile of OWL, called EL which consists of a subset of the constructs available in the original language. This profile is designed to be *tractable*, meaning that the constructs available have a polynomial complexity, easier to compute than the full version of OWL. These constructs are called *axioms* and are the fundamental block behind OWL knowledge bases. Axioms assert the facts and relations present in the knowledge base and are understood by a computer program named *reasoner*. Based on the logical structure of the axioms, the reasoner is indeed capable of deriving new facts from the asserted ones as

*to whom correspondence should be addressed well as retrieving some implicit information, enabling powerful query mechanisms surpassing the Structured Query Language (SQL) expressivity. Brain is supporting primarily the OWL 2EL profile for its computational properties, as it is suitable for real-life biomedical applications, where millions of axioms could be potentially extracted from a repository such as Chembl. Moreover, the profile is expressive enough for a good portion of the biomedical knowledge: Most of the ontologies such as the Gene Ontology or Chebi are already included in this profile and any relational model can be easely converted and represented using OWL 2EL, opening doors for large scale meaningful data integration. Brain builds on the top of Elk, a fast reasonner dedicated to EL ontologies. Elk shows very good performances at classifying large datasets and is implemented in a multi-threaded friendly way. It indeed offers the possibility to run some reasoning tasks in parallel. Clusters or multicore architecture can therefore scale the speed of reasoning as more data are added to the knowledge base, common situtation for biomedical repositories. Brain wraps and simplifies the interaction with Elk while still leaving the possibility to fine tune the configuration for advanced users.

4 PROGRAMMATIC FEATURES

In practice, the implementation of OWL ontologies and knowledge bases can be done either in a programmatic way via the OWL-API or with the help of a visual tool such as Protege (itself built from the OWL-API). The graphical interface of Protege allows the user to focus plainly on the generation of axioms rather than on the underlying programmatic implementation. OWL axioms and expressions can indeed be entered with the user-friendly Manchester syntax and the name of the classes are displayed in a convenient way for instance. More complex applications requiring a deeper control over the ontology can employ the OWL-API but it requires a fairly good understanding of Java and can be daunting for the new comers. The API deals in great details with all the aspects around ontology generation for semantic web purposes, nonetheless the level of detail can be cumbersome for some applications. Brain aims at filling the gap between the OWL-API and graphical interfaces: It is designed as a facade, providing a series of convenience methods for the common use-cases encountered in the biomedical domain and leveraging the access to the OWL-API. Table 1 highlights some typical OWL constructs with an example of implementation using Brain. The full list of supported constructs is available as supplementary material. Brain relies on the intuitive and explicit Manchester syntax to formulate OWL class expressions, just like in the Protege editor. The interaction with the OWL-API resolves around strings rather than Java objects, making Brain suitable to parse and answer requests in the context of a web service or an OWL end-point for instance. Using strings of characters as input speeds as well the production and flexibility of the code written to move to an OWL representation from a relational or flat file database for instance. The library supports the loading and referencing of external ontologies in order to integrate and reason over data coming from different sources. An important feature of Brain is the query mechanism, which is very similar to the one implemented in Protege. Powerful questions can be formulated over the knowledge base using the Manchester syntax, abstracting away a complex interaction with the Java object provided by the OWL-API.

Table 1. Example of some common OWL 2EL constructs written using the Manchester syntax alonside an example of implementation using Brain.

OWL	Brain implementation
Class: A	brain.addClass(”A”);
ObjectProperty: P	brain.addObjectProperty(”P”);
C and D	brain.getEquivalentClasses(”C and D”);
owl:Thing	brain.getOWLClass(”Thing”);
P some C	brain.subClassOf(”X”, ”P some C”);
C SubClassOf: D	brain.subClassOf(”C”, ”D”);
S SubPropertyChain: R1 o R2	brain.chain(”R1 o R2”, ”S”);
R Characteristics: Transitive	brain.transitive(”R”);

5 CONCLUSION

Why is brain interesting for the community tool to build KB scalable solution in order to preapre the next stage of data integration with KB reasoner driven and semantic web.

ACKNOWLEDGEMENT

blabalba

Funding: This work was supported by

REFERENCES