Ontology-based Search of Genomic Metadata

Javier D. Fernández[1], Maurizio Lenzerini[1],

Marco Masseroli[2]*, Francesco Venco*[2]*, Stefano Ceri*[2]
{*fernandez,lenzerini*}*@dis.uniroma1.it*
{*stefano.ceri,marco.masseroli,francesco.venco*}*@polimi.it*
[1]*Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti,*
Sapienza Università di Roma, Italy
[2]*Dipartimento di Elettronica, Informazione e Bioingegneria,*
Politecnico di Milano, Italy

May 27, 2015

Abstract

The Encyclopedia of DNA Elements (ENCODE) is a huge and still expanding public repository of more than 4,000 experiments and 25,000 data files, assembled by a large international consortium since 2007; unknown biological knowledge can be extracted from these huge and largely unexplored data, leading to data-driven genomic, transcriptomic and epigenomic discoveries. Yet, search of relevant datasets for knowledge discovery is limitedly supported: metadata describing ENCODE datasets are quite simple and incomplete, and not described by a coherent underlying ontology. Here, we show how to overcome this limitation, by adopting an ENCODE metadata searching approach which uses high-quality ontological knowledge and state-of-the-art indexing technologies. Specifically, we developed *S.O.S. GeM* (`http://www.bioinformatics.deib.polimi.it/SOSGeM/`), a system supporting effective semantic search and retrieval of ENCODE datasets. First, we constructed a Semantic Knowledge Base by starting with concepts extracted from ENCODE metadata, matched to and expanded on biomedical ontologies integrated in the well-established Unified Medical Language System; we prove that this inference method is sound and complete. Then, we leveraged the Semantic Knowledge Base to semantically search ENCODE data from arbitrary biologists' queries; this allows correctly finding more datasets than those extracted by a purely syntactic search, as supported by the other available systems. We empirically show the relevance of found datasets to the biologists' queries.

1 Introduction

Continuous improvements of Next Generation Sequencing (NGS) technologies in quality, cost of results[1] and sequencing time are leading shortly to the possibility of sequencing an entire human genome in few minutes for a cost of less

[1] `http://www.genome.gov/sequencingcosts/`

than $1,000 [17, 25]. As a consequence, very large-scale sequencing projects are emerging, including the *1000 Genomes Project*, aiming at establishing an extensive catalog of human genomic variation [1], *The Cancer Genome Atlas* (TCGA), a full-scale effort to explore the entire spectrum of genomic changes involved in human cancer [28], and the *Encyclopedia of DNA elements* (ENCODE) [15].

The ENCODE project is the most general and relevant world-wide repository fueling basic biology research. It provides public access to more than 4,000 experimental datasets, including the just released data from its Phase 3, which comprise hundreds of experiments of mainly RNA-seq, ChIP-seq and DNase-seq assays in human and mouse. The high availability of many different genomic features in distinct conditions and of a new generation of bioinformatics systems [23, 20] enables the discovery of genetic and epigenetic phenomena, offering huge opportunities for a variety of applications (notably cancer research).

But availability of ENCODE datasets is not effective in the lack of adequate search systems. Unfortunately, while the quality of experimental data is typically very high, the documentation and associated metadata is not comparatively rich or equally curated, resulting in a difficulty to locate all the experimental data corresponding to a given phenomena. Current interfaces to ENCODE data, available from both the UCSC Genome Bioinformatics site[2] and ENCODE Project Portal[3], provide very useful exploration, browsing, visualization and downloading functionalities, but partial support for metadata extraction and only limited search capabilities; to date, the evaluation of an ENCODE data search query is strictly based only on syntactic (textual) matching of search terms. A state-of-the-art "*syntactic-based*" retrieval system allows to retrieve a set of similar results (e.g. on the basis of well-known technologies such as *Apache Lucene*[4]); yet, this functionality is again based only on string distances. This prevents finding items described with synonyms or semantic variants of the query terms used. Additional support for such advanced search capabilities is needed in order to significantly increase the number and quality of relevant datasets found.

In this work, we overcome current limitations in the search for ENCODE datasets by supporting ontology-based search of their metadata. For our genomic and semantic purposes, we consider the *global ontology* provided by the Unified Medical Language System (UMLS) [7], which collects and integrates well-established biomedical ontologies.

Intuitively, our approach relies on semantically annotating the metadata about ENCODE datasets by means of UMLS, and completing the information by materializing inferred facts, i.e. by performing the semantic closure [19] of such annotations. Then, we set up an abstraction layer to allow users searching relevant ENCODE experiments from text-based queries.

This approach has been practically implemented in **S.O.S. GeM**[5], standing for *Sapienza Ontology-based Search of Genomic Metadata*. Thus, the S.O.S. GeM system builds a *Semantic Knowledge Base* (SKB) of ENCODE metadata that includes the concepts extracted from ENCODE experiment metadata and their associated concepts inferred by using the UMLS global ontology. Then,

[2] `http://genome.ucsc.edu/ENCODE/`
[3] `http://www.encodeproject.org/`
[4] `https://lucene.apache.org/`
[5] `http://www.bioinformatics.deib.polimi.it/SOSGeM/`

S.O.S. GeM provides an intuitive Web-based interface in which users are simply asked for a text query. The proposed query answering algorithm inspects the user query for both UMLS concepts and interesting syntactic tokens, obtaining the relevant set of ENCODE metadata from the SKB.

S.O.S GeM is part of a larger project called **GenData 2020**[6], which has recently produced a high-level, declarative **GenoMetric Query Language** (**GMQL**)[7] [20] for querying heterogeneous NGS data. In other words, the user query answers semantically computed by S.O.S. GeM can be directly routed to the GMQL query processing engine, serving an integrated semantic access for fine-grained genomic queries.

This paper is structured as follows. Section 2 provides some background on the GenData 2020 project, and Section 3 presents the state of art. Then, Section 4 dwells into the description of our proposed solution, discussing the ontology construction, then the query formulation and the query processing algorithm, including the proof of soundness and completeness. Section 5 illustrates the system implementation, and Section 6 presents the system evaluation both in terms of size of involved data and of offline and online performance; Section 7 concludes.

2 Background: The GenData 2020 Project

GenData 2020[6] is a large project sponsored by the Italian Ministry of University and Research, advocating a new, holistic approach to genomic data management that uses cloud-based computing. Data are organized as datasets consisting of several data samples, each containing many genomic regions, where each sample is associated with different experimental conditions described by its metadata; one can think to samples as objects and to datasets as their containers. Our objective is not to address raw data processing, but rather to embrace all processed data formats through an interoperable data model, and by enabling queries over tens of datasets, hundreds or even thousands of data samples and several millions of genomic regions - thereby opening genomics to big data management.

GenData 2020 has adopted a new data model, called **Genomic Data Model** (**GDM**) [20], providing two fundamental abstractions for each data sample:

- **Metadata** describe the biological and clinical properties associated with each sample, e.g. experimental condition, cell line, biological sample, antibody used in experiment preparation, considered antibody target, and also patient phenotype when data have clinical nature. Due to the great heterogeneity of the metadata information that can be associated with each sample, they are represented as arbitrary attribute-value pairs.

- A **region** corresponds to all the DNA nucleotides whose position is between the region left and right ends, typically within a chromosome; in general, we do not include a full nucleotide sequence within the region data, but rather we store high-level properties of the region, which are produced by the post-processing of sequencing data.

[6] `http://www.bioinformatics.deib.polimi.it/GenData/`
[7] `http://www.bioinformatics.deib.polimi.it/GMQL/`

Gendata 2020 has also defined and implemented a new, high-level query language for bio-informaticians, called **GenoMetric Query Language** (**GMQL**)[7] [20], which enables building new datasets from a repository of existing datasets. S.O.S. GeM can be used as the first component of GMQL query execution workflow, by producing an enhanced set of ENCODE experiments corresponding to the query conditions; of course, it can also be used stand-alone[5].

3 State of the Art

In the last decade, semantic developments and biology research are following intersecting paths. A nice overview on big biological databases, bio-ontologies and knowledge discovery problems can be found in [12, 2, 18, 10]. In particular, ontology-based access to biological repositories is a relevant but challenging area. In [29], Xuan *et al.* proposed an ontology-based exploratory system, called *PubOnto*, to enable the interactive exploration and filtering of search results in the medical publication database *Medline*, using multiple ontologies taken from the well-established *Open Biological and Biomedical Ontologies* (OBO) foundry[8]. The authors developed a general purpose ontology to free-text mapping which relies on the pre-generation of lexical variations, word order permutations of ontology terms, their synonyms and a suffix-tree based string matching algorithm.

The *Gene Ontology* (GO) project [4], founded in 1998, is a notable collaborative effort to address the need for consistent descriptions of gene products. The GO project has developed three ontologies that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner. An interesting application that make use of the GO is *GoPubMed* [14]. It is a service that submits keywords to the PubMed repository of medical publication abstracts, extracts GO terms from the retrieved abstracts and presents the induced ontology for browsing; such ontology is the minimal GO subset which comprises all the GO terms found in the retrieved documents.

A somehow similar approach was developed by Müller *et al.* [22] in *Textpresso*, a text-mining system for scientific literature. It splits papers into sentences and sentences into words or phrases. Each word or phrase is then labeled using the eXtensible Markup Language (XML) according to the lexicon of the Textpresso ontology, mainly built from GO. Sentences are indexed with respect to labels and words to allow a rapid search for sentences that have a desired label and/or keyword.

The *Conceptual Open Hypermedia Service* (COHSE) proposed by Bechhofer *et al.* in [5] permits to take advantage of the common understanding provided by an ontology by constructing hypertext structures using the information that the ontology encodes. The COHSE Agent takes documents/pages in and provides back to the user pages enhanced with links; such links are derived through the use of an ontology and associated lexicon along with a mapping from concepts to possible link targets.

Finally, several works explore the query expansion strategy in the medical domain to improve the average precision and recall of user queries. Zhu *et al.* [30], and more recently Thesprasith and Jaruskulchai [27], first identify medical terms in the query using a basic lexical tool and match them to MeSH ontology

[8] `http://www.obofoundry.org/`

concepts. Then, they expand the found concepts by adding UMLS co-concepts, i.e. semantic terms that often appears together. In order to restrict the query expansion, they rank expanded terms by frequency and apply some thresholds. In an earlier work, Díaz-Galiano *et al.* [13] performed a simpler approach and used MeSH descriptors to expand the queries on a collection composed of images and text. All these approaches, though, report slight improvements and are applied to limited domains and ontologies, which may show scalability issues at large scale once term expansion largely increases. A review on ontology-based query expansion can be found in [6].

Complementary works focus on ontology-based automatic annotation. In [26], Taboada *et al.* address semantic annotation of relevant literature about rare disease patients. They identify concepts by means of the entity recognizer *Mgrep* [11] (a comparison between Mgrep and *MetaMap* is performed in [24]) and expand them using the hierarchical structure of their considered OBO ontologies. However, it is not possible to test the correctness of the approach as neither details on the closure nor a formal semantics are given.

4 Ontology-Based Search: Description of our Solution

We base the task of defining an ontology to support the search for genomic datasets on two important items that we assume available:

- A set M of metadata about a collection of genomic experiments and samples (or data files). In particular, for a sample S of an experiment and for a metadata attribute that is present in the collection, M specifies which is the value associated to S by the attribute.
- An ontology $\mathcal{G}$, called the *global ontology*, which is the union of all the ontologies that are considered relevant for describing the domain of genomics, and are accessible through UMLS.

Let $(\mathtt{A}, v)$ be the metadata pair associated with a sample S in M, specifying that v is the value associated with S by the attribute $\mathtt{A}$. From such value v, which can always be seen as a string, the following data are extracted:

- a set $t^{\mathtt{A}}_S(M)$ of strings, called tokens, representing relevant substrings of the string v;
- a set of classes $c^{\mathtt{A}}_S(M)$ that are present in the global ontology, representing relevant classes that, either implicitly or explicitly, are mentioned in v, according to a text analysis of v.

We will make use of the above two notions in the following, where the mechanism to extract $t^{\mathtt{A}}_S(M)$ and $c^{\mathtt{A}}_S(M)$ for the various $\mathtt{A}$ and S are described.

Given the metadata set M and the global ontology $\mathcal{G}$, we build the ontology $\mathcal{O}_{\mathcal{M},\mathcal{G}}$ on the basis of M and $\mathcal{G}$. When M and $\mathcal{G}$ are understood, we simplify the notation and denote the ontology by $\mathcal{O}$. To specify $\mathcal{O}$, we use the ontology language `OWL 2 QL`, a profile of `OWL 2` derived from the `DL-lite` family [8], specifically designed to perform ontology-based accesses to big data collections while keeping inferences tractable. We refer the reader to [21] for a complete

description of OWL 2 QL. Here, we only recall some of the most important notions. A *class* (or *concept*) is a unary predicate representing a set of individual objects (or simply individuals) in the domain of interest, called its instances. A *data property* is a binary predicate representing an attribute, i.e. a relation associating with the individuals a set of values of a particular type (i.e. a string). An *object property* is a binary predicate representing a relationship between classes, i.e. a relation between the instances of such classes.

Given an alphabet Σ for individuals, classes, data properties and object properties, an ontology in OWL 2 QL is a set of OWL 2 QL axioms, where each axiom is a formula over the alphabet Σ. Here is a list of the types of OWL 2 QL axioms that will be of particular importance for us, together with their intuitive semantics:

- `SubClassOf ( C D )`, stating that all instances of class C are also instances of class D;
- `SubObjectPropertyOf ( R Q )`, stating that all pairs of objects that are instances of the object property R are also instances of the object property Q;
- `SubDataPropertyOf ( A1 A2 )`, stating that all pairs that are instances of the data property A_1 are also instances of the data property A_2;
- `ObjectPropertyDomain ( R C )`, stating that, in all the pairs that are instances of the object property R, the first component of the pair is an instance of the class C;
- `ObjectPropertyRange ( R C )`, stating that, in all the pairs that are instances of the object property R, the second component of the pair is an instance of the class C;
- `DataPropertyDomain ( A C )`, stating that, in all the pairs that are instances of the data property A, the first component of the pair is an instance of the class C;
- `ClassAssertion ( C a )`, stating that a is an instance of class C;
- `ClassAssertion ( ObjectSomeProperty ( R C ) a )`, stating that there is an object x such that (i) (a, x) is an instance of the object property R, and (ii) x is an instance of C;
- `ObjectPropertyAssertion ( R a b )`, stating that (a, b) is an instance of the object property R;
- `DataPropertyAssertion ( A a s )`, stating that (a, s) is an instance of the data property A.

The formal semantics of OWL 2, and therefore of OWL 2 QL, is based on the classical notion of interpretation in logic. An *interpretation* $\mathcal{I}$ for an ontology defined over the alphabet Σ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of $\mathcal{I}$, and $\cdot^{\mathcal{I}}$ is the interpretation function of $\mathcal{I}$, i.e. a function assigning an element of $\Delta^{\mathcal{I}}$ to every constant in Σ, a subset of $\Delta^{\mathcal{I}}$ to every class in Σ, a set of pairs of elements in Σ for every data property and every object property in Σ. We now specify when an interpretation $\mathcal{I}$ satisfies an axiom α, written $\mathcal{I} \models \alpha$.

- $\mathcal{I} \models$ `SubClassOf ( C D )` if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I} \models$ `SubObjectProperty ( R Q )` if $R^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$.
- $\mathcal{I} \models$ `SubDataPropertyOf ( A1 A2 )` if $A_1^{\mathcal{I}} \subseteq A_2^{\mathcal{I}}$.
- $\mathcal{I} \models$ `ObjectPropertyDomain ( R C )` if for all $(a, b) \in R^{\mathcal{I}}$, we have that $a \in C^{\mathcal{I}}$.
- $\mathcal{I} \models$ `ObjectPropertRange ( R C )` if for all $(a, b) \in R^{\mathcal{I}}$, we have that $b \in C^{\mathcal{I}}$.
- $\mathcal{I} \models$ `DataPropertyDomain ( R C )` if for all $(a, s) \in R^{\mathcal{I}}$, we have that $a \in C^{\mathcal{I}}$.
- $\mathcal{I} \models$ `ClassAssertion ( C a )` if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I} \models$ `ClassAssertion ( ObjectSomeProperty ( R C ) a )` if there exists $b \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, b) \in R^{\mathcal{I}}$, and $C^{\mathcal{I}}$.
- $\mathcal{I} \models$ `ObjectPropertyAssertion ( R a b )` if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.
- $\mathcal{I} \models$ `DataPropertyAssertion ( A a s )` if $(a^{\mathcal{I}}, s^{\mathcal{I}}) \in A^{\mathcal{I}}$.

As usual, we say that $\mathcal{I}$ is a *model* of $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, if all the axioms of $\mathcal{O}$ are satisfied by $\mathcal{I}$, and we say that an axiom α is logically implied by an ontology $\mathcal{O}$, written $\mathcal{O} \models \alpha$, if α is satisfied by every model of $\mathcal{O}$.

4.1 The Ontology

We now turn our attention to the issue of how to build the ontology $\mathcal{O}$ in our setting. As we said before, we base the task of defining $\mathcal{O}$ on the metadata M and the global ontology $\mathcal{G}$. We first specify the alphabet of $\mathcal{O}$ and then we illustrate its axioms.

The alphabet of $\mathcal{O}$ is constituted by the set $\Sigma_{\mathcal{G}}$ of class symbols that are present in the global ontology $\mathcal{G}$, plus another set of symbols, disjoint from $\Sigma_{\mathcal{G}}$, comprising the following:

- the individual object symbols, at least one for each experiment and one for each sample represented in M,
- the constants of type string, at least one for each token present in M,
- the class symbols `Experiment` and `Sample`,
- the object property symbol `consistsOf`,
- the data property symbols `experimentType` and `dccAccession`,
- the data property symbol `hasValueFor-A`, for each attribute `A` that is present in the metadata set M,
- the object property symbol `hasLinkTo-A`, for each attribute `A` that is present in the metadata set M,
- the data property symbol `hasAssociatedValue`,

- the object property symbol `hasAssociatedObject`.

Intuitively, the class `Experiment` represents all experiments managed by the knowledge base, whereas the class `Sample` represents the samples associated to the experiments. The object property `consistsOf` associates to each experiment the corresponding samples. The data properties `experimentType` and `dccAccession` model properties of experiments: for each instance of `Experiment`, `experimentType` provides its type, and `dccAccession` provides a value identifying such instance. The data property `hasValueFor-A` represents the association between S and the tokens in $t^{\mathtt{A}}_S(M)$. On the other hand, the object property `hasLinkTo-A` represents the association between S and a set of objects that are instances of the various classes in $c^{\mathtt{A}}_S(M)$. Note that we do not have an exact knowledge about such objects. Therefore, we model them using the notion of existential quantification in logic, and more precisely in `OWL 2 QL`. In particular, if `C` is a class in $c^{\mathtt{A}}_S(M)$, then we sanction that there is some object B such that the pair (S, B) is an instance of the relationship `hasLinkTo-A`, and B is an instance of `C`.

Taking into account the above intuitive considerations, we now provide the precise definition of the axioms of the ontology $\mathcal{O}$ on the basis of M and $\mathcal{G}$, as follows.

- All the axioms of $\mathcal{G}$ are in $\mathcal{O}$; we observe that all the axioms of this category are of the form `SubClassOf ( C D )`.
- For every experiment E in M, the axiom `ClassAssertion ( Experiment e )` is in $\mathcal{O}$, where `e` is the individual representing E. The axiom simply states that `e` is an instance of the class `Experiment`.
- For every sample S associated to the experiment E in M, the axioms `ClassAssertion ( Sample s )`, `ObjectPropertyAssertion ( consistsOf e s )` are in $\mathcal{O}$, where `s` and `e` are the individuals representing S and E, respectively. These axioms state that `s` is an instance of the class `Sample`, and that `e` is linked to `s` by the object property `consistsOf`.
- The axioms `ObjectPropertyDomain ( consistsOf Experiment )`, and `ObjectPropertyRange ( consistsOf Sample )` are in $\mathcal{O}$, stating that the domain and the range of the relation `consistsOf` are `Experiment` and `Sample`, respectively. This axiom simply sanctions that the relation `consistsOf` connects experiments to samples.
- For every data property `hasValueFor-A`, the axiom `SubDataPropertyOf ( hasValueFor-A hasAssociatedValue )` is in $\mathcal{O}$, stating that `hasValueFor-A` is a subset of the data property `hasAssociatedValue`. In other words, the data property `hasAssociatedValue` collects all the pairs that are instances of some `hasValueFor-A`.
- The axioms `DataPropertyDomain ( hasAssociatedValue Sample )` is in $\mathcal{O}$, stating that the data property `hasAssociatedValue` is defined on the class `Sample`. Note that this implies that every data property `hasValueFor-A` is also defined on the class `Sample`.
- For every object property `hasLinkTo-A`, the axiom `SubObjectPropertyOf ( hasLinkToA hasAssociatedObject )` is in $\mathcal{O}$, stating that `hasLinkTo-A`

is a subset of the object property `hasAssociatedObject`. Thus, the object property `hasAssociatedValue` collects all the pairs that are instances of some `hasLinkTo-A`.

- The axioms `ObjectPropertyDomain ( hasAssociatedObject Sample )` is in $\mathcal{O}$, stating that the object property `hasAssociatedObject` is defined on the class `Sample`. Note that this implies that every object property `hasLinkTo-A` is also defined on the class `Sample`.

- For every data property `hasValueFor-A`, for every `s` asserted to be an instance of `Sample` in $\mathcal{O}$, and for every value `v` in $t_{\mathtt{s}}^{\mathtt{hasValueFor\text{-}A}}(M)$, the axiom `DataPropertyAssertion ( hasValueFor-A s v )` is in $\mathcal{O}$.

- For every object property `hasLinkTo-A`, for every `s` asserted to be an instance of `Sample` in $\mathcal{O}$, and for every class `C` in $t_{\mathtt{s}}^{\mathtt{hasLinkTo\text{-}A}}(M)$, the axiom `ClassAssertion ( ObjectSomeProperty ( hasLinkTo-A C ) s )` is in $\mathcal{O}$.

4.2 Queries

We now define the class of queries that we are interested in, which we call *search queries*. We express search queries in SPARQL, and we obviously assume the `OWL 2` Direct Semantics entailment regime for such query. Actually, we first define a subclass of search queries, the subclass of *positive search queries*. Intuitively, a query of this class asks for all samples that are related, by means of any attribute in the metadata, to a given set of tokens and to a given set of classes. Formally, given tokens $v_1, \ldots, v_m$ and classes $C_1, \ldots, C_n$ that we want to be related to the samples we are searching for, the corresponding positive search SPARQL query is defined as:

`select` $?x$
`where` { $\alpha_1 . \cdots . \alpha_m . \beta_1 . \gamma_1 \cdots . \beta_n . \gamma_n$ }

where

- $m \geq 0$, $n \geq 0$, and $m + n > 0$,

- the block following the word `where` is called the *body* of the positive search query, and is constituted simply by a conjunction of atoms (in SPARQL, the atoms in a conjunction are separated by "."),

- each α_i is an atom of the form `DataPropertyAssertion ( hasAssociatedValue` $?x$ v_i `)`,

- each β_i is an atom of the form `ObjectPropertyAssertion ( hasAssociatedObject` $?x$ $?y_i$ `)`,

- each γ_i is an atom of the form `ClassAssertion (` C $?y_i$ `)`.

The intuitive semantics of the above query in an interpretation $\mathcal{I}$ for $\mathcal{O}$ is as follows: the query asks for all samples that in $\mathcal{I}$ are related to the various input tokens $v_1, \ldots, v_m$ by means of the data property `hasAssociatedValue`, and are related by means of the object property `hasAssociatedObject` to at least one

individual (represented by $?y_i$) that is an instance of C_i, for each C_i in the set of the input classes $C_1, \ldots, C_n$.

Formally, if Q is a positive search query of the above form, and $\mathcal{I}$ is an interpretation for $\mathcal{O}$, the *extension of query* Q in $\mathcal{I}$, denoted by $Q^{\mathcal{I}}$, is defined as the set of individuals d in $\mathcal{O}$ such that:

- for each α_i in the body of Q, $(\mathtt{d}, v_i) \in \mathtt{hasAssociatedValue}^{\mathcal{I}}$,
- for each $\beta_i.\gamma_i$ in the body of Q, there is some $b \in C_i^{\mathcal{I}}$ such that $(\mathtt{d}, b) \in \mathtt{hasAssociatedObject}^{\mathcal{I}}$.

With the notion of extension of a query in an interpretation in place, we can now provide the definition of *certain answer* to a query Q with respect to an ontology $\mathcal{O}$. An individual $\mathtt{d}$ is a certain answer to Q with respect to the ontology $\mathcal{O}$, if $\mathtt{d} \in Q^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. Finally, the *set of certain answers* to Q with respect to $\mathcal{O}$, denoted as $cert_{\mathcal{O}}(Q)$, is simply the set constituted by each individual that is a certain answer to Q with respect to $\mathcal{O}$, i.e. $cert_{\mathcal{O}}(Q) = \bigcap_{\mathcal{I} \in \{\mathcal{J} | \mathcal{J} \models \mathcal{O}\}} Q^{\mathcal{I}}$.

We now turn to the class of basic search queries. Informally, a query of this class simply asks for the difference between two positive search queries. Given tokens $v_1, \ldots, v_m$ and classes $C_1, \ldots, C_n$ representing the features we want the desired samples to have, and given tokens $v'_1, \ldots, v'_m$ and classes $C'_1, \ldots, C'_n$ representing the features that we do not want the desired samples to have, the corresponding *basic search query* is defined as follows:

```
select ?x
where {{ α1.⋯.αm.β1.γ1⋯.βn.γn }
        MINUS
        { { α'1 } UNION ⋯ UNION { α'm' } UNION
          { β'1.γ'1 } UNION  ⋯ UNION { β'n'.γ'n' } }
       }
```

where

- $m \geq 0$, $n \geq 0$, $m + n > 0$, $m' \geq 0$, and $n' \geq 0$.
- As usual, the (complex) block following the word `where` is called the *body* of the query.
- The block following the reserved word `MINUS` is called the `MINUS`-block of the query.
- If $m' = 0$ and $n' = 0$, then the `MINUS`-block is missing.
- The `select` query with $?x$ as a distinguished variable and $\{ \alpha_1.\cdots.\alpha_m.\beta_1.\cdots.\beta_n \}$ as a body, i.e. the query

  `select` $?x$

  `where` $\{ \alpha_1.\cdots.\alpha_m.\beta_1.\gamma_1\cdots.\beta_n.\gamma_n \}$

  is called the *positive part of* Q and is denoted Q_p.
- The `select` query with $?x$ as a distinguished variable and the `MINUS`-block as body, i.e. the query

  `select` $?x$

```
where { { α'_1 } UNION ··· UNION { α'_m' } UNION
        { β'_1.γ'_1 } UNION  ··· UNION { β'_n'.γ'_n' } }
      }
```

is called the *negative part of* Q and is denoted Q_n. Note that the body of the negative part is constituted by the union of several atoms, where the semantics of union is the usual one in SPARQL (the result of the union query is the union of the results of the components).

The intuitive semantics of a basic search query Q is the obvious one: Q asks for all the samples that are among the certain answers to the positive part Q_p of Q, but are not among the certain answers to the negative part Q_n of Q. Thus, the formal semantics of basic search queries is immediate: given an ontology $\mathcal{O}$ and a basic search query Q, we have that $cert_{\mathcal{O}}(Q) = cert_{\mathcal{O}}(Q_p) \setminus cert_{\mathcal{O}}(Q_n)$.

Finally, we introduce the notion of search query. In our approach a *search query* is simply the union of a set of basic search queries. More precisely, if $Q_1, \ldots, Q_h$ are basic search queries, then the search query built on them is the query of the form:

```
select ?x
where { B_1 UNION ··· UNION B_h }
```

where $B_1, \ldots, B_h$ are the bodies of $Q_1, \ldots, Q_h$, respectively. The formal semantics of search queries is as follows: given an ontology $\mathcal{O}$ and a search query Q built on $Q_1, \ldots, Q_h$, we have that $cert_{\mathcal{O}}(Q) = cert_{\mathcal{O}}(Q_1) \cup \cdots \cup cert_{\mathcal{O}}(Q_h)$.

4.3 The Query Answering Algorithm

In this subsection, we illustrate our algorithm for computing the set of certain answers to a search query Q with respect to $\mathcal{O}$.

In principle, there are two approaches for computing the set of certain answers of Q with respect to $\mathcal{O}$. The first approach is based on query rewriting [8]: taking into account the intensional axioms of the ontology $\mathcal{O}$, the query is rewritten into a new query Q', which is evaluated over the extensional portion of the ontology, seen as a database. Unfortunately, this approach is not suited to our scenario, because, due to the size of $\mathcal{G}$, the size of the rewritten query may be prohibitive. The second approach is based on materialization [19]: the ontology $\mathcal{O}$ is transformed into another ontology $\mathcal{O}'$ obtained from $\mathcal{O}$ by adding suitable extensional axioms implied by $\mathcal{O}$ itself, and then the query Q is evaluated over the ontology $\mathcal{O}'$ seen as a database. The latter approach is the one that we follow in our work. As such, our method is similar to the one presented in [16], developed in the context of Web search. However, both our notion of search query and our notion of materialization of an ontology (hereafter called *completion*) are specialized to genomic metadata, and therefore differ considerably from the one presented in [16]. In particular, in our context, the completion of the ontology includes axioms that are not taken into account in [16], requiring different proofs of the correctness of the method.

Before delving into the details of the algorithm, we need three new notions, namely the notions of evaluation of a search query over an ontology, completion of an ontology, and canonical model of the completion of an ontology.

We start with the *evaluation of a search query Q over an ontology $\mathcal{O}$*, by first referring to positive search queries, and then generalizing to the whole class of search queries. If Q is a positive search query of the form:

`select` $?x$
`where` { $\alpha_1.\cdots.\alpha_m.\beta_1.\gamma_1\cdots.\beta_n.\gamma_n$ }

where each α_i has the form `DataPropertyAssertion ( hasAssociatedValue` $?x$ v_i `)`, each β_i has the form `ObjectPropertyAssertion ( hasAssociatedObject` $?x$ $?y_i$ `)`, and each γ_i has the form `ClassAssertion (` C $?y_i$ `)`, then the evaluation of Q over $\mathcal{O}$, denoted $eval_{\mathcal{O}}(Q)$, is the set of individuals `d` in $\mathcal{O}$ such that:

- for each α_i, the assertion `DataPropertyAssertion ( hasAssociatedValue d` v_i `)` is in $\mathcal{O}$, and
- for each β_i, the assertion `ClassAssertion ( ObjectSomeProperty ( hasAssociatedObject` C_i `) d )` is in $\mathcal{O}$.

If Q is a basic search query, then the evaluation of Q over $\mathcal{O}$ is defined as $eval_{\mathcal{O}}(Q_p) \setminus eval_{\mathcal{O}}(Q_n)$, where Q_p is the positive part of Q, and Q_n is the negative part of Q. Finally, if Q is a search query built on queries $Q_1, \ldots, Q_h$, then the evaluation of Q over $\mathcal{O}$ is defined as $eval_{\mathcal{O}}(Q_1) \cup \cdots \cup eval_{\mathcal{O}}(Q_h)$.

Note the difference between evaluating a query and computing its certain answers. When we evaluate a query over an ontology, we essentially treat the ontology assertions as a database, and we evaluate the query over such database. When we compute the certain answers to a query with respect to an ontology, we must consider all the models of the ontology, and return those tuples that are answers to the query in all such models.

We now define the notion of *completion of an ontology $\mathcal{O}$*. Roughly speaking, the completion $\mathcal{O}'$ of $\mathcal{O}$ is obtained from $\mathcal{O}$ by adding all assertions on the instances of `Sample` that are logically implied by $\mathcal{O}$. More precisely, we obtain $\mathcal{O}'$ by first letting all the axioms of $\mathcal{O}$ be also in $\mathcal{O}'$, and then by repeating the following rules until no more axiom can be added:

- For every pair (`a`,`b`) such that the axiom `ObjectPropertyAssertion ( consistsOf a b )` is in $\mathcal{O}'$, add the axiom `ClassAssertion ( Sample b )` and the axiom `ClassAssertion ( Experiment a )` to $\mathcal{O}'$, if they are not already in $\mathcal{O}'$;
- For every pair (`a`,`b`) such that the axiom `DataPropertyAssertion ( hasValueFor-A a v )` is in $\mathcal{O}'$, add the axiom `DataPropertyAssertion ( hasAssociatedValue a v )` to $\mathcal{O}'$, if it is not already in $\mathcal{O}'$;
- For every pair (`a`,`b`) such that `ObjectPropertyAssertion ( hasLinkTo-A a b )` is in $\mathcal{O}'$, add the axiom `ObjectPropertyAssertion ( hasAssociatedObject a b )` to $\mathcal{O}'$, if it is not already in $\mathcal{O}'$;
- For every assertion `ClassAssertion ( ObjectSomeProperty ( hasAssociatedObject` C_i `) a )` in $\mathcal{O}'$, and for every class C_j such that $\mathcal{O}' \models$ `SubsetOf (` C_i C_j `)`, add the assertion `ClassAssertion ( ObjectSomeProperty ( hasAssociatedObject` C_j `) a )` to $\mathcal{O}'$, if it is not already in $\mathcal{O}'$.

It is easy to see that $\mathcal{O}'$ is logically equivalent to $\mathcal{O}$, i.e. $\mathcal{O}'$ and $\mathcal{O}$ have the same models. Moreover, since $\mathcal{G}$ consists of `SubsetOf` axioms only, checking whether $\mathcal{O} \models$ `SubsetOf ( `C_i C_j `)` can be done in polynomial time, which implies that computing $\mathcal{O}'$ from $\mathcal{O}$ can be done in polynomial time, and that the size of $\mathcal{O}'$ is polynomial with respect to the size of $\mathcal{O}$.

We now turn to the notion of *canonical model* of a completion $\mathcal{O}'$ of an ontology $\mathcal{O}$, which is a particular interpretation for $\mathcal{O}'$, denoted $can(\mathcal{O}')$, defined as follows:

- The domain of $can(\mathcal{O}')$ contains one element v for each token v in $\mathcal{O}'$, one element d for each individual d that is an instance of the class `Experiment` in $\mathcal{O}'$, one element d for each individual d that is an instance of the class `Sample` in $\mathcal{O}'$, and one element $e_{\mathtt{d},\mathtt{A},C}$ for each triple $(d, \mathtt{A}, C)$ such that the assertion `ClassAssertion ( ObjectSomeProperty ( hasLinkToObject` C `) d )` is in $\mathcal{O}'$.
- The extensions of the various predicates in $can(\mathcal{O}')$ are as follows:
  - For every `d` in the domain of $can(\mathcal{O}')$, `d` $\in$ `Experiment`$^{can(\mathcal{O}')}$ if and only if the assertion `ClassAssertion ( Experiment d )` is in $\mathcal{O}'$.
  - For every `d` in the domain of $can(\mathcal{O}')$, `d` $\in$ `Sample`$^{can(\mathcal{O}')}$ if and only if the assertion `ClassAssertion ( Sample d )` is in $\mathcal{O}'$.
  - For every (`d`,`e`) in the domain of $can(\mathcal{O}')$, and for every `hasValueFor-A`, `(d,v)` $\in$ `hasValueFor-A`$^{can(\mathcal{O}')}$ if and only if the assertion `DataPropertyAssertion ( hasValueFor-A d v )` is in $\mathcal{O}'$.
  - For every (`d`,`e`) in the domain of $can(\mathcal{O}')$, `(d,e)` $\in$ `consistsOf`$^{can(\mathcal{O}')}$ if and only if the assertion `ObjectPropertyAssertion ( consistsOf d e )` is in $\mathcal{O}'$.
  - For every (`d`,$e_{\mathtt{d},\mathtt{A},C}$) in the domain of $can(\mathcal{O}')$, `(d,`$e_{\mathtt{d},\mathtt{A},C}$`)` $\in$ `consistsOf`$^{can(\mathcal{O}')}$ and $e_{\mathtt{d},\mathtt{A},C} \in C^{can(\mathcal{O}')}$ if and only if the assertion `ClassAssertion ( ObjectSomeProperty ( hasLinkToObject` C `) d )` is in $\mathcal{O}'$.

Note that all the axioms of $\mathcal{O}'$ are clearly satisfied by $can(\mathcal{O}')$, and therefore $can(\mathcal{O}')$ is a model of $\mathcal{O}'$. This also implies that $can(\mathcal{O}')$ is a model of $\mathcal{O}$. The following theorem immediately derives from the relationship between $\mathcal{O}'$ and $can(\mathcal{O}')$.

Theorem 4.1. *If $\mathcal{O}$ is an ontology, and $\mathcal{O}'$ is its completion, then for every search query Q, `d` $\in eval_{\mathcal{O}'}(Q)$ if and only if `d` $\in Q^{can(\mathcal{O}')}$.*

We can now present our method for computing the set of certain answers of a search query Q with respect to the ontology $\mathcal{O}$. The method is actually very simple, and is based on the algorithm that simply returns the set of answers $eval_{\mathcal{O}'}(Q)$. In other words, our algorithm can be described as follows: given $\mathcal{O}$, we compute the completion $\mathcal{O}'$ of $\mathcal{O}$, and then, when we have a search query Q, we compute the set of certain answers to Q with respect to $\mathcal{O}$ by simply evaluating such query over the ontology $\mathcal{O}'$. The next theorem shows that the algorithm is sound and complete, i.e. it correctly computes the set of certain answers to the query.

Theorem 4.2. *If $\mathcal{O}$ is an ontology and Q is a search query, then `d` $\in cert_{\mathcal{O}}(Q)$ if and only if `d` $\in eval_{\mathcal{O}'}(Q)$.*

Proof. It is immediate to observe that, if Q is built on the basic search queries $Q_1, \ldots, Q_h$, then $\mathtt{d} \in cert_{\mathcal{O}}(Q)$ if and only if $\mathtt{d} \in cert_{\mathcal{O}}(Q_1)$ or $\mathtt{d} \in cert_{\mathcal{O}}(Q_2)$, or ... or $\mathtt{d} \in cert_{\mathcal{O}}(Q_h)$, and $\mathtt{d} \in eval_{\mathcal{O}}(Q)$ if and only if $\mathtt{d} \in eval_{\mathcal{O}}(Q_1)$ or $\mathtt{d} \in eval_{\mathcal{O}}(Q_2)$, or, ..., $\mathtt{d} \in eval_{\mathcal{O}}(Q_h)$. Similarly, for each Q_i $(1 \leq i \leq h)$ having Q_p as positive part and Q_n as negative part (where Q_p and Q_n are positive search queries), $\mathtt{d} \in cert_{\mathcal{O}'}(Q_i)$ if and only if $\mathtt{d} \in cert_{\mathcal{O}}(Q_p)$ and $\mathtt{d} \notin cert_{\mathcal{O}}(Q_n)$. Similarly, $\mathtt{d} \in eval_{\mathcal{O}}(Q_i)$ if and only if $\mathtt{d} \in eval_{\mathcal{O}'}(Q_p)$ and $\mathtt{d} \notin eval_{\mathcal{O}'}(Q_n)$. Therefore, it is sufficient to prove the theorem for positive search queries. Thus, in what follows we assume that Q is a positive search query.

If-part We must show that $\mathtt{d} \in eval_{\mathcal{O}'}(Q)$ implies $\mathtt{d} \in cert_{\mathcal{O}}(Q)$. Suppose that $\mathtt{d} \in eval_{\mathcal{O}'}(Q)$. From theorem 4.1 we know that $\mathtt{d} \in Q^{can(\mathcal{O}')}$. We now prove that, for each model of $\mathcal{O}$, there is a homomorphism from $can(\mathcal{O}')$ to such model. Let M' be a model of $\mathcal{O}$, and let h be the function defined as follows:

- For every v that is a token in $\mathcal{O}'$, $h(v) = w$, where $w = v^{M'}$.
- For every $\mathtt{d}$ in $\mathtt{Experiment}^{can(\mathcal{O}')} \cup \mathtt{Sample}^{can(\mathcal{O}')}$, $h(\mathtt{d}) = e$ where $e = \mathtt{d}^{M'}$.
- For every $e_{\mathtt{d},\mathtt{A},C}$ in $can(\mathcal{O}')$, $h(e_{\mathtt{d},\mathtt{A},C})$ is the object in M' such that $(\mathtt{d}, e_{\mathtt{d},\mathtt{A},C}) \in \mathtt{hasLinkTo\text{-}A}^{M'}$.

To show that h is indeed a homomorphism, we have to show that, for every class C, $f \in C^{can(\mathcal{O}')}$ implies $f \in C^{M'}$, and for every binary relation R, $(f, g) \in R^{can(\mathcal{O}')}$ implies $(f, g) \in R^{M'}$. The only non-trivial case is the case involving objects $e_{\mathtt{d},\mathtt{A},C}$. Consider an object $e_{\mathtt{d},\mathtt{A},C}$ and observe that $(\mathtt{d}, e_{\mathtt{d},\mathtt{A},C}) \in \mathtt{hasLinkTo\text{-}A}^{can(\mathcal{O}')}$, and $e_{\mathtt{d},\mathtt{A},C} \in C^{can(\mathcal{O}')}$. By construction, $h(e_{\mathtt{d},\mathtt{A},C})$ is the object f in M' such that $(\mathtt{d}, f) \in \mathtt{hasLinkTo\text{-}A}^{M'}$. Also, since the axiom `ClassAssertion ( SomeObjectProperty hasLinkTo-A` C `) d )` is in $\mathcal{O}'$, and M' is a model of $\mathcal{O}'$, we have that $f \in C^{M'}$, which means that $h(e_{\mathtt{d},\mathtt{A},C}) \in C^{M'}$. It follows that h is indeed a homomorphism.

Now, by virtue of the property of homomorphism [9], we have that $\mathtt{d} \in Q^{can(\mathcal{O}')}$ implies that there is a homomorphism h' from the query Q to $can(\mathcal{O}')$. By composing h' and h, we obtain a new homomorphism $h' \circ h$ from Q to M', and this proves that $\mathtt{d} \in Q^{M'}$. We have thus shown that $\mathtt{d} \in eval_{\mathcal{O}'}(Q)$ implies $\mathtt{d} \in Q^{M'}$ for every model M' of $\mathcal{O}$, i.e. $\mathtt{d} \in cert_{\mathcal{O}}(Q)$.

Only-if-part We must show that $\mathtt{d} \notin eval_{\mathcal{O}'}(Q)$ implies $\mathtt{d} \notin cert_{\mathcal{O}}(Q)$. The proof is immediate. Indeed, it is easy to verify that $\mathtt{d} \notin eval_{\mathcal{O}'}(Q)$ implies $\mathtt{d} \notin Q^{can(\mathcal{O}')}$, and, since $can(\mathcal{O}')$ is a model of both $\mathcal{O}'$ and $\mathcal{O}$, this means that there is a model of $\mathcal{O}$ where $\mathtt{d}$ does not satisfies the query Q, which proves that $\mathtt{d} \notin cert_{\mathcal{O}}(Q)$. □

As for the computational complexity of the algorithm, the following observations hold:

- As we said before, computing $\mathcal{O}'$ from $\mathcal{O}$ can be done in polynomial time, and the size of $\mathcal{O}'$ is polynomial with respect to the size of $\mathcal{O}$.
- Since a search query Q can be seen as a first-order query, whose basic components are tree-shaped conjunctive queries, computing the set $eval_{\mathcal{O}'}(Q)$, given Q and $\mathcal{O}'$, can be done in `LOGSPACE` with respect to the size of $\mathcal{O}'$, which implies that computing the certain answers to search queries with respect to $\mathcal{O}$ can be done in polynomial time with respect to the size of both $\mathcal{O}$ and Q.

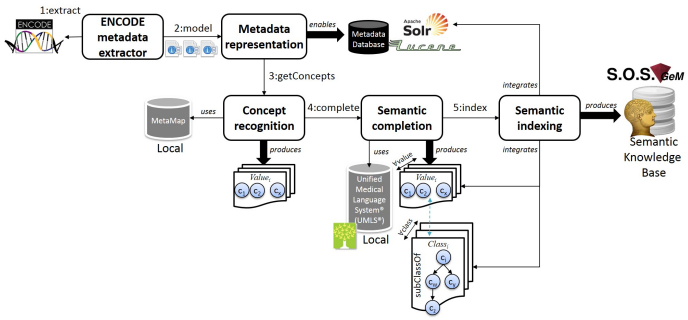

Figure 1: Workflow to create the S.O.S. GeM Semantic Knowledge Base of ENCODE metadata.

The next section illustrates an efficient implementation of the algorithm, based on the use of a semantic index.

5 Practical Deployment

In this section we give technical details of the implementation of our S.O.S. GeM solution. Section 5.1 first explains the process of constructing the S.O.S. GeM Semantic Knowledge Base (SKB), i.e. the practical realization of the completion $\mathcal{O}'$ of the ontology $\mathcal{O}$ discussed in Section 4. In practice, the SKB includes the concepts extracted from ENCODE metadata and their associated concepts inferred by using well-known biomedical ontologies. Then, we explain the search process for queries over ENCODE metadata (Section 5.2). Finally, we illustrate a Web-based interface (Section 5.3), dedicated to the non-expert users, which provides a simple ontology-based access and its obtained results.

5.1 Semantic Knowledge Base Creation

In building the SKB, we distinguish five different steps, which are performed sequentially in an offline process. They are depicted in Figure 1, namely *ENCODE metadata extraction*, *metadata representation*, *concept recognition*, *semantic completion computation* and the final *semantic indexing*.

5.1.1 ENCODE metadata extraction

This first step aims at producing the metadata set M by extracting the freely-available ENCODE metadata. Our extractor is a Python script that crawls the ENCODE site and collects all metadata for human and mouse ENCODE data.

The crawling process is performed sequentially: our metadata extractor explores each provided URL and navigates the subdirectories that group the experiments; for each of them, the extractor retrieves a list of files and their metadata, typically named in a *files.txt* text file. Then, metadata associated with each listed data file in FASTQ, BAM, bigWig, bigBed, BED, broadPeak, narrowPeak or GTF data format are extracted and completed with related metadata from the ENCODE controlled vocabulary[9]. The final metadata are saved in a single tab delimited text file, including all the metadata attribute-value

[9] `http://hgdownload.cse.ucsc.edu/goldenPath/encodeDCC/cv.ra`

pairs (e.g. "antibody_target CTCF") for each available ENCODE data file. We adopt this data schema as this is the standard format that can be directly used in the GMQL toolkit [20].

5.1.2 Metadata representation

As S.O.S. GeM proposes an incremental Knowledge Base creation, the first objective is to allow a syntactic search functionality of the extracted ENCODE metadata M. Intuitively, S.O.S. GeM builds a database hosting the extracted metadata M, associating each `Sample` with the set of extracted metadata attributes and values of an experiment data file. Formally speaking, the objective of this step is to satisfy the matching needs of the `hasValueFor-A` property, representing the syntactic information $t_S^{\mathtt{A}}(M)$ associated with each `Sample` S.

To do so, we make use of the well-known *Apache Lucene/Solr 5.0* framework[10], which proves to be a feasible and scalable solution to tokenize and index text documents. S.O.S. GeM represents the `Sample` metadata using three related Lucene/Solr elements: *documents*, *fields* and *nested documents*. As shown in an example in Figure 2, each `Sample` is a Lucene/Solr document, hereinafter called *ENCODE metadata document*. Each document is composed of a set of attribute-value pairs describing the `Sample` metadata, which are encoded as Lucene/Solr fields. Following the notation in Section 4, we rename these pairs as *attribute* and *tokens*. Lucene/Solr provides different "out of the box" tokenizers and filters to normalize the text values[11]. In our system, we initially stick to a basic word-delimiter tokenizer.

Given that attribute-tokens fields tend to be repeated across ENCODE samples, we make use of a recent Lucene/Solr technique, named *nested documents,* which simulates a relational scheme having several tables and foreign keys. Thus, we place each pair in a different sub-document, called *metadata pair document*, which can be referenced by different ENCODE `Sample` metadata. In this way, the different pairs are stored only once, while we maintain cross-references with their associated samples which can be navigated at query time. Finally, we label each `Sample` with a specific field, *dccAccession* which corresponds to an `Experiment` ID), in order to quickly link the `Sample` to the ENCODE `Experiment` it belongs. Although not discussed here, we also use other fields (such as *version*, *creation_date*, etc.) for maintenance purposes.

The result of this process is a metadata database resolving the `hasValueFor-A` property for each `Sample`. This will allow us to retrieve the set of ENCODE `Sample` metadata, together with the links to the corresponding ENCODE data, matching the user query.

5.1.3 Concept recognition

This step focuses on inspecting ENCODE metadata values and recognizing concepts that they may include from the biomedical ontologies in the Unified Medical Language System. Formally, this process analyses the text values and identifies the set of classes $c_S^{\mathtt{A}}(M)$ from the *global ontology* $\mathcal{G}$ that are in the ENCODE metadata M, being $\mathcal{G}$ the union of freely-available UMLS ontologies. Note that the terms "class" and "concept" are interchangeable in this paper.

[10] `http://lucene.apache.org/solr/`

[11] `https://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters/`

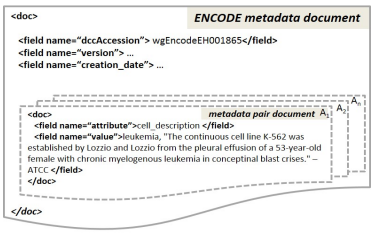

Figure 2: A first ENCODE metadata document.

S.O.S. GeM performs this task on the basis of *MetaMap* [3], a tool specifically designed to discover concepts in the UMLS Metathesaurus [7] that are referred to in a given text. MetaMap relies on a knowledge intensive approach based on symbolic, natural language processing and computational linguistic techniques; its efficiency has been largely shown for this particular task. S.O.S. GeM manages a local installation of MetaMap (version 2014) to boost the performance, by processing all the information requests through the provided MetaMap Java API (release 2014), and obtain the set of UMLS concepts found in each metadata value.

Here, it is worth recalling the integration effort provided by the UMLS Metathesaurus [7]. Current UMLS version (2014AB) includes 173 vocabularies from twenty-one languages, contributing with more than three million concepts (grouping almost twelve million atoms). The UMLS model is built under two main notions, namely *concept* and *atom*. A concept is an abstract unit of meaning, such as *heart* or *myocardial infarction*, that groups multiple (or at least one) vocabulary atoms, which are manually assessed to be synonyms in particular vocabularies. For instance, *myocardial infarction* groups "heart attack" in the Alcohol and Other Drug Thesaurus (AOD) and "cardiovascular stroke" in the Medical Subject Headings (MeSH). Furthermore, lexical variations of atoms (punctuation, language, etc.) are integrated and linked to their related concepts. Besides providing a curated repository of many biomedical vocabularies, UMLS includes relationships among concepts in different vocabularies, being synonyms the mappings of particular interest.

S.O.S. GeM parses each different metadata value and uses MetaMap to recognize all atoms/variations in the provided text, so that to provide as a result the UMLS concept grouping all the atom set. Since all atoms/variations under the same concept are synonyms, this allows us to minimize space and to manage a unique reference concept per recognized class. The recognized concepts per value are then stored in an *enriched metadata pair document*, as shown in Figure 3. In this way, we give a realization for the class `ObjectSomeProperty ( hasLinkTo-A C )` representing the fact that each `Sample` is associated with a set of objects that are instances of the various classes in $c_S^{\mathtt{A}}(M)$.

Finally, note that we use additional fields to store the position of each concept within each metadata value description. This allows highlighting to the user the text in the value description that represents a semantic concept found.

5.1.4 Semantic completion computation

After the phase of concept recognition, our SKB corresponds to the ontology $\mathcal{O}$, as described in Section 4. In the next step, we compute the completion $\mathcal{O}'$ of $\mathcal{O}$.

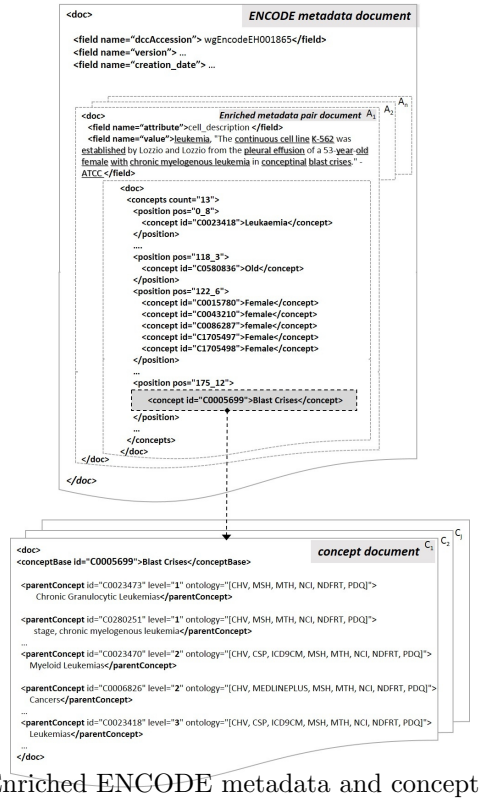

Figure 3: Enriched ENCODE metadata and concept documents.

To do so, we need to compute the semantic completion of the various concepts, by taking care of the class assertions `SubsetOf`.

For this purpose, S.O.S. GeM makes use of the UMLS Metathesaurus intra-source "distance 1" IS_A hierarchical relationships (immediate parents and children). Here "intra-source" denotes that these relations are not seen at the level of concepts, but at the level of atoms. That is, UMLS states the immediate `SubsetOf ( ` A_i A_j `)` relationship, where A_i and A_j are atoms in a given vocabulary. This means that one atom in a vocabulary can be navigated to get its more general concepts, or parents. For instance, in the AOD vocabulary, heart IS_A cardiovascular system, and mammal IS_A vertebrate, which formally state `SubsetOf (` *heart cardiovascularsystem* `)`, and `SubsetOf (` *mammal vertebrate* `)`.

In practice, S.O.S. GeM makes use of forward chaining to materialize all the UMLS concepts that can be inferred from the recognized metadata concepts in the enriched metadata pair documents. Thus, for every single recognized concept, we retrieve all its atoms and, for every atom, we navigate its hierarchies by iteratively retrieving the `SubsetOf` relationships up to the top level concept of the specific UMLS ontology. Note that, at every step, each retrieved atom A_j is again inspected, i.e. its associated UMLS concepts are retrieved and the process is repeated. To avoid processing each concept more than once, we maintain an ad-hoc data structure and, to make the computation efficient, we use a local installation of UMLS (ver. 2014AA on a MySQL database) to access both concept/atom data and hierarchies.

We store the semantic closure also by means of Apache Lucene/Solr documents (Figure 3). We consider each recognized UMLS concept as a document, named *concept document*, and we nest as many concept documents as different UMLS concepts can be inferred from it by following the `SubsetOf` relationships; we also store the number of inference steps (i.e. level) and the set of ontologies that have been used to reach each fact. For example, the fact `SubsetOf ( primate vertebrate )` is obtained in 2 steps, using AOD or NCI (National Cancer Institute) vocabularies, since in either vocabulary `SubsetOf ( primate mammal )` and `SubsetOf ( mammal vertebrate )`. An example is shown in Figure 3 (bottom).

5.1.5 Semantic indexing

The final step consists of building a semantic index for efficiently managing $\mathcal{O}'$. More precisely, we index the *enriched metadata pair documents*, i.e. the ENCODE metadata set M in which UMLS recognized concepts are annotated, and the *concept document*, stating the `SubsetOf`-based inferred concepts. Thanks to established Apache Lucene/Solr document schema, this process is straightforward: in practice, S.O.S. GeM loads and indexes the aforementioned documents from previous processes in our Lucene/Solr backend system which, all together, constitutes the S.O.S. GeM SKB.

It is worth mentioning that, while the SKB is built once in an offline process, S.O.S. GeM allows new ENCODE samples to be easily added to the SKB, by manually loading new data samples or rerunning the S.O.S. GeM ENCODE metadata extractor and monitoring changes between versions. The updating process runs efficiently as the closure computation, and final metadata indexing sticks only to the new recognized concepts; besides adding the enriched metadata pair documents for the new data samples, only the new concept documents need to be added to the system, without affecting the rest of the SKB.

We currently do not consider sample deletions, given that the set of ENCODE experiments is monotonically increasing. As for the UMLS concepts, the underlying database is typically updated just twice a year, so that S.O.S. GeM can synchronize at such given points and re-run the offline process (concept recognition, semantic completion and semantic indexing).

5.2 Search Process

The main objective of the search process is to enable a transparent semantic query facility to users. For instance, an ENCODE metadata document including "*53-year-old female*" should be retrieved when the user provides a search query Q "*mammal of 53 years*". Query resolution consists of *query parsing* and *query answering*.

Query parsing. S.O.S. GeM parses each query Q both syntactically and semantically. First, we use the S.O.S. GeM local MetaMap instance to recognize semantic concepts, i.e. all classes C_i contained in the provided $\beta_i.\gamma_i$ of the query. Note that we follow the same configuration as for the offline SKB creation, obtaining the general UMLS concept(s) referred in the text (e.g. "*mammal*" and "*years*"), together with their associated position in the text. We take this into account to extract the non-recognized parts of the query (e.g. "*of 53*" in

the previous example). Finally, we use a common list of stop words to filter such remaining text and extract interesting syntactic tokens α_i (e.g. "*53*").

Query answering. S.O.S. GeM strictly performs the *evaluation of the query* (Section 4.3) on the S.O.S. GeM SKB. Intuitively, S.O.S. GeM extracts (i) those samples that match the syntactic tokens, (ii) those matching the semantic concepts, and (iii) merges the results. In the first process, the syntactic tokens α_i of the query are matched against the tokens in the *enriched metadata pair documents*. The second process requires two complementary steps. First, we match the recognized classes C_i in the query against the nested *concept documents*, i.e. we try to align each C_i (e.g. "*mammal*") with the Y concepts that are part of a `SubsetOf ( X Y )` relationship, e.g. `SubsetOf (` $female\ mammal$ `)`. Then, we obtain the samples from the *enriched metadata pair documents* which are annotated with the more specific X concepts (e.g. *female*). The final merging process combines the results from both processes, filtering out samples satisfying the negative part of the query, Q_n, if present.

To date, the S.O.S. GeM parsing process does not automatically recognize negative parts in the query, so they have to be explicitly stated. By default, S.O.S. GeM considers each provided term as a positive *basic search query*, so that the *search query* is seen as the `UNION` of all basic search queries. This is fully compliant with our formalization in Section 4.2 and, in practice, states that results match at least one of the provided query terms. After results are shown, in a *query refinement step*, S.O.S. GeM allows the user to specify which of the query terms are part of the negative query (Q_n), and which terms are mandatory, thus grouping these latter ones in a conjunctive query inside a *basic search query.*

Additionally, S.O.S. GeM implements the ranking of the result set of samples. Given that the answer set could be large, we first group samples by their associated `Experiment`, making use of the aforementioned *dccAccession* property of each `Sample`, which unequivocally identifies the `Experiment`. Then, S.O.S. GeM applies a sequence of heuristics to provide a global order for this `Experiment` set: (i) the *number of query terms*, i.e. semantic concepts or syntactic tokens, *matched in the experiment metadata* (the higher the better), as not all terms might have been matched (in the default `UNION` interpretation, as stated in Section 4.2), (ii) the *maximum number of query terms matched in a single metadata value* (the higher the better), (iii) the *number of different metadata attributes A matched* (the higher the better), (iv) the *minimum number of inference steps* (i.e. `SubsetOf` relationship steps) between matching concepts (the smaller the better - the shorter the inference path, the closer is the meaning), (v) the *number of inferred facts from shorter to longer paths* (the higher the better - the more inferred concepts at shorter paths, the more accurate is the matching), and finally (vi) the *dccAccession* unique ID (taken lexicographically), in order to assure a global order.

5.3 Web Interface: Functionalities

S.O.S. GeM provides an intuitive Web interface[12], which acts as a high-level façade of the created SKB. As stated, our main aim is to allow users to query in a natural free text, so that, after the aforementioned search process (Section

[12] `http://www.bioinformatics.deib.polimi.it/SOSGeM/`

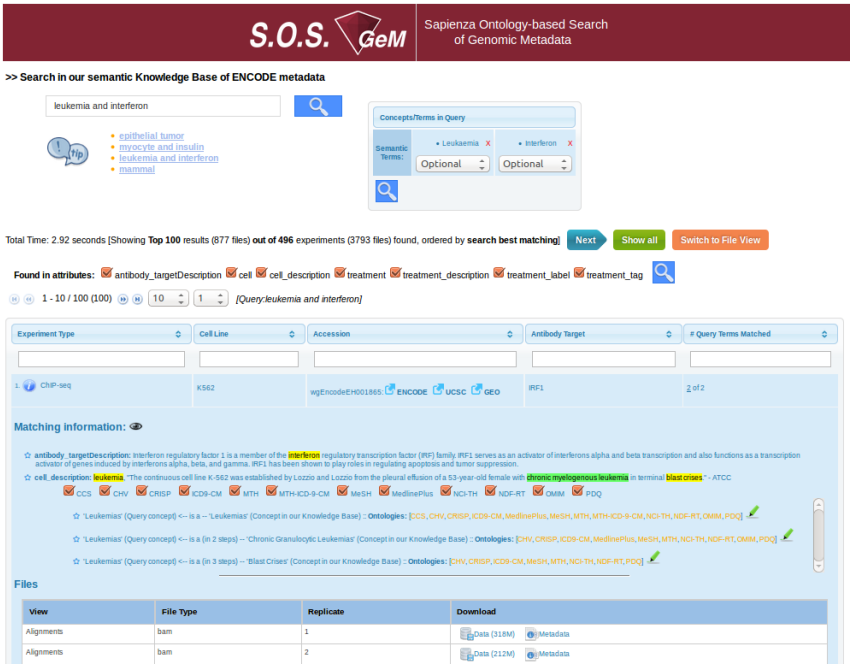

Figure 4: S.O.S. GeM Web interface showing matching details between concepts identified in the query and in the metadata of the found experiment.

5.2), the resulting ENCODE data are found. As shown in Figure 4, results are visualized grouped by `Experiment`, ordered and paginated by relevance (we paginate by the top-100 results using the heuristic in Section 5.2).

We also support the aforementioned *query refinement step* through the logical composition (with AND, OR or NOT operators) of the syntactic and semantic terms identified in the query text. Thus, the user can explicitly force (AND) or establish optional (OR) terms of the positive query (Q_p), and define the negative (NOT) part of the query (Q_n). Furthermore, we allow filtering the metadata attributes in which the query terms should be matched.

Upon user request, S.O.S. GeM shows the information that supports current results (see *matching information* in Figure 4); we (i) report the metadata values of the samples matching the query terms, (ii) highlight the matching concepts in the metadata values, and (iii) illustrate the concrete IS_A relationships (i.e. `SubsetOf` relations) that support the semantic match, together with the UMLS ontologies they belong to. Furthermore, we allow the user to select/deselect ontologies, thus showing/hiding their related facts.

Finally, for each resulting `Experiment`, we list the matched samples and allow downloading the extracted ENCODE metadata or the available ENCODE data. To enable a bulk download, S.O.S. GeM facilitates a "*Switch to File View*" option, where matched samples are not grouped by experiments, so that multiple data files can be selected for downloading with their metadata files in tab-delimited attribute-value text format. This supports the direct processing of found ENCODE genomic feature data with the novel GMQL toolkit for genomic knowledge extraction [20].

6 Evaluation

We first provide details on the SKB size and creation performance (Section 6.1), and next a qualitative analysis on the query performance (Section 6.2).

6.1 SKB Size and Computation Time

Table 1 shows the size of S.O.S. GeM SKB. We indexed 2,156,333 attribute-value pairs describing the metadata of 25,873 ENCODE samples of many different experiment types (5C, CAGE, ChIA-PET, ChIP-seq, Combined, DNA-PET, DNase-DGF, DNase-seq, Exon, FAIRE-seq, Methyl, Nucleosome, Proteogenomics, Repli-chip, Repli-seq, RIP, RIP-seq, RNA-chip, RNA-PET, RNA-seq); they correspond to 710 antibody targets for 69 different human or mouse cell tissues of affected, cancer or normal cells.

We recognized 4,000 different UMLS semantic concepts, which were completed with additional 12,330 UMLS concepts; their semantic closure produced 3.5 million concept combinations over 875 different IS_A concept paths.

The SKB computation figures are shown in Table 2; they report efficient performance in space and time in all the considered processes. Our extraction process compiles a corpus of ENCODE metadata that occupies 78 MB. As expected, the MetaMap-driven concept recognition excels in performance (51 minutes), while our document-based modelling of concepts produces a total of 578 MB. The semantic completion enlarges this size up to more than 1 GB; it runs efficiently (49 minutes), taking even less time than the concept recognition phase. Finally, the last semantic indexing performs in just 28 minutes by means of Apache Lucene/Solr. Overall, the S.O.S. GeM SKB is built in slightly more than two hours; thanks to the proposed nested document modelling, its space need is reduced up to 400 MB.

6.2 Query Performance

S.O.S. GeM supports any type of textual queries, which can be categorized as (i) single word representing a single concept, (ii) multiple words representing a single concept, (iii) multiple words representing multiple concepts; the latter category comprises the subtype of free text queries, which can include, besides relevant words, also several words to be disregarded (e.g. verbs, adverbs, adjectives, conjunctions, etc.). For each category, we defined some biologically meaningful example queries, listed in Table 3; we used them to evaluate S.O.S. GeM performance and compare it against that of the other systems available to search for ENCODE data, i.e. the ENCODE Project Portal (ENCODE-PP)[13] and the UCSC Genome Bioinformatics (UCSC-GB)[14] sites.

[13] `http://www.encodeproject.org/`

[14] `http://genome.ucsc.edu/ENCODE/search.html`

Table 1: Statistics of the S.O.S. GeM Semantic Knowledge Base

E	S	Metadata Pairs	C	CC	P
3,196	25,873	2,156,333	4,000	12,330	3.5 million

E: number of experiments; S: number of samples; C: number of classes; CC: number of classes after completion; P: number of SubClassOf axioms

Table 2: Computation of the S.O.S. GeM Semantic Knowledge Base

Initial Samples	**Concept Recognition**		**Semantic Expansion**		**Semantic Indexing**	
Size	*Time*	*Size*	*Time*	*Size*	*Time*	*Size*
78 MB	51 min	578 MB	49 min	1,137 MB	28 min	400 MB

Table 3: Types and examples of biologically meaningful queries and their results provided by S.O.S. GeM, ENCODE-PP and UCSC-GB

Query categories and their examples	**S.O.S. GeM**		**ENCODE-PP**		**UCSC-GB**	
	E	*S*	*E*	*S*	*E*	*S*
1) single word/ single concept						
Cachectin	20	159	-	-	-	-
TNF	20	159	4	42	10	40
estrogen	55	449	-	-	51	144
2) multiple words/ single concept						
proto-oncogene	58	500	-	-	26	74
peptide hormone	73	465	-	-	-	-
white blood cell	428	3,627	-	-	-	-
3) multiple words/ multiple concepts						
epithelial tumor	797	6,558	-	-	1	4
H1 proto-oncogene	6	42	-	-	-	-
leukemia interferon	30	239	-	-	27	78
myocyte insulin	27	114	-	-	-	-
peptide hormone oncogene	3	21	-	-	-	-
3b) free text						
I would like to search for CTCF in human leukemia	13	113	-	-	-	-

E: number of experiments; S: number of samples

S.O.S. GeM provides more results than the other systems (Table 3); while the other systems support just keyword-based queries, we also answer free text queries, leveraging extraction and semantic matching of query concepts. Expert evaluation of S.O.S. GeM results for the considered example queries confirmed their correctness in almost all cases.

6.2.1 Single word / single concept queries

An example is the query "*Cachectin*". It is a synonym for *Tumor necrosis factor* (*TNF*), a cell signaling protein involved in systemic inflammation which is one of the cytokines (i.e. proteins) that make up the acute phase reaction. Searching for it within the ENCODE metadata using S.O.S. GeM gives as a result 20 experiments (159 sample data files), all correctly selected based on the value "*TNF-alpha*" of their *treatment* metadata attribute, which is recognized expressing the same concept of the word *Cachectin*; no result is found for this search using ENCODE-PP or UCSC-GB. When searching for "*TNF*" S.O.S. GeM finds the same results as of the "*Cachectin*" search, whereas ENCODE-PP and UCSC-GB retrieve 4 experiments (42 samples) and 10 experiments (40 samples), respectively. This example shows the advantage offered by the semantic support in answering very specific queries. Also other examples of this category (e.g. "*estrogen*") are better answered by S.O.S. GeM.

6.2.2 Multiple word / single concept queries

The first example is the query "*proto-oncogene*". A proto-oncogene is a normal gene that can become an oncogene due to mutations or increased expression. S.O.S. GeM finds 500 sample data files of 58 different experiments with various antibody targets whose encoding gene is known to be a proto-oncogene, such as Myc or c-Fos. For the same search, no results are found with ENCODE-PP, whereas UCSC-GB gives only 74 sample data files of 26 different experiments.

The second example is the query "*peptide hormone*". ENCODE-PP and UCSC-GB do not retrieve results for this search, although UCSC-GB finds 116 and 345 samples for the individual search for *peptide* and *hormone*, respectively; conversely, S.O.S. GeM finds 73 experiments (465 samples), which are retrieved mainly because they regard a treatment with a peptide hormone (e.g. insulin, or activin) or a target protein related to some peptide hormone (e.g. involved in the responses of a peptide hormone). An expert reviewing these search results, through inspection of their metadata, determined that 65 of them are correctly retrieved; the remaining 8 are incorrectly retrieved due to the MetaMap match of the "SRF" acronym both to the "Serum Response Factor" (which is the right match in this context, but is not a peptide) and to the Somatotropin-Releasing Hormone (which is a synonym of the "Growth Hormone Releasing Hormone" concept, and is a peptide).

The third example is the query "*white blood cell*". White blood cells are a broad category of cells present in the blood which are part of the immune system. While any biologist would be able to search for all types of cells in this category one by one, the list is long and it would be easy to forget some of them or to misspell any of their names. This search in S.O.S. GeM provides 3,627 samples of 428 experiments, which are all deemed as correct by the reviewing expert, whereas no results are found with ENCODE-PP or UCSC-GB. The UMLS ontologies involved in this example are usually 7 (CHV, CRISP, FMA, LOINC, MeSH, MTH, NCI-TH). In the majority of cases the query match is found in one or two steps. In some cases the match occurs also in the *antibody_targetDescription* metadata attribute, and involves a higher number of semantic steps (3 or 4).

6.2.3 Multiple word / multiple concept queries

The first example is the query "*H1 proto-oncogene*", which adds to the term "*proto-oncogene*" the term "*H1*"; the latter one is found as a syntactic term, abbreviation of the H1-hESC name of an embryonic stem cell line largely analyzed in ENCODE. In S.O.S. GeM, 42 samples of 6 experiments are found as a result of setting both terms as mandatory; all found experiments regard the H1-hESC cell-line and proto-oncogene related antibody targets, including c-Myc, FOSL1, MAFK and BCL11A. Both ENCODE-PP and UCSC-GB find many experiments related to H1, as expected, but none using the more specific query "*H1 proto-oncogene*".

The second example is the query "*leukemia interferon*". S.O.S. GeM answers with 3,793 samples of 496 experiments, related to at least one of the two concepts. Setting both terms as mandatory retrieves 239 samples of 30 experiments. Expert evaluation of the latter ones confirmed their correctness; all such experiments regard the K562 cell line, a chronic myelogenous leukemia cell

line, treated with interferon alpha or gamma for different time length; additionally, the four top ranked experiments involve the interferon regulatory factor 1 (IRF1) as antibody target. For the same query ENCODE-PP provides no results, whereas UCSC-GB finds 78 samples of 27 experiments.

The third example is the query "*myocyte insulin*". S.O.S. GeM answers with 980 samples of 154 experiments, including 114 samples of 27 experiments associated with both insulin and muscle cell concepts (since a myocyte is a muscle cell). Expert evaluation of these results proved that all of them correctly matched the query terms. Conversely, UCSC-GB and ENCODE-PP provide no results for the same query. However, UCSC-GB finds 11 samples of 4 experiments for the "*insulin*" query and 46 samples of 13 experiments for the "*myocyte*" query, whereas ENCODE-PP finds 38 experiments for the "*myocyte*" query, but no results for the "*insulin*" query.

6.2.4 Free text queries

An example of such queries, which only S.O.S. GeM can answer, is "*I would like to search for CTCF in human leukemia*". In this case, S.O.S. GeM first identifies that the query includes the "*CTCF*", "*human*" and "*leukemia*" semantic terms; then, it finds 22,331 samples of 2,805 experiments whose metadata are semantically related to at least one of these concepts. When all three identified terms are set as mandatory, S.O.S. GeM finds 113 samples of 13 experiments. Expert inspection of these latter results showed that all of them correctly answer the query with appropriate ranking and can be useful to elucidate possible effects (if any) of the CTCF transcription factor in human leukemia.

7 Conclusion and Future Work

S.O.S. GeM introduces a semantic, ontology-based approach to support the search and retrieval of ENCODE data of interest; we described in depth our solution from theoretical and practical standpoints, proving that our approach is sound and complete, and we provided a thorough evaluation. We plan to further develop and enhance S.O.S. GeM; in particular, we plan to extend it to support the search and retrieval of publicly accessible TCGA data. The TCGA repository regards gene expressions and DNA mutations of several different cancer types from many patients (at the time of writing, 34 cancer types of more than 11,000 patients); they very well integrate with and complement the functional genomic and epigenomic data provided by ENCODE. TCGA data are also associated with metadata values of several clinical parameters characterizing the patient and biological sample from where they were obtained; thus, the S.O.S. GeM approach immediately applies to them. The effective search, retrieval and join evaluation of both ENCODE and TCGA data, using the GMQL toolkit, has a strong potential of boosting biomedical knowledge discovery.

References

[1] 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[2] E. Antezana, M. Kuiper, and V. Mironov. Biological Knowledge Management: the Emerging Role of the Semantic Web technologies. *Brief. Bioinform.*, 10(4):392–407, Jul 2009.

[3] A. R. Aronson and F. M. Lang. An Overview of MetaMap: Historical perspective and recent advances. *J. Am. Med. Inform. Assoc.*, 17(3):229–236, 2010.

[4] M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, et al. Gene Ontology: tool for the unification of biology. the Gene Ontology Consortium,. *Nat. Genet.*, 25(1):25–29, 2000.

[5] SK Bechhofer, Robert D Stevens, and Phillip W Lord. Gohse: Ontology driven linking of biology resources. *Web Semant.*, 4(3):155–163, 2006.

[6] Jagdev Bhogal, Andy Macfarlane, and Peter Smith. A review of ontology based query expansion. *Inf. Process. Manag.*, 43(4):866–886, 2007.

[7] O. Bodenreider. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Nucleic Acids Res.*, 32(Database issue):D267–270, Jan 2004.

[8] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning And Efficient Query Answering in Description Logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[9] Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *J. Comput. Syst. Sci.*, 25(1):99–128, 1982.

[10] H. Chen, T. Yu, and J. Y. Chen. Semantic Web Meets Integrative Biology: A Survey. *Brief. Bioinform.*, 14(1):109–125, Jan 2013.

[11] Manhong Dai, Nigam H Shah, Wei Xuan, Mark A Musen, Stanley J Watson, Brian D Athey, Fan Meng, et al. An efficient solution for mapping free text to ontology terms. *AMIA Summit on Translational Bioinformatics*, 21, 2008.

[12] Marie-Dominique Devignes, Malika Smaïl, Emmanuel Bresso, Adrien Coulet, Chedy Raïssi, and Amedeo Napoli. Knowledge discovery from biological big data: Scalability issues. *Int. J. Metadata Semant. Ontol.*, 5(3):184–193, 2010.

[13] Manuel Carlos Díaz-Galiano, Maite Teresa Martín-Valdivia, and LA Ureña-López. Query expansion with a medical ontology to improve a multimodal information retrieval system. *Comput. Biol. Med.*, 39(4):396–403, 2009.

[14] Andreas Doms and Michael Schroeder. Gopubmed: Exploring pubmed with the gene ontology. *Nucleic Acids Res.*, 33(suppl 2):W783–W786, 2005.

[15] ENCODE Project Consortium et al. An integrated encyclopedia of dna elements in the human henome. *Nature*, 489(7414):57–74, 2012.

[16] Bettina Fazzinga, Giorgio Gianforme, Georg Gottlob, and Thomas Lukasiewicz. Semantic web search based on ontologica conjunctive queries. *Web Semant.*, 9(4):453–473, 2011.

[17] Erika Check Hayden. Technology: the $1,000 genome. *Nature*, 507(7492):294–5, 2014.

[18] Robert Hoehndorf, Michel Dumontier, and Georgios V Gkoutos. Evaluation of research in biomedical ontologies. *Brief. Bioinform.*, 2012.

[19] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Knowledge Representation and Reasoning, 12th Int. Conf. on*, pages 247–257, 2010.

[20] M Masseroli, P Pinoli, F Venco, A Kaitoua, V Jalili, F Palluzzi, H Muller, and S Ceri. GenoMetric Query Language: A novel approach to large-scale genomic data management. *Bioinformatics*, 28(7):691–693, Feb 2015.

[21] Boris Motik, B Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (Second Edition), 2012. W3C Recommendation 11 December 2012.

[22] Hans-Michael Müller, Eimear E Kenny, and Paul W Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biol.*, 2(11):e309, 2004.

[23] M. C. Schatz, B. Langmead, and S. L. Salzberg. Cloud Computing and the DNA Data Race. *Nat. Biotechnol.*, 28(7):691–693, Jul 2010.

[24] Nigam H Shah, Nipun Bhatia, Clement Jonquet, Daniel Rubin, Annie P Chiang, and Mark A Musen. Comparison of concept recognizers for building the open biomedical annotator. *BMC bioinformatics*, 10(Suppl 9):S14, 2009.

[25] Cormac Sheridan. Illumina claims $1,000 genome win. *Nat. Biotechnol.*, 32(2):115, 2014.

[26] Maria Taboada, Hadriana Rodríguez, Diego Martínez, María Pardo, and María Jesús Sobrido. Automated semantic annotation of rare disease cases: a case study. *Database*, 2014:bau045, 2014.

[27] Ornuma Thesprasith and Chuleerat Jaruskulchai. Query expansion using medical subject headings terms in the biomedical documents. In *Intelligent Information and Database Systems*, pages 93–102. Springer, 2014.

[28] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nat. Genet.*, 45(10):1113–1120, 2013.

[29] Weijian Xuan, Manhong Dai, Barbara Mirel, Jean Song, Brian Athey, Stanley J Watson, and Fan Meng. Open biomedical ontology-based medline exploration. *BMC bioinformatics*, 10(Suppl 5):S6, 2009.

[30] W. Zhu, Xuheng Xu, Xiaohua Hu, I.-Y. Song, and R.B. Allen. Using umls-based re-weighting terms as a query expansion strategy. In *Granular Computing, 2006 IEEE Int. Conf. on*, pages 217–222, May 2006.