UNIVERSITY OF CAGLIARI

CLOUD-BASED SOLUTIONS SUPPORTING DATA AND KNOWLEDGE INTEGRATION IN BIOINFORMATICS

by
Gabriele Milia

A thesis submitted for the degree of
Philosophiæ Doctor

PhD in Computer Science
XXVII Cycle

Supervisor: Prof. Nicoletta Dessì

PhD Coordinator: Prof. Giovanni Michele Pinna

INF/01

2013 - 2014

Abstract

In recent years, computer advances have changed the way the science progresses and have boosted studies in silico; as a result, the concept of “scientific research” in bioinformatics has quickly changed shifting from the idea of a local laboratory activity towards Web applications and databases provided over the network as services. Thus, biologists have become among the largest beneficiaries of the information technologies, reaching and surpassing the traditional ICT users who operate in the field of so-called "hard science" (i.e., physics, chemistry, and mathematics). Nevertheless, this evolution has to deal with several aspects (including data deluge, data integration, and scientific collaboration, just to cite a few) and presents new challenges related to the proposal of innovative approaches in the wide scenario of emergent ICT solutions.

This thesis aims at facing these challenges in the context of three case studies, being each case study devoted to cope with a specific open issue by proposing proper solutions in line with recent advances in computer science.

The first case study focuses on the task of unearthing and integrating information from different web resources, each having its own organization, terminology and data formats in order to provide users with flexible environment for accessing the above resources and smartly exploring their content. The study explores the potential of cloud paradigm as an enabling technology to severely curtail issues associated with scalability and performance of applications devoted to support the above task. Specifically, it presents Biocloud Search EnGene (BSE), a cloud-based application which allows for searching and integrating biological information made available by public large-scale genomic repositories. BSE is publicly available at: *http://biocloud-unica.appspot.com/*.

The second case study addresses scientific collaboration on the Web with special focus on building a semantic network, where team members, adequately supported by easy access to biomedical ontologies, define and enrich network nodes with annotations derived from available ontologies. The study presents a cloud-based application called Collaborative Workspaces in Biomedicine (COWB) which deals with supporting users in the construction of the semantic network by organizing, retrieving and creating connections between contents of different types. Public and private workspaces provide an accessible representation of the collective knowledge that is incrementally expanded. COWB is publicly available at: *http://cowb-unica.appspot.com/*.

Finally, the third case study concerns the knowledge extraction from very large datasets. The study investigates the performance of random forests in classifying microarray data. In particular, the study faces the problem of reducing the contribution of trees whose nodes are populated by non-informative features. Experiments are presented and results are then analyzed in order to draw guidelines about how reducing the above contribution.

With respect to the previously mentioned challenges, this thesis sets out to give two contributions summarized as follows. First, the potential of cloud technologies has been evaluated for developing applications that support the access to bioinformatics resources and the collaboration by improving awareness of user's contributions and fostering users interaction. Second, the positive impact of the decision support offered by random forests has been demonstrated in order to tackle effectively the curse of dimensionality.

Contents

List of Figures

List of Tables

Acronym

AJAX	Asynchronous JavaScript and XML
AUC	Area Under the ROC curve
API	Application Programming Interface
BSE	Biocloud Search EnGene
COWB	COllaborative Workspaces in Biomedicine
DaaS	Data as a Service
GAE	Google App Engine
IaaS	Infrastructure as a Service
IG	Information Gain
NoSQL	Not Only Structured Query Language (i.e., non-relational)
PaaS	Platform as a Service
RDF	Resource Description Framework
REST	REpresentational State Transfer
SaaS	Software as a Service
URI	Uniform Resource Identifier

Chapter 1

Introduction

In recent years, the advent of high-throughput methodologies (~ omics) has favoured the exponential growth of heterogeneous data in biology. Every year, the cost of generating, acquiring and spreading data continues to decrease, so biologists and computer scientists have to cope with processing an increasingly amount of data. Therefore, biology is more and more a data-intensive science and has to effectively exploit these growing available pieces of information hosted in vast numbers of independent and heterogeneous resources spread all over the Web [Pas08]. In this scenario, biologists expect more and more capabilities from Web applications and databases. On the other hand, the development of specialized tools involves several standards, technologies, and frameworks which are often complex, expensive to develop and maintain, and require accurate planning and management [SK10]. In particular, the development of Web applications and databases in bioinformatics arises several challenges and issues, such as data access, visualization, and representation (i.e., standards). Solutions, which aim at addressing these problems related to databases and Web applications for integrating data and knowledge, have to tackle some challenging aspects, including the data deluge, data integration, and scientific collaboration.

In general, data are currently considered the fourth paradigm in Science [HGP12, HTT09, HTT11] being the previous paradigms the empirical science, theoretical science and computational science. Nowadays, biomedical data are typical of the category of "Big Data" [RSK+11]; that is, data which are characterized by the so-called 5 Vs: volume, velocity, variety, value, and veracity [DdLM14, DgdL+13]. In addition, biomedical data rely on a wide range of data sources and are easily shared and replicated. On the other hand, they present significant reuse opportunities which accelerate investigations already under way by taking advantage of past efforts in science [Lyn08]. Moreover, they present

many attractive opportunities as regards the knowledge discovery from data (KDD) that requires more complex and sophisticated tools in order to transform data in meaningful knowledge [HKP11]. Therefore, advances in data mining technology are necessary to improve the quality of data and the analysis results. In spite of past research, it is still a challenge for the scientific community to individuate algorithms to effectively integrate, clean, and represent data.

Nowadays, biological research is becoming more and more interdisciplinary, and searching for information often requires the integration of data with multiple levels of granularities and relates data which pertain to different disciplines. Usually, searches are carried out over resources distributed over the Web which use different standards to represent data. The Web 2.0 [Mur07] has dramatically changed the way of managing Web contents and promoted the use of collaborative systems such as wikis, blogs and social networks. This shift from a static web to a dynamic one has pointed up two crucial bioinformatics problems. First, the rate of growth of user-generated contents requires methods to effectively exploit these data which are characterized by rapid-obsolescence [FNS11]. Second, the success of the Linked Data paradigm [BHL09] is boosting the data-oriented vision in bioinformatics. Many projects try to promote this vision by releasing, sharing, and linking data by means of URIs, HTTP and RDF in order to replace the conventional resource-oriented model and get something closer to a global repository which complies with the Semantic Web paradigm.

Data management is only part of the open issues in bioinformatics. Currently, researchers need to share their data with the whole scientific community throughout the world in order to yield new useful information and hypotheses derived from processing existing public datasets. Web 2.0 technologies allow researchers to interact with their colleagues, in that transcending traditional data integration technologies [SK10]. Unlike Web 1.0 that was static, these technologies make the Web highly collaborative and allow users to create large network of academic peers. Nonetheless, data originate problems related to their semantic heterogeneity, integrity and formats (i.e., interoperability) which prevent researchers to exchange information and use different tools [Mar13]. However, the main problem in bioinformatics remains the lack of integration between different resources such as ontologies, databases or individual resources.

In order to face the above-mentioned issues and challenges, recent advances in computer science continue to significantly influence the development of databases and Web applications in bioinformatics.

Specifically, the service-oriented computing (SOC) [SQV+14] has paved the way for thinking biomedical resources in terms of computational infrastructures by posing services as primary functional elements for data integration, delivery and usage. Many bionformatics institutes, such as Swiss Insitute of Bioinformatics (SIB) [SIB14] and European Bioinformatics Institute (EBI) [EBI14], make available their scientific databases and software tools (i.e., resources) as Web services through exposing Web application programming interfaces (API).

Web 2.0 exploits Web services to get interoperability between data resources and software, and programming techniques such as AJAX (Asyncronous JavaScript and XML) to support dynamic user interaction on the Web [SK10]. The exponential growth of biomedical information over the Internet dramatically increases the benefits of using Web 2.0 applications which rely on services available on the Web in form of APIs.

Cloud computing [LFZ+09, BYV+09] and non-relational databases (i.e., NoSQL databases) [HHL+11] are two significant components of the Web 2.0 era and seem to offer interesting features in order to meet some crucial requirements of large-scale applications. Cloud computing is a set of technologies which allows service providers to delivery services over a network in a *pay-as-you-go* manner. Although applications that exploit cloud computing are still at a preliminary stage in bioinformatics, their number is rapidly increasing [SOH14, CQY+13]. However, most of computing applications deal with processing and analysing large datasets [WKF+10, HGV+09, GTB+08, AGT+08], while few work has been done to explore different architectural solutions [QEG+10, WN11, FPG+11]. As regards NoSQL databases, they represent the next generation of databases that mostly addresses some crucial points in data management, including horizontal scalability, flexibility, and weak consistency. They are getting more and more attention since they are schema-free; that is, they have a flexible structure [CZ14]. This feature is extremely attractive for domains as biology because biological data are very heterogeneous.

The remainder of this dissertation is organized as follows. Chapter 2 provides a background on some relevant technologies in bioinformatics. In Chapter 3, I present a cloud-based application (BSE) which provides a comprehensive environment for capturing, integrating, and searching genetic and genomic data coming from resources distributed over the Web. COWB, a cloud-based application which relies on an extensible framework for handling collaborative biomedical knowledge, is discussed in Chapter 4. Chapter 5 presents an experimental analysis about the effect of a filtering process on the predictive performance of a random forest classifier and its critical parameters. Finally, Chapter 6 presents concluding remarks.

Chapter 2

Emerging Technologies and Trends in Bioinformatics

This chapter presents the state of the art about some emerging technologies and trends that support scientific advances in bioinformatics. First, cloud computing is introduced with focus on its essential characteristics, service models, and deployment models. Next, advantages of cloud computing in bioinformatics are highlighted. Second, this chapter discusses the basic features of the next generation databases also known as NoSQL databases and outlines their potential in the context of bioinformatics. Finally, advantages and drawbacks of ontologies are outlined, especially in life science domains.

2.1 Cloud Computing

According to National Institute of Standards and Technology (NIST) [NIST15], cloud computing can be defined as follows [MG11]:

> *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services), that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

This definition points out that these resources are quickly provided in a *pay-as-you-go* fashion (i.e., consumers pay only for what they use) as it happens for public utilities.

Moreover, they are subjected to Quality of Service (QoS) parameters that are defined in Service Level Agreements (SLA).

Hence, cloud computing has the following characteristics:

- *On-demand self-service.* A customer can unilaterally get computing resources (e.g., storage) without interacting with a service provider.
- *Broad network access.* Resources are made available over the network using standard mechanisms in order to allow both thin or thick devices (e.g., smartphone or Personal Computer) to access these capabilities.
- *Resource pooling.* A service provider uses a multi-tenant model for serving multiple consumers; thus, it guarantees that customers and their data are protected from each other. Both physical and virtual resources are provided and released as user computing requirements change. Actually, nobody has the knowledge over the exact location of the computing resources but it may be possible to specify location (e.g., country or data center).
- *Rapid elasticity.* Computational resources are elastically assigned and released in order to scale rapidly. From the consumer point of view, capabilities appear unlimited and always available.
- *Measured Service.* Service provider monitors and reports the capability usage to the customer in a transparent way.

Since cloud computing virtualizes the resources, users consider the offered resources as something unlimited. Consequently, they do not have to plan in advance the amount of resources they need. Moreover, users avoid up-front investments and rent resources according to their real needs.

The cloud services are broadly divided into four abstract layers:

- *Infrastructure as a Service (IaaS).* This layer provides the physical assets (e.g., servers) as a metered service using a *pay-as-you-go* fashion. Resources can be accessed by consumers without knowing where they are physically hosted. At IaaS layer, Amazon EC2 [Ama14] is a famous case in the industry.

- *Platform as a Service (Paas).* This layer provides programming languages, libraries, APIs, environments and tools (i.e., a platform) in order to enable developers to build applications onto the cloud infrastructure. User does not handle the underlying cloud infrastructure but manages deployed applications and settings of the app. Examples of PaaS are Google App Engine [AE14], Microsoft Azure [MA14] and Heroku [Her14].

- *Software as a Service (SaaS).* It delivers software services on-line; therefore, SaaS eliminates the need for local installation and both software maintenance and updates are easier than on-premises software. As a successful example, Google Drive [GD15] allows users to create and share documents online for accessing them "anywhere, anytime".

- *Data as a Service (DaaS).* It supplies dynamic data access on-demand. Data are up-to-date and accessible by heterogeneous thin or thick client platforms that are connected over the Internet [DGG+12]. Current example of DaaS is Amazon Web Service (AWS) [AWS14], one unit within *Amazon.com* which hosts and makes available a variety of public datasets, including 1000 Genomes Project [GPA14], Ensembl Annotated Human Genome Data [EAH14] and Human Microbiome Project [HMP14] (available on AWS at *http://aws.amazon.com/public-data-sets/)*.

These services are usually represented as a stack because each service is built upon the previous layer (Figure 2.1).

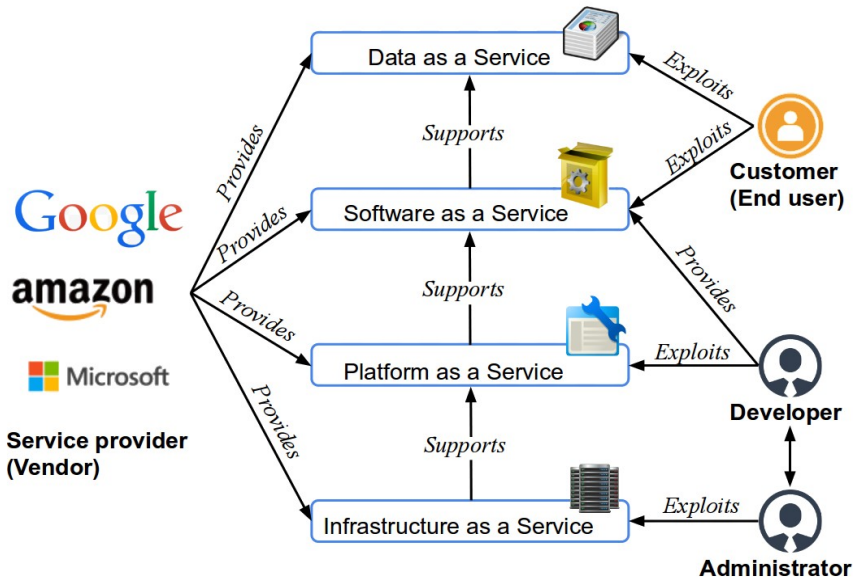

Figure 2.1. Cloud computing: service models

Depending on their deployment, cloud infrastructures can broadly classified as follows:

- *Public cloud*. Data centers (i.e., hardware and software) are made available in a *pay-as-you-go* fashion to the public [AFG+09].
- *Private cloud*. This service deployment model indicates internal data centers of a single business, government, or academic organization which are not provided to the public. Note that a private cloud may be managed by the organization itself, a third party, or some combination of them.
- *Community cloud*. The cloud infrastructure is made available to a specific community of users with similar requirements. A community cloud is a generalization of a private cloud where consumers may belong to different organizations which share concerns such as mission or policy.
- *Hybrid cloud*. The cloud infrastructure consists of two or more mixed models (e.g., private and public). These distinct data centers exploit standardized or proprietary technology in order to work together and get data/application portability. Hybrid cloud aims at addressing the limitations of public and private clouds offering more flexibility. The hybrid model's shortcoming is that determining the best trade-off between public and private cloud component is a difficult task.
- *Virtual Private Cloud*. A Virtual Private Cloud (VPC) builds an infrastructure on a Public Cloud by exploiting Virtual Private Network technology [ZCB10].

For small organizations, institutes and laboratories, cloud computing offers to the following advantages:

- Lower entry cost to use compute-intensive resources available only to the largest organizations.
- Cost-effective *pay-per-use* utility computing model.
- High scalability dynamically adjusted according to user demand.
- Immediate access to hardware resources with no upfront capital investment.

A lot of work on bioinformatics [HGV+09, KJD+12, TOG+10, ZGL+11] makes use of IaaS for deploying data-intensive applications which address issues related to large-scale data processing. Indeed, recent research [DG10] has shown the utility of MapReduce [DG04], a programming model for processing vast amounts of data in a parallel and distributed manner. By partitioning data on different servers, this programming model allows for the parallel, distributed processing of large datasets across clusters of computer. Hadoop [Had15], an open-source implementation of MapReduce, has effectively seen a widespread adoption in bioinformatics [ODS13]. However, MapReduce is especially suitable for cloud platforms because they make available scalable clusters of nodes; in fact, there are many services which make use of MapReduce on top of their cloud platforms. For example, Amazon provides Amazon Elastic MapReduce (Amazon EMR) [EMR15], a Web service that exploits Hadoop and allows users to quickly and cost-effectively process large datasets by distributing data across a resizable cluster of Amazon EC2 [Ama14] instances. Currently, Amazon EMR is employing in a variety of applications, including data warehousing, machine learning, and bioinformatics. Further examples are App Engine MapReduce [GMP15] and HDInsight [HDI15], respectively an open source library built on top of the PaaS provided by Google (i.e., Google App Engine [AE14]) and a framework for the Microsoft Azure [MA14] cloud implementation of Hadoop.

A less explored layer of cloud services is PaaS. As far as I know, few work has been done to investigate the potential of bioinformatics applications developed at this layer. Undergoing studies include two recent projects developed at the Institute for Systems Biology [ISB14]: Regulome Explorer [RE14] and Pubcrawl [PC14], respectively deployed on App Engine and Amazon Web Services. Specifically, Regulome Explorer supports the exploration of datasets which give information about common gene disruptions across different cancers, whereas Pubcrawl is an application that combines literature based semantic distances with protein domain interactions to dynamically create network topologies of terms. However, creating, deploying and managing a scalable Web application in the cloud at PaaS layer seems interesting, particularly in bioinformatics. Indeed, PaaS, which relies on a cloud infrastructure, facilitates the development, maintenance, and update of an application by making available several programming languages, standard protocols, libraries, relational and non-relational databases, and frameworks. In addition, PaaS avoids to think in terms of virtual machines [Sur12].

Finally, PaaS seems to address the challenge of facilitating the development of applications as a distributed, scalable and widely-accessible service in the Web [SLB+11].

As regards bioinformatics, the interest in investigating PaaS layer is also motivated by the fact that this kind of platform often adopts non-relational databases whose models meet the need of organizing loosely structured and heterogeneous data, as it often happens in bioinformatics. To better explain this interest, the next section details the basic features of this kind of databases.

2.2 NoSQL Databases

Featured by a well-structured and rigid model, relational database systems have serious problems in coping with large datasets that change quickly and are mostly unstructured and connected. Nowadays, relational DMBS are still very popular in the database market; indeed, they are a mature technology which represent a lot of investments by vendors, users, and developers in terms of money and technical know-how. With the advent of distributed architectures and cloud computing, alternative models for storing and managing data have been proposed in order to address challenges mainly originated from Web 2.0 requirements. These solutions do not replace relational databases, but they are adopted for specialized projects such as those that are distributed, that involve large datasets, or that need scalability [Lea10].

These solutions are generally referred to as NoSQL databases. The term “NoSQL” is an acronym which stands for “Not Only SQL” and denotes the “Next Generation Databases mostly addressing some points, such as being non-relational, distributed, and horizontally scalable” [Nos14].

NoSQL solutions have been developed as in-house custom solutions by companies such as Google, Amazon, LinkedIn in order to solve their real specific problems which arose from three broad issues: unprecedented transaction volumes, expectations of low-latency access to large amounts of data and availability in an unreliable environment [Bur11]. Projects/products include BigTable (provided by Google) [CDG+06], Amazon DynamoDB [DDB15], LikedIn Voldemort (LinkedIn) [Vol15], and Apache Cassandra (Facebook) [AC15].

The main feature of NoSQL systems is that they generally do not guarantee ACID (atomicity, consistency, isolation and durability) properties. Since these properties are not essential in some applications [ABR14, HWC+14], NoSQL databases relax these properties in favor of the following BASE properties:

- *Basically Available (BA).* The system works most of the time.
- *Soft-state (S)*. The system is not write-consistent; consequently, different replicas can be not mutually consistent at some point.
- *Eventual consistency (E).* The system eventually reaches consistency; in other words, it does not guarantee the consistency at a specific time.

Differently from the ACID properties, the co-existence of the above properties must comply with the CAP theorem [Bre00] which states that every networked shared-data system can retain at most two of the following properties:

- *Consistency (C).* It indicates that after an update operation of some writers all readers see the updates of the shared-data system; that is, all client always have the same up-to-date copy of the data.
- *Availability (A).* It means that a system is designed and implemented in order to keep working in case of problems. For example, a system must cope with crash or hardware/software update without stopping its tasks. Therefore, each client can always read and write.
- *Partition (P).* This property is referred to the ability of system to operate in the presence of physical network partitions.

The CAP theorem results in a shift from the strong consistency guaranteed by ACID properties to a weak consistency (e.g., eventual consistency) provided by BASE approach (see Figure 2.2).

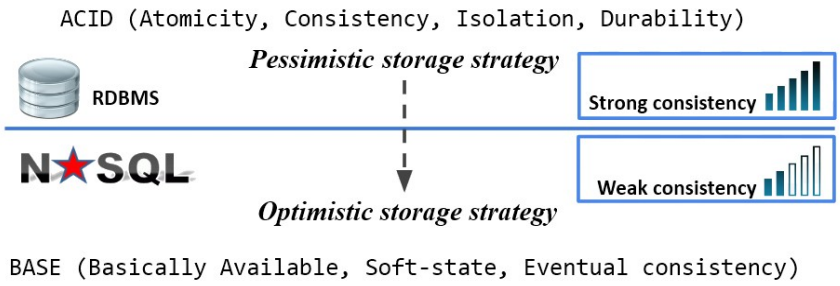

Figure 2.2. From ACID properties to BASE approach.

Figure 2.3 shows the well-known triangle which is usually used to explain the CAP theorem. It details the properties (consistency, availability and partition) preserved by some commercial databases such as MySQL and PostgreSQL (CA), and NoSQL solutions (CP or PA).

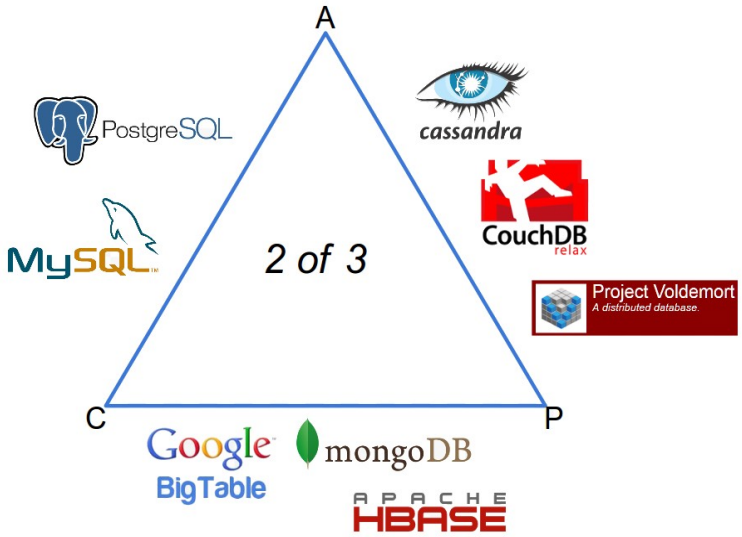

Figure 2.3. Visual guide to CAP theorem

NoSQL databases can broadly classified into the following categories:

- *Wide Column Store / Column Families.* Column family stores have been designed after Google's BigTable [CDG+06]. They consist of a sparsely populated table whose rows can contain arbitrary columns [RWE13]. Therefore, column stores are

column-centric; that is, they handle data by column instead of by row as it happens in traditional relational databases. Row keys are used in order to build column indexes. This category includes several famous databases such as BigTable (via Google App Engine [AE14]), Apache Cassandra [AC15], and Apache Hbase [AHB15].

- *Document store or Document-oriented storage.* Document databases handle semi-structured data; specifically, they store and retrieve documents which organize data according to several standard formats, including JSON, XML, and YAML. Each document has an ID and is considered as an independent entity. In general, document databases rely on indexes which allow for retrieving documents based on their attributes [RWE13]. Document-oriented storage is the most popular paradigm for managing hierarchically structured documents. Examples of document store are MongoDB [Mon15], CouchDB [Cou15], and RavenDB [Rav15].

- *Key Value / Tuple Store.* Key-value stores can be considered cousins of the document store family. As the name suggests, they store data as a collection of key-value pairs (i.e., a dictionary) [GRR14]; therefore, key-value stores are large, distributed hashmaps which allow for retrieving values by means of keys [RWE13]. Amazon DynamoDB [DDB15] and the Oracle NoSQL database [Ora15] are two popular databases of this category.

- *Graph databases.* Graph databases rely on a graph for representing data and making available CRUD functions (Create, Read, Update, and Delete). Specifically, there are several types of graph data model, including property graph, hypergraphs, and triples [RWE13]. Therefore, this kind of database is well suited to store data and their relationships (e.g., data from social networks). It is worth noticing that graph databases use graph algorithms for optimising traversal performance. A good example of this category is Neo4J [Neo15].

- *Multi-Model Databases.* Multi-Model databases have schemas with several different features. For example, OrientDB [Ori15] can store documents like any other document database, but also handles relationships such as a graph database. Therefore, it aims at exploiting both the advantages of a distributed graph database engine and the flexibility of a document database. As a result, OrientDB maintains constant the traversing speed regardless of the database size.

However, bioinformaticians often need to cope with the following types of data [AD11] simultaneously:

- *Structured data.* They have a well-defined schema which allows for querying data by means of a structured query language (e.g., SQL). The schema is defined according to a data model; for example, the relational model. The drawback of this kind of data arises when it is necessary to change the schema in order to meet new requirements.

- *Unstructured data.* This type of data does not have a schema which organizes information; therefore, unstructured data are not arranged in accordance with a data model. Examples of unstructured data are pictures, digital audio and video, text documents, and emails.

- *Semi-structured data.* They refer to data that have themselves some pieces of information to convey their schema (e.g., XML tags or RDF statements).

- *Partially structured data.* These data comprise both free text and information which complies with a schema; in other words, information is partly formatted according to metadata encoded as a database schema and partly in the form of free text [KP00].

In general, organizations that must cope with storing and processing large collection of unstructured data are more and more turning to NoSQL databases [Lea10]; in fact, relational DBMSs do not fit well in facing the following issues:

- Data variety imposes new requirements to data storage solutions and database design which should be quite flexible in order to cope with a increasingly number of data sources with diverse data (e.g., spreadsheets, Web sources, XML, traditional DBMSs) [SR13]. RDBMSs are recognized as optimal solution for handling structured data; in fact, they gear data to multiple tables and provide excellent data integrity [MPR+12]. On the contrary, they offer little support for unstructured data, semi-structured, and partially structured data.

- RDBMSs usually allow for scaling up (i.e., vertical scaling), but do not allow for scaling out (i.e., horizontal scaling) [HWC+14]. Therefore, they have problems with architectures in which nodes or servers can be added in order to increase the capacity of an application environment because relational table joins across server/nodes are complicated and expensive [MPR+12].

- RDBMSs are not successful in processing large datasets quickly because they comply ACID properties in order to guarantee strong consistency; that is, all clients must see the same data at the same time. The main drawback of strong consistency is that it requires time-consuming tasks. Finally, note that ACID guarantees are often too pessimistic in many problems (e.g., social networks).

Unlike RDBMSs, NoSQL databases do not have to comply with rigid schemas and scale out easier than traditional RDBMSs. These features are particularly attractive in bioinformatics to build specialized tools [MPR+12]. Some bioinformatics tools have already been developed by exploiting NoSQL databases such as Hbase [OMN10], CouchDB [MPR+12], App Engine Datastore [WN11]. Despite their relative immaturity, NoSQL databases, such as graph databases, are considered ready for bioinformatics [HJ13]. For example, Bio4j [PTPT+13] exploits Neo4j [Neo15] in order to provide a powerful framework for protein which integrates most data available in UniProt KB (SwissProt + Trembl), Gene Ontology (GO), UniRef (50, 90, 100), RefSeq, NCBI taxonomy, and Expasy Enzyme DBs.

Recent achievements of some bioinformatics institutes, such as EMBL-EBI, seem to confirm that NoSQL is a frontier research in bioinformatics. Specifically, EMBL-EBI databases group acquired MongoDB [Mon15] skills and started to plan a central infrastructure in order to make available this document database as a service [EMB13].

2.3 Semantic Web technologies: Ontologies

In recent years, the Semantic Web technologies [WS14] have played a key role in many scientific disciplines (e.g., bioinformatics) which rely heavily on computational infrastructures for managing large-scale data [LKN+13]. This preeminent role mainly

arises from their suitability in supporting knowledge management [OLe98]. Perhaps the most important benefit of Semantic Web technologies is the use of ontologies for defining the concepts and relationships (also known as "terms") used to formally describe and represent an area of interest [WO15].

The role of ontologies on the Semantic Web is to support machines in understanding the meaning (i.e., the semantics) of information on the World Wide Web; in other words, ontologies must enable machines to reason about the semantics of terms automatically. Therefore, machine-readable ontologies play a key role in Semantic Web development [XMR+11] since they are the fundamental building blocks which support inference techniques on the Semantic Web [WO15]. Furthermore, since ontologies organise data in a form that allows for integrating pieces of information, they facilitate data integration when ambiguities can arise from terms used in different datasets. Combining the knowledge from various datasets enables a whole range of applications such as decision support tools, advanced web search engine, and Web applications that support scientific collaboration.

As observed by recent research [DW12], ontologies are not only powerful but also complex resources. Moreover, the success of ontologies in integrating data has led to an uncontrolled proliferation of ontologies [SAR+07]. Nowadays, several ontologies are available, but merging information from different ontologies still requires knowledge about ontologies or at least about how to reuse conceptual descriptions provided by others and how to publish new ones. However, the relevance of ontologies has widely acknowledged in bioinformatics. Some projects, such as OBO foundry [SAR+07], aim at overcoming the before-mentioned problems by establishing a set of design principles that can boost interoperability of ontologies. In fact, these principles want to ensure a gradual improvement of quality and formal precision in ontologies. Nevertheless, designing, creating and publishing a set of interoperable ontologies in the biomedical domain is still a challenge.

Currently, a lot of ontologies are written in RDF [RDF15] and RDF-based semantic languages such as OWL [OWL15]. Unfortunately, to become familiar with RDF syntax requires a steep learning curve. For this reason, different repositories provide services with

the aim of facilitating the exploitation of public ontologies. For example, BioPortal [WNS+11], which is a repository of biomedical ontologies, currently contains 105 OBO, 31 UMLS, and 273 OWL ontologies. It allows for browse public biomedical ontologies and provides many services for working with them (e.g., an annotator that extracts annotations for biomedical text with concepts from the biomedical ontologies). Furthermore, BioPortal provides REST API to access ontologies programmatically. Another interesting service is OBA [DW12] that provides a connector for embedded usage of ontologies in applications in order to help developers in building an applications based on information available in ontologies without being familiar with ontologies or query languages to process them.

Biomedical ontologies are gaining a fundamental role in enabling researchers and their tools the exchange of interoperable data with minimum ambiguity. Moreover, ontologies can meet the increasing requirements of data and knowledge integration in the biomedical domain. Finally, by exploiting ontologies, data generated by biomedical research could form a single, coherent, expandable, and manageable comprehensive knowledge which would represent a great added value for biomedical data.

2.4 Concluding Remarks

Nowadays, traditional centralized computing platforms not only are expensive but also risk to become increasingly inadequate to meet the application requirements. According to a recent poll on biomedical research facilities [CJL+13], a significant number of research institutes are experiencing the capacity limits of their computing facilities. Collecting and configuring tools and resources for certain research purposes is a non-trivial job, even for expert developers and technicians. Furthermore, functionalities and capabilities expected from bioinformatics Web application and databases are becoming more and more sophisticated; thus, flexibility, scalability and interoperability have to be placed at the core of expected features for bioinformatics tools. Therefore, bioinformaticians call for a new stack of technologies that exploits interoperable technologies and highly scalable computing models, frameworks, and platforms. As outlined in this chapter, this new stack can rely on three emerging technologies. The first technology is cloud computing that

offers a feasible platform with demonstrated elasticity and parallelism capacity for managing large datasets [CSP13 ,CQY+13]. The second technology refers to NoSQL databases which provide modern web-scale databases for fast and efficient queries on huge amount of data. Finally, the last technology is represented by ontologies that enable both scientists and tools to convey with minimum ambiguity, especially in challenging domains such as bioinformatics.

Chapter 3

Data Integration on the Cloud: a Case Study

Currently, there is a large number of databases and gene annotation resources which are greatly relevant in the genetics and genomics communities since each one presents a particular aspect regarding available gene notations. For instance, the 2014 Nucleic Acids Research (NAR) Database Issue [SRG13] expounded 181 articles of molecular biology databases, sixty-eight percent of which provided updates on the databases previously presented in NAR and other journals.

For investigating a concept, scientists usually have to deal with a set of untied data sources, but at the same time, they want to search information in the whole collection of data without having a deep knowledge about information sources [MHF06]. In acquiring information from web resources, they merely retrieve a small amount of information about a particular concept; as a result, they must filter huge pieces of data available in different web resources to get the information of interest. Since biology spreads over multiple domains, scientists have to search information stored in several databases and web sites for each concept they investigate. Each system presents a different user interface, terminology and data formats; therefore, the quality of the search depends on user ability to exploit these distributed resources over the Web. Hence, the discovery of specialized information can be difficult for three reasons. First, researchers have to remember how to navigate each specific web site and this task can be time-consuming and daunting. Second, different systems implement the same functionalities in different ways and often deal with overlapping data. Finally, a lot of identifiers are used to pinpoint the same concept.

In order to search effectively in biomedical databases and cope with the growing number of resources available worldwide, it is necessary to answer three basic questions.

- How to integrate structured, semi-structured, partially structured, and unstructured available data with diverse and sparse schemas?

- How to retrieve meaningful information in an easy and efficient way?

- How to implement a searching infrastructure which has to scale, hence change, in order to meet new requirements stemming from the growth of its searching domain?

Computational solutions, which range from database to data warehouse, poorly adapt to face the above questions for the following reasons:

- Many resources are large in size, dynamic, and physically distributed; as a result, it is necessary to implement mechanisms that can efficiently extract the relevant data from disparate sources on demand.

- Searching strategies must be devised for obtaining the necessary information within constraints imposed by the different owners of the data source in order to comply with the various policies.

- Being heterogeneous in structure and content, resources represent data according to their own schema which defines its concepts and relationships among concepts.

- Searching happens in different contexts and from different user perspectives; therefore, it is necessary to implement mechanisms for mining context-dependent information.

This first case study present an application which aims at proposing a feasible solution to above questions and focuses on genomics, a key area of biology which places stress on trying to solve the problem of gathering and processing large amounts of biological data. Specifically, this case study presents BioCloud Search EnGene (henceforth BSE) [DPM+13], a comprehensive searching environment which facilitates the versatile integration of existing genetic and genomic information from multiple heterogeneous resources. The key idea is to conceive BSE as a cloud-based application which essentially rents its capacity from a cloud computing platform and relies on a new operational framework in which genetic information and computing technologies are reshaping each other. Like popular online gene portals, BSE adopts a gene-centric approach: researchers can find information by means of a simple query interface that accepts standard gene identification as keywords. Moreover, by using advanced searching and tools, users are enabled to explore many resources via high quality, interoperable services offered in a "neutral" space. As it happens for web search engines which are designed to look for information on the World Wide Web, several services act as specialists which extract data available in many databases or open directories and return real-time information. They are a mean of organizing and integrating information from different web sources and making them manageable and satisfactory for the user.

BSE is publicly available at *http://biocloud-unica.appspot.com*.

3.1 Architectural aspects

BSE grounds on the dataspace paradigm [FHM05, HFM06], a new scenario for handling information relevant to a particular organization (e.g. enterprises, government agencies, universities), regardless of its format and location in a *data co-existence* perspective [MHF06]. Dataspace paradigm indicates a set of principles which aim at enhancing traditional technology [Jef08]. Dataspaces set out to provide an alternative to classical data integration methods by reducing up-front costs and integrating data in an incremental manner. A dataspace consists of a set of participants (i.e., individual data sources) and the relationships among them [HFM06]. Specifically, a dataspace is an abstraction of a database that does not require structured data and has a base "off-the-shelf" set of

functionalities over all data sources. The key idea is to enhance the quality of data integration and the semantic meaning without defining a schema for all the data sources in advance [DH07, HMR+08, AD11].

In contrast to the traditional data integration approaches, a dataspace integrates data according to a very loosely structured data model (hence, *data co-existence*) which allows the system to manage heterogeneous data coming from a diverse set of sources. The core of a dataspace is a catalogue which contains both information about participants and their relationships. In addition, the catalogue provides operations to extend and update itself in an incremental fashion. Advanced DBMS-like operations, queries and mappings are made available over time by different components. Unified views over participants are provided following the *pay-as-you-go* principle that is currently getting more and more attention on the Web [JFH08, HBP+11]. In fact, this principle is especially attractive in order to get data integration on the Web because *pay-as-you-go* systems attempt to provide services on a set of heterogeneous data with a limited up-front effort [DHZ12].

According to the dataspace paradigm, BSE undertakes the responsibility of coordinating and organizing the search across a dataspace whose participants are a set of distributed resources. In addition, BSE indexes these resources by a catalogue which represents the core of the dataspace. Data integration expects no data transfer to any central repository, except for the data stored in BSE catalogue which is initially built and gradually updated. In some way, the catalogue has the same role of the table of facts in a data warehouse system where the dimension table are distributed across several web resources. However, compared to a data warehouse schema, the catalogue presents the following differences:

- It avoids the definition of an a priori schema.
- It stores pieces of information about dataspace participants instead of relational tables.
- Besides storing and indexing participants, the catalogue provides mechanisms for creating new relationships among participants.

From a logical point of view, the catalogue is a multi-level index that indicates how data from various web resources is captured and tied together. Physically, it is implemented by an object-oriented, NoSQL database which stores gene annotations, acquires and combines data from external resources which participate in the dataspace.

The current version of BSE implements a dataspace which counts 34 participants. According to their role in supplying data these participants are divided in three categories:

- *Local participants.* Resources from which some useful content is captured and permanently stored into the catalogue.

- *Service-based participants.* Resources whose content is captured at running time by specific BSE services in a *pay-as-you-go* fashion according to user demands.

- *External participants.* Resources whose web links are dynamically built and activated on demand.

The catalogue organizes objects in classes, each corresponding to one local participant. In details, a local participant is mapped in a repository of objects associated with the catalogue and relationships among participants are expressed by means of key-value pairs. Table 3.1 shows the list of local participants and the corresponding catalogue content.

The schema-free structure of the catalogue makes possible to implement new ways for querying and extracting information based on the notion of context. Specifically, a context is a logical structure that supports queries about common points of interest that users share in surfing dataspace participants. For example, if a user wants to search information about genes associated with a specific disorder, he/she refers to the context '*Human Mendelian Genetic disorder*'. Contexts are the only way to query the catalogue. Each context presents a gene-centric view where the users can easily identify the relevant resources and browse the content of the resources to which the context relates. Contexts hide the complexity of underlying dataspace by exploiting BSE services which capture and present the information of interest.

Dataset	**Catalogue content**
Entrez Gene *homo sapiens gene info* [MOP+11, BHK+14]	Main annotations about human genes
Entrez Gene Relations *human gene relations* [MOP+11, BHK+14]	Gene to gene relationships
Manually Annotated Targets and Drugs Online Resource (M.A.T.A.D.O.R) [GKD+08]	Gene drug relationships
Entrez gene ID to pathways [COW+11]	Human genes pathways according to Reactome
Entrez gene ID to Mendelian Phenotype [McK98]	Human Mendelian Phenotypes and their gene associations
Entrez gene ID to RefSeq [PTB+12]	Cumulative set of transcripts and proteins
Human Ageing Genome Resource [TCB+13]	Ageing-Related humans genes
Wellcome Trust Sanger Institute - Cancer genomics annotations [YSP+13]	Cancer Drug sensitivity Annotated genes

Table 3.1. Local participants and corresponding catalogue content.

From a technical point of view, contexts identify specific perspectives on dataspace participants that are kept in the catalogue. These perspectives resemble views in relational databases. However, being the catalogue implemented by a NoSQL database, they do not result from joining relational tables, but from relationships expressed by key-value pairs. In addition, contexts take very little space to be stored since the catalogue contains only the definition of contexts without a copy of all the data that the context relates to.

The current version of BSE implements the following contexts:

- *Query by gene.* It allows users to search information about a specific gene by means of standard identifiers.

- *Query by Human Mendelian Genetic disorder*. This context permits users to extract a list of genes by specifying the name of a certain phenotype associated with a genetic disorder with Mendelian transmission character.

- *Query by pathway*. It allows users to extract a list of human genes annotated in a given biological pathway. Specifically, a pathway is a set of chemical reactions related to one or more processes within a cell. It results in expression products whose knowledge is very important in the study of biological phenomena.

- *Bulk queries*. This context allows users to extract a list of human genes which abide by the following criteria: gene type, chromosome, ageing related annotation, chemotherapeutic sensitivity related to annotated genes according to their mutational status.

- *Query by drug*. It permits users to search information about a specific drug. Therefore, it shifts the query focus from a purely genetic perspective to a context which deals with the relationships between pharmacologically active molecules and the human genome expression products.

3.2 BSE Functionalities

BSE provides a simple graphical user interface (GUI) that takes account of user experience and usability in presenting information. In particular, BSE GUI exploits a set of user interface interactions, effects and widgets, including accordions, tabs and tooltips in order to address the problem of presenting a lot of information in a scarce space. Specifically, an accordion is a widget which organizes a web page for showing a lot of contents in a limited amount of space. The contents are broken into logical sections which are visualized in collapsible panels. By clicking headers, each panel is expanded/collapsed in order to show/hide a specific section. Tabs are used to further arrange pieces of information inside the accordions; indeed, a tab is a single content area that can break information into multiple panels. In some way, an accordion splits information vertically, whereas tab

breaks data horizontally. Finally, a tooltip is a widget for providing users with some text messages in order to help them during the navigation of BSE.

BSE provides four accordions:

- The *basic accordion* allows users to search information in the different contexts that are made available within the application;

- The *gene accordion* is displayed whenever the user click on a gene returned by a query and arranges a lot of detailed information about that gene.

- The *drug accordion* is visualized whenever the user click on a drug returned by a query and allows him/her to get information about the drug properties and interactions with proteins.

- The *tool accordion* provides, as its name suggests, two tools. The first one supports users in looking for overlapping information from a list of genes. The second one allows users to search articles by specifying a keyword.

In what follows I present each accordion above mentioned.

Figure 3.1 shows the basic accordion of BSE. The panel *Search gene by IDs*, which corresponds to the context *'Query by gene'*, is expanded and presents three mutually exclusive text fields where users can type a single identifier in order to search a specific gene. Within this search context the user can search for genes by Entrez ID, or UNIPROT accession. The Alias gene identification is supported too. BSE provides an autocomplete for each text field in order to help users. For example, in Figure 1 the user is typing the keyword “tp53” as standard gene identifier while BSE dynamically provides predictive suggestions by expanding the keyword “tp53” in a sliding list of its synonyms and variants. The user chooses the appropriate identifier from the list, submits his/her query and gets information depicted in Figure 3.2.

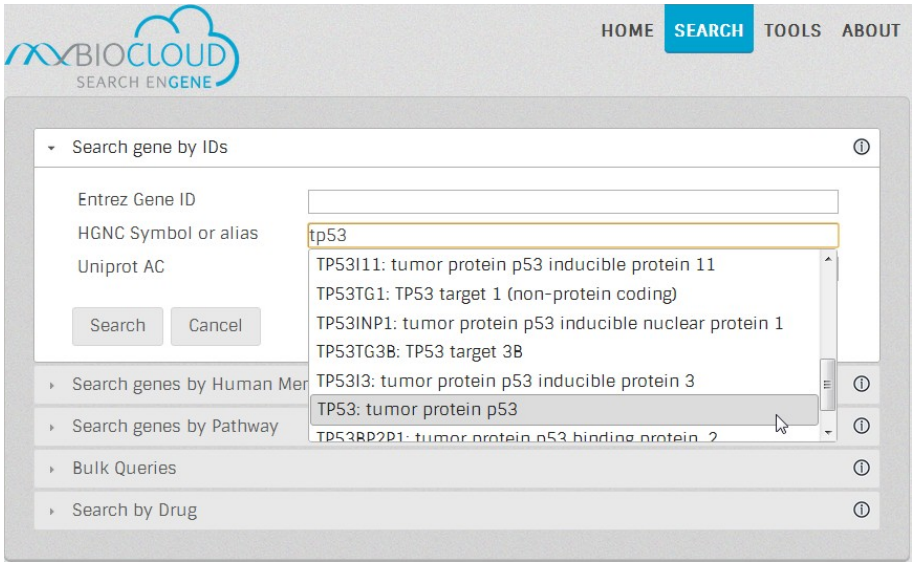

Figure 3.1. The basic accordion of BSE.

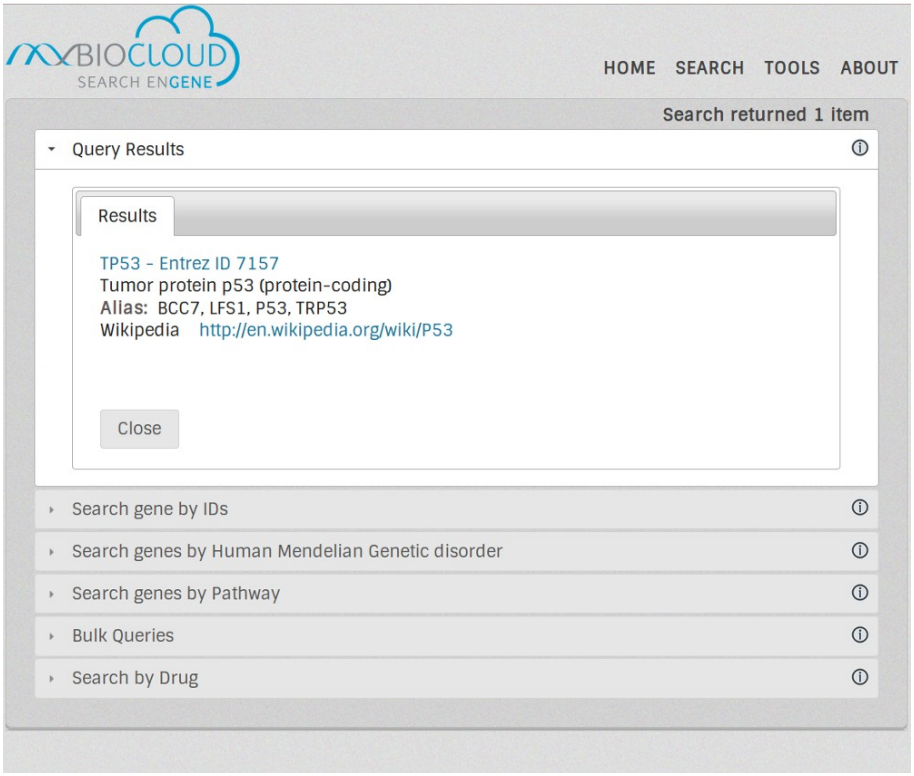

Figure 3.2. Example of query results.

When the user clicks on the link that represents the gene identifier (e.g., TP53 – Entrez ID 7157, see Figure 3.2), he/she is redirected to the *gene accordion* (see Figure 3.3) which details the context for exploring information about TP53. By expanding the panels of this accordion, the user can obtain a lot of highly detailed information and investigate every aspect of its interest in specialized databases with a redirection that is consistent with the initial query.

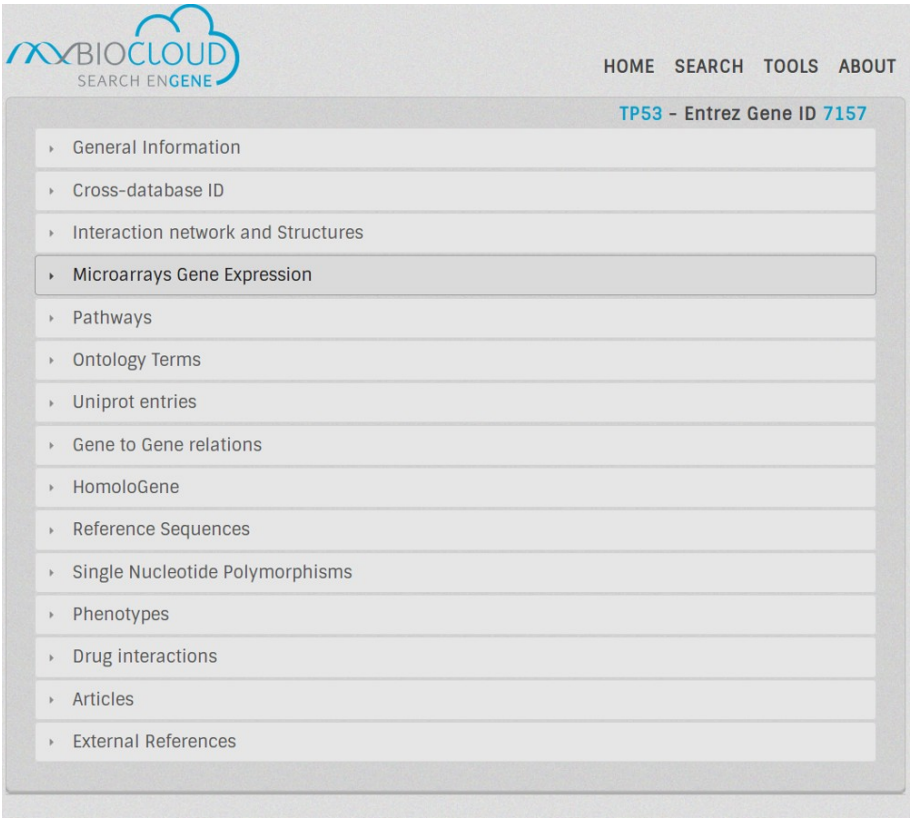

Figure 3.3. The gene accordion and its panels.

For example, Figure 3.4 shows the effects of expanding the panel "Interaction network and Structures" (*gene accordion*). Here, in order to limit the number of query results, the searching process follows a *pay-as-you-go* approach; in other words, the user is invited to load additional information if he/she needs it. In this case, the user can interactively trigger the capture of the structures related to TP53 by clicking the "Load PDB IDs" button. Captured information is stored into a distributed RAM cache with high-performance for 24 hours for fast access to cached results of datastore queries; thus, this distributed memory object caching system improves the responsiveness of the application.

Figure 3.5 presents the results of this capture, including PDB IDs, images of 3D structures from Protein Data bank, and FASTA Sequences of the corresponding structure. As shown on the left of Figure 3.5, images can be expanded. Clicking on the blue arrows (see Figure 3.5, on the right), the user is redirected to an external web site that provides more detailed information.

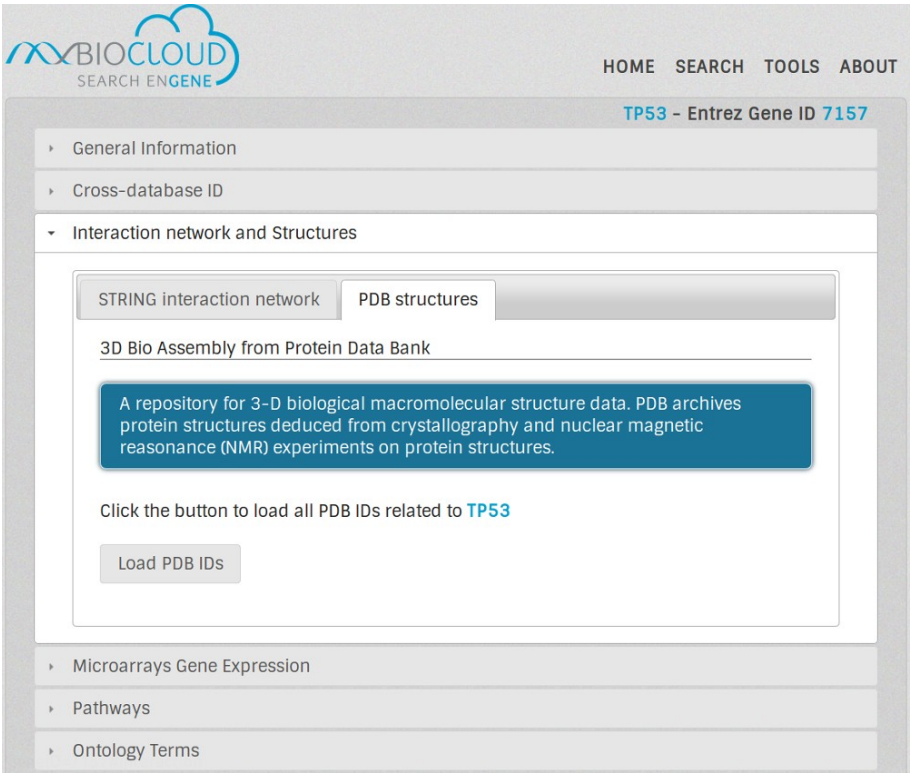

Figure 3.4. Expansion of the panel "Interaction network and Structures" (gene accordion). The "Load PDB IDs" button triggers the the capture of data in pay-as-you-go fashion.

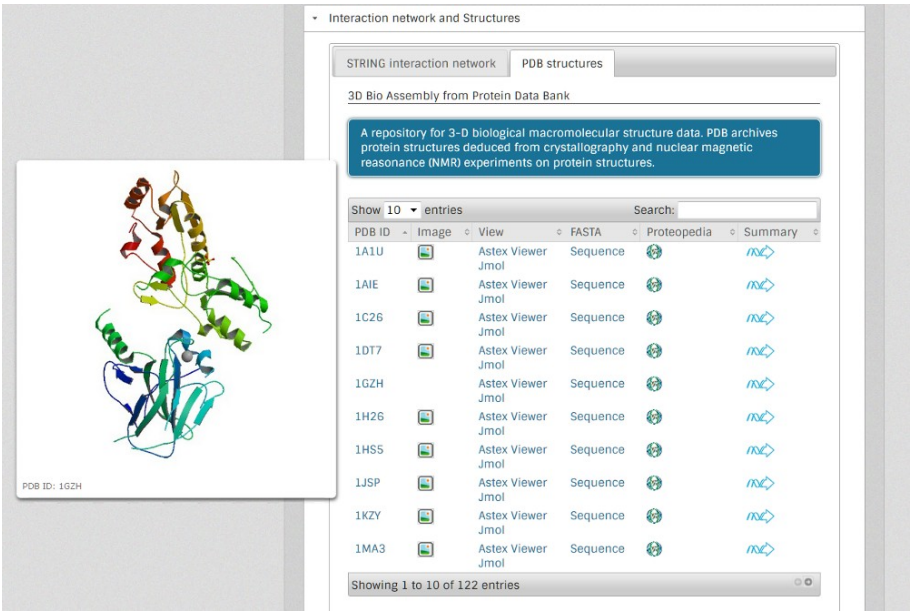

Figure 3.5. Data caught in pay-as-you-go fashion.

The same design logic affects the organization of the other panels of basic accordion (i.e., *Search genes by Human Mendelian Genetic disorder*, *Search genes by Pathway*, *Bulk Queries*, *Search by Drug*), each corresponding to a context.

As a further example, in the panel *Search by Drug,* which corresponds to the *context 'Query by drug'*, the user types a drug name and BSE provides an autocomplete based on M.A.T.A.D.O.R. [GKD+08], a public repository that annotates relationships between human genes and drugs. Figure 3.6. shows results of looking for the drug "aspirin".

The *drug accordion* appears whenever the user clicks on a drug name. As shown in Figure 3.6, when the user clicks on the link "Aspirin – Pubchem ID 2244" in the tab "Results", he/she is redirected to the *drug accordion* (Figure 3.7) which presents a lot of detailed information about this drug.

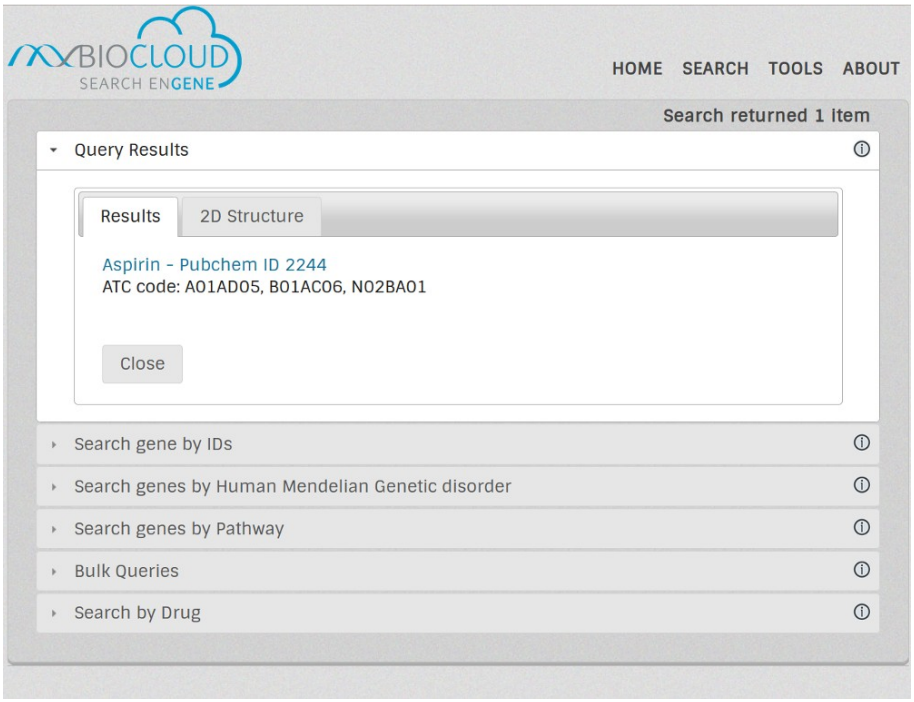

Figure 3.6. Search by drug context: query results.

Figure 3.7 depicts the panel *General Information* which is expanded by default. It shows details about the drug "Aspirin" and the related 2D structure. The *drug accordion* allows user to search for specific molecular information about drugs. For example, the *Protein Interactions* panel shows the relationships between drugs (i.e., chemicals) and genes (i.e., protein-coding genes) as annotated in the M.A.T.A.D.O.R. dataset [GKD+08].

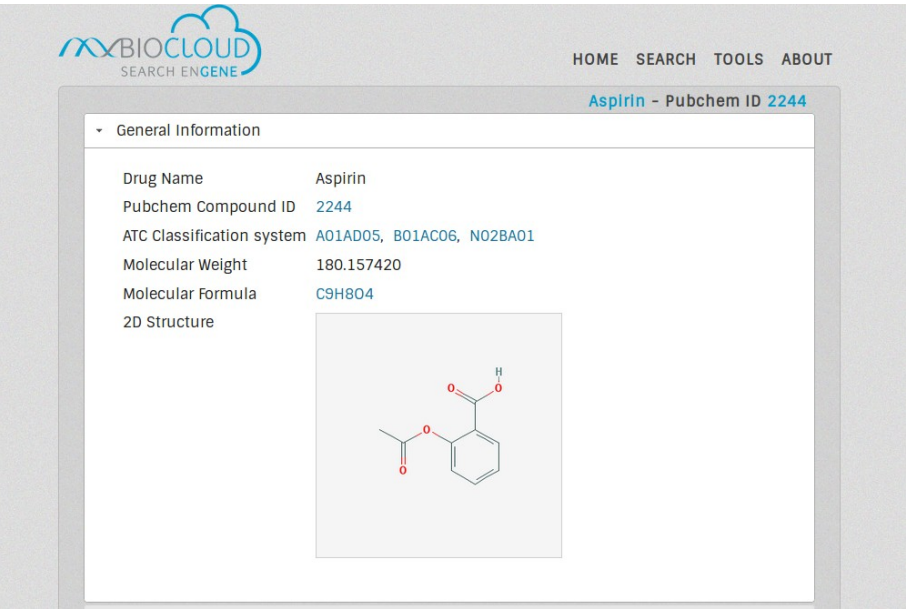

Figure 3.7. General information about a specific drug.

As illustrated in Figure 3.8, the *tool accordion* is divided into two panels: *Search overlapping information* and S*earch articles*. The first panel makes available a tool for identifying the overlapping information about shared morbid phenotypes, pathways and interacting drugs given a list of genes. The second one provides a tool for searching articles in Europe PubMed Central database by specifying a keyword. At the time of writing, Europe PMC consists of 28 million+ abstracts and 2.6 million+ full text research articles from PubMed and PubMed Central. Unlike PubMed Central, both full-text articles and the abstracts provided by PubMed are released by Europe PMC in a single point of access. Note that the *Search Articles* tool returns at most 25 entries and results are sorted by relevance.

Finally, BSE takes advantage of Google Charts to summarize some data in order to provide some information at a glance such as the gene type distribution (Pie chart in Figure 3.9) or the chromosomal distribution (stepped area chart) given a list of genes.

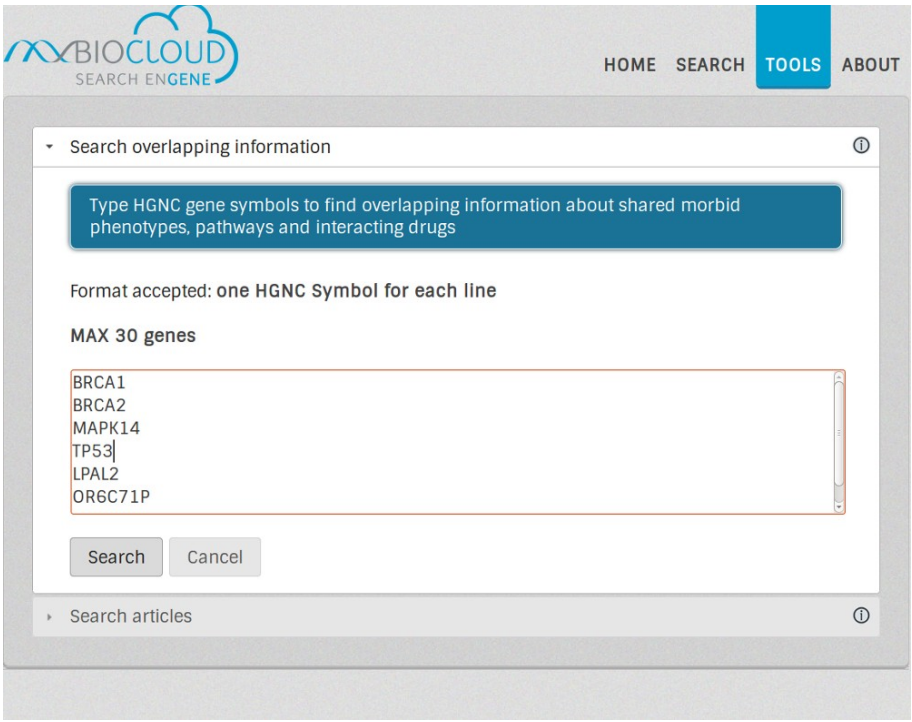

Figure 3.8. Tool accordion and its panels.

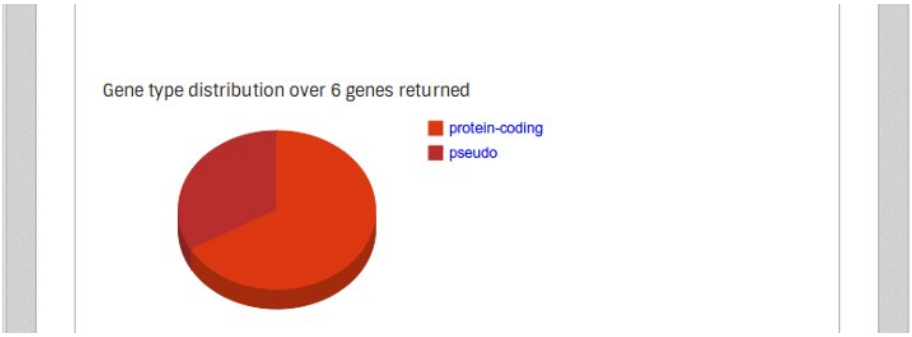

Figure 3.9. Example of chart about genes.

3.3 Implementation

BSE is built and run on Google App Engine (henceforth GAE) [AE14] (see Appendix A). It is written in Python and the catalogue of dataspace is stored into the App Engine Datastore, a managed, NoSQL, schemaless database.

As regards the graphical user interface, it was basically implemented using JavaScript and a feature-rich JavaScript library called jQuery [JQ14].

The *pay-as-you-go* approach heavily relies on Biopython [Bio14], a rich set of Python libraries which provides the ability to deal with "things" of interest to biologists while working on the cloud. In details, the Entrez Programming Utilities provided by NCBI [NCB14] were accessed by means of the *Bio.Entrez* library available in Biopython. This library was modified to run on the cloud just making some changes in the source code. BSE also exploits Django [Dja15], a high-level Python web framework, provided by GAE.

BSE exploits the following external services:

- NCBI Entrez Programming Utilities (E-utilities) [Say13]
- UniChem RESTful Web Service API [CDG+13]
- Database identifier mapping [UPA14]
- STRING API [JKS+09]
- WikiPathways Webservice/API [KPH+09]
- REST-style version of KEGG API [KGS+12]
- mygene.info REST web services [WMS+13]
- RESTful web service Europe PMC [PMC15]

BSE mainly adopts RESTful Web Services (i,e, web services that comply with the REST architectural principles [Fie00]) in order to integrate biological data because they are lightweight and particularly well suited for *ad hoc* integration on the Web [SQV+14].

3.4 Related Work

Among the closest works to BSE, I cite BioGPS [WOB+09], MyGene.info [WMS+13], and EntrezAJAX [LP10]. BioGPS makes available a centralized gene portal for integrating

distributed gene annotations. It uses PostgreSQL [Pos15] as the database backend. MyGene.info [WMS+13] provides programmatic access to BioGPS resources; that is, it offers REST Web services to query/retrieve gene annotation data. MyGene.info exploits CouchDB [Cou15]; thus, it stores data as "key-document" pairs. Both BioGPS and MyGEne.info are hosted in Amazon EC2 [Ama14] at IaaS layer. Therefore, BioGPS and MyGene.info are two interrelated services that provide a remarkable “Gene Annotation Query as a Service”. Nevertheless, the choice to pose their services directly on the IaaS layer can result in time-consuming tasks for administrate the server instances and update libraries, frameworks and so on.

As regards EntrezAJAX [LP10], it harnesses GAE [AE14] in order to provide an interface for accessing to biomedical resources accessible via the Web. Specifically, it returns JSON data from the NCBI e Utils [Say10, Say 13]. EntrezAJAX demonstrates the usefulness of using AJAX for data exchange with a server in order to build rich and interactive applications. However, EntrezAJAX focuses only on Entrez services provided by NCBI and essentially aims at representing a stepping-stone along the path of integration of biomedical resources. In addition, it benefits only minimally from the cloud capabilities of GAE; for example, it stores only the registry of developer API key and cache query results. Thus, BSE relying heavily on this stepping-stone makes further efforts in order to deeply explore cloud computing at PaaS layer and NoSQL technologies in a broader integration perspective.

3.5 Concluding Remarks

BSE is a scalable cloud-based application which allows people involved in the analysis of biological data (e.g., molecular biologists) to carry out simple and advanced searches in different specialized databases. Going further the integration of content within genetic databases, as data warehousing systems do, BSE considers dataspaces and cloud computing the basic paradigms for effective searching information from genomic resources.

Specifically, I explored how the convergence of cloud computing and dataspaces can offer both added-value service components and flexibility, making this convergence an attractive combination also for other scientific domains. BSE meets some important requirements, including high performance, scalability, and elasticity.

Most importantly, I tried to identify a set of technologies necessary in order to address big data searching issues in bioinformatics and complement the capabilities of genetic portals. Cloud computing and dataspace are paradigms relatively new; nevertheless, they seem to offer new insights in bioinformatics. Finally, even though this approach is implemented for searching data stored in genetic databases, it might reveal new directions for enhancing web-based exploration of big data in life science.

Chapter 4

Scientific Collaboration: a Case Study

Despite the large acceptance of Semantic Web technologies and their key role in bioinformatics, some concerns begin to emerge about their suitability for supporting the requirements of collaborative environments in which a research community shares and creates new knowledge.

First of all, one concern is about the possibility of widening the perspective offered by ontologies in representing interdisciplinary biomedical knowledge. An ontology provides a schema which expresses the model of a knowledge domain in terms of concepts, relationships between concepts, class hierarchies and properties. The most common types of relationships are “is-a” and “part-of”; as a result, the schemas of biomedical ontologies do not take account of other important relationships useful to tie concepts that belong to different ontologies or domains. For example, Gene Ontology (GO) [GO14], which is a *de facto* standard for knowledge representation about gene products, has developed three ontologies (i.e., structured, controlled vocabularies) that describe genes in terms of cellular components, molecular functions, and biological processes. Being these aspects defined by three different sub-ontologies, they seem independent, but actually, they are not for scientific communities.

A second concern is about the limited support offered by ontologies for an effective user interaction and collaboration. Moreover, ontologies are powerful but, at the same time, they are also complex resources [DW12] with several thousands of terms. The framework provided by RDF [RDF15] and SPARQL is effective, but researchers are often requested to be familiar with the SPARQL syntax which calls for a steep learning curve. Given the

availability of several ontologies, it is natural that biologists would create personalized versions of ontologies which reflect their particular interests. However, the merge of information from different ontologies still requires knowledge about how to reuse conceptual knowledge provided by others and how to publish the new one. Within this regard, it is increasingly hard to extract knowledge by browsing several web sites, each having its own organization, its terminology and its data formats. Instead of focusing on their real scientific interests, researchers are often involved in unearthing specialized information and remembering the navigation paths of each specific web site. This task is time consuming and daunting.

The third concern is about the role of the Semantic Web in the context of current technologies which continues to significantly influence the development of computational tools in bioinformatics. Specifically, the service-oriented paradigm has provided a new way of thinking biomedical resources in terms of computational infrastructures by positioning services as primary functional elements for data integration. Many biomedical organizations have now started to expose their IT searching services as Web services to extract valuable information from ontologies. Furthermore, new service-based paradigms have been proposed [BDP11, QEG+10] in order to help scientists in validating new collaborative research practices such as workflow systems [WHF+13]. However, despite their compliance to Semantic Web, many proposals are only suitable for solving specific problems at hand and often hinder the development of a common terminology for the representation of the domain knowledge [BBB13].

I approach the above concerns in a pragmatic way and propose COWB (COllaborative Workspaces in Biomedicine), an extensible framework for managing biomedical knowledge. COWB harnesses cloud services to provide a collaborative environment as SaaS in which biologists are actively supported, rather than just enabled, for representing and sharing knowledge about a biomedical domain they are interested in. Beyond the exploitation of the cloud paradigm, these functionalities are also provided by giving a central role to semantic information: ontologies are at core of the proposed framework because they drive the creation, storage and validation of data and metadata. Specifically, ontologies are used to define the precise meaning of biomedical concepts and their relationships in order to ensure those who are generating knowledge that they are using the most up-to-date, unambiguous, and appropriate terms.

In designing COWB, I was inspired by the typical behaviour of biomedical researchers which capture specialized knowledge from the Web. Usually, these scientists begin from the centralized view of a biomedical concept. Then, they seek to explore outward by accessing additional information from multiple resources spread all over the Web. The aim of COWB is to facilitate this approach by modelling and visualizing the domain knowledge by means of a semantic network; that is, a graph where nodes designate biomedical concepts and arcs represent relationships between those concepts. At same time, collaboration is the main prerequisite of COWB as users in different locations visualize the semantic network, interact with the same data and carry on the network implementation while they afford a collaborative environment. Although COWB is geared towards biomedical research, the ontology-centric model which supports knowledge representation in COWB is domain-independent and can be applied in any scientific area where the basic concepts can be semantically structured by a semantic network.

COWB is publicly available at *http://cowb-unica.appspot.com*.

4.1 Modelling Collaborative Knowledge

From a biomedical perspective, collaboration is very attractive for a lot of circumstances, including community learning, training and scientific research. However, it is not conceivable to have a single and universally accepted ontology which covers all biomedical domains. Thus, it becomes almost impossible to manage the biomedical knowledge in a distributed research environment where scientists are independent of each other.

To support autonomy and intelligent coordination of researchers in creating and managing shared knowledge, COWB organizes the knowledge at different levels according to the framework shown in Figure 4.1. The bottom layer, namely the *domain knowledge layer*, describes the set of meta-concepts relevant for the considered biomedical domain. It can be viewed as a semantic network of concepts where nodes indicate biomedical concepts and arcs (i.e., directed links) represent relationships between concepts.

The second layer, namely the *functional knowledge layer*, handles functional resources (FRs) which extend the *domain knowledge* and support its management. The functional resources can be divided into four categories:

- The COMMUNITY FR that describes individuals or research groups. For each user, an identifier, personal data, skills, group memberships and topics of interest are represented. A group is described through its goals, its research topics and contains information about its participants.

- The TEMPORAL FR that describes additional knowledge through common metadata; for example, data of creation of a concept.

- The SEMANTIC FR which allows for defining an unambiguous meaning of a biomedical concept and its relationships by means of ontologies.

- The DOCUMENTS FR that relies on biomedical documentation such as abstracts of scientific papers.

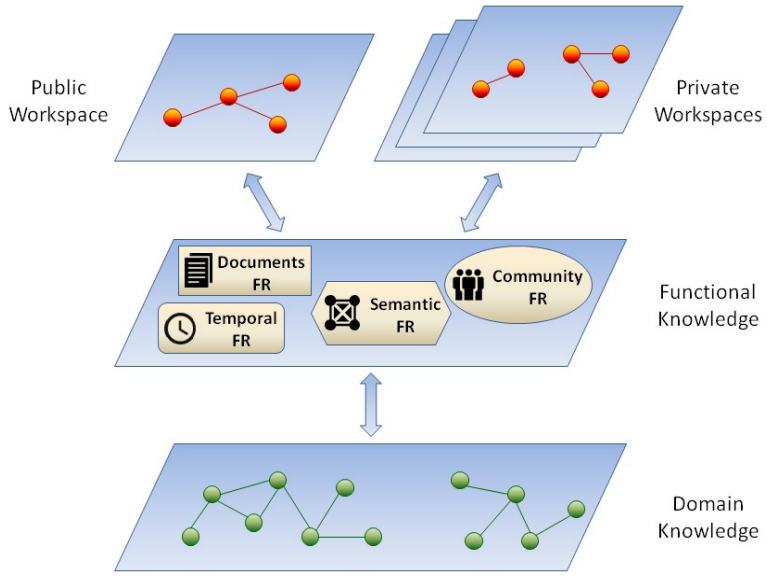

Figure 4.1. The proposed framework. The bottom layer models the domain knowledge, the second layer describes the functional resources, and the upper layer contains collaborative workspaces.

Annotations can be defined between the FRs and the domain knowledge layer to enrich the content of a particular instance and establish the foundation for its retrieval when requested. This means that FR instances can be semantically associated with the concepts of the domain knowledge by following the principle of superimposed information; that is, data or metadata "placed over" existing information sources [MD99]. For example, an annotation can associate a community resource (e.g., a researcher called Joe) with a specific domain knowledge instance created by Joe. Thus, this annotation can be exploited when searching all the domain knowledge created by a specific user or a group.

The third layer of the framework consists of a set of views over the underlying layers. These views can be divided into two categories: *public workspace* and *private workspaces*.

The *public workspace* contains knowledge objects of different granularity which deepen a specific concept of the domain knowledge in which a researcher is interested. For example, suppose that a researcher wants to explore the knowledge about the concept "mast-cell". The corresponding knowledge object in the *public workspace* is a view over the domain knowledge; that is, a semantic network which contains the node "mast-cell" and all its arcs with the other nodes. Within this semantic network, each element is annotated with the functional knowledge (i.e., author, data of creation, URI, etc.). The *public workspace* can be explored both by the community members and by public users (i.e., viewer).

As regards the *private workspace*, COWB assigns this kind of space to each community member when a researcher joins the community. Therefore, a *private workspace*, as the name suggests, is exclusively for the use of his/her owner. It has the same structure of a public workspace but operates at individual level as semantic support for personal knowledge management operations. For example, suppose that a researcher called Joe is interested in creating the concept "plasma". Joe accesses his private workspace and creates a knowledge object which contains (1) the meta-concept "plasma" captured from a specific ontology which guarantees the precise meaning of that concept, (2) the functional knowledge for its management (i.e., the author, the date of creation, the ontology which defines the concept, etc). Within his private workspace, Joe creates and manages his part of the collective knowledge and is enabled to identify the community

members which collaborate with him in order to extend and specialize the domain knowledge.

In a real collaborative Semantic Web environment, it is important to provide means for knowledge to cross the boundaries of closed local information and make accessible the broad community by visualizing the community participants which are interested in collaborating with a given user. In this way the knowledge produced by each user could be extended and reused within the community.

To tackle collaborative issues COWB provides a community model that distinguishes between two different user behaviours:

- Users who simply want to explore the semantic network (i.e., public users or viewers).

- Users who want to join the existing community in order to work on the existing semantic network (i.e., private users).

A role-based policy prescribes the rules to access to the workspaces; specifically, a role is a set of rights which determine what operations a user can perform. In agreement with user behaviours:

- *Public user or viewer roles* provide the passive access to the only *public workspace*. These roles do not require any approval and may be upgraded by the application manager. As a result, public users can only explore the knowledge.

- *Private user roles*. The application manager assigns these roles to users which want to collaborate in building and/or editing the semantic network within the community. As such, a private user can create and modify his/her own network. Furthermore, a user can connect his/her network to the network of another community member by adding new relationships between his/her nodes and nodes that belong to other users. Note that each private user can edit only his/her network element; consequently, he/she

cannot modify or delete network nodes and relationships created by other users.

Dealing with a potentially large community, COWB tracks the authorship of the pieces of semantic information that are introduced into the knowledge base. To join a community, a user must fill in an on-line form in which he/she must indicate some general information and his/her Google Account. Next, if the application manager assigns the user to the role of private user, COWB will exploit the Google Account for authentication. This authentication option permits COWB to track and verify the authorship of each piece of semantic network that users create or update.

4.2 COWB Architecture and Functionalities

The implementation of COWB in a web server with locally held data presents practical limitations not only in terms of physical resources availability (e.g., to meet peak demands), but also about the following technical concerns:

- Several semantic resources are often large in size and physically distributed; thus, there is the need for developing mechanisms that mine, on demand, only the relevant information efficiently.

- The resources of interest are often heterogeneous in structure and content. Furthermore, these resources represent data according to their own schema which defines its own concepts and relationships between concepts. Accordingly, searching strategies have to be designed for capturing information within the constraints imposed by the data source in order to comply with the data policy.

- Collaboration happens in different contexts and from different user perspectives. Therefore, it is necessary to implement mechanisms for handling and sharing the collective knowledge effectively.

The deployment of COWB in a PaaS contributes to alleviate these problems and severely curtails issues associated with scalability and performance, especially when

collaboration expands across multiple sites. Being the collaborative environment hosted in the physical infrastructure of the cloud platform, COWB exploits a close integration with web servers and standard protocols and facilitates rapid development and updates.

The COWB architecture is made up of a knowledge base, a procedural component (i.e., a set of services) and a user interface as shown in Figure 4.2.

In details, the knowledge base takes advantages of a schemaless NoSQL database that provides robust and scalable storage. In particular, COWB exploits an object-oriented database which provides a great flexibility in storing the different layers of knowledge defined by the framework improving the data management tasks in terms of elasticity and scalability.

The knowledge base stores the domain knowledge and its annotations from functional resources into three classes of data objects: the class *Node*, the class *Triple* and the class *Community*.

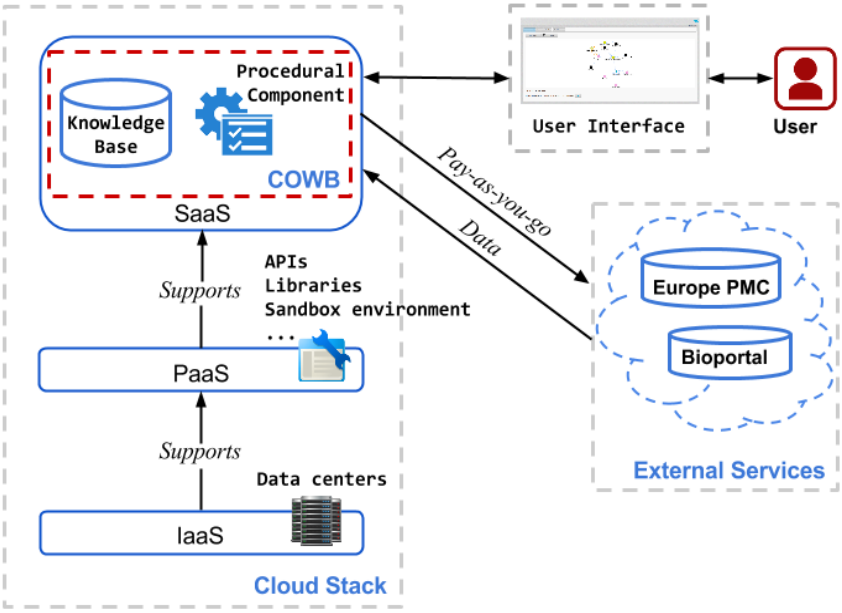

Figure 4.2. The COWB architecture.

Specifically, the class *Node* describes network nodes which map biomedical concepts. Each node has a preferred name tied to a specific ontology, a semantic type (if available) and is labelled with a unique identifier (i.e., a URI). Further pieces of information include the node author, the date of the node creation and a list of keys: each key identifies a specific triple to which the node belongs to.

As regards the class *Triple*, it maps the network structure. Each data object (i.e., an entity) stores a single triple; that is, a statement about the domain knowledge in the form of a *subject-predicate-object* expression. According to RDF terminology, triple elements are resources which are identified by means of unique identifiers (URIs). Both the subject and the object describe resources, while the predicate expresses aspects of a relationship between the subject and the object. Annotations from functional knowledge include the author and the date of creation of the triple.

Finally, the class *Community* stores information about users; that is, this class handles profiles of users which belong to the community.

The procedural component exploiting a set of services copes with issues associated with creation, management and interactive visualization of the knowledge related to public and private workspaces. It implements the following classes of services:

- *Data extraction/management services* deal with the accommodation of information extracted from web resources using services from external partners.

- *Task-oriented services* support specific procedures for network visualization.

- *Administrative services* support network management; that is, they allow private users to handle the network. In addition, these services look after security aspects in order to avoid problems related to the collaboration among users.

A collaborative environment must deal with ambiguity effectively. In order to avoid this problem, I chose to exploit the ontologies as basic mechanisms to univocally identify a concept (i.e., a resource) and efficiently structure the network. Since there is not a single ontology that contains a comprehensive knowledge to deal with all biological sub-domains,

COWB captures and integrates knowledge from many different sources. Specifically, COWB uses BioPortal [WNS+11] and Europe PMC REST services [PMC15]. According to the *pay-as-you-go* paradigm, the database holds only information about the node properties. COWB exploiting this best-effort approach includes information about nodes such as a list of synonyms, the tree associated to node's ontology, a list of scientific publications about the considered node when user requests it.

The user interface makes available some wizards for guiding researcher in managing domain knowledge and supports him/her within both the public and private workspaces. Specifically, the user interface provides two interaction modes: knowledge exploration and knowledge editing. These interactions modes are described in details in what follows.

4.2.1. Knowledge Exploration

As previously mentioned, COWB envisions users searching, selecting and capturing domain knowledge from the visual representation of the semantic network (i.e., a graph) which models the domain knowledge.

Within this graph, in order to explore the knowledge, a user must specify a preferred name of a biomedical concept associated with a network node and a number that indicates the level of neighbourhood (i.e., the number of hops). However, an autocomplete provides suggestions while a user types in the research field. Therefore, given a Graph G, the first level neighbourhood of a node N is a graph composed of all the triples in G that have N as subject or object. The second level includes the first level neighbourhood and its neighbourhood at the first level and so on. This strategy allows users to browse a highly connected network efficiently and improves the readability of the knowledge.

Figure 4.3 shows a case in which a user has searched for "mast cell" by specifying a number of hops equal to two. The concept of interest, in this case "mast cell", is represented by a triangular node and its neighbours are represented as coloured circles. Each node is assigned to one of the 15 UMLS semantic groups [MBB01] and its colour depends on its semantic group. Tooltip widgets show information about the node (i.e., preferred name, definition, author and semantic type) or about the relationship (i.e.,

predicate label and author). Triples which originate the visualized network can be exported in N-Triples format [NT14].

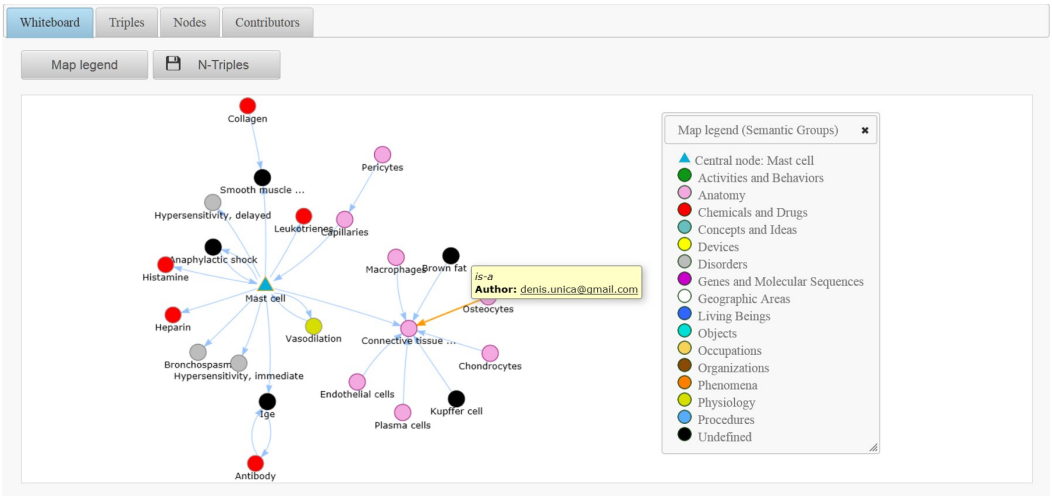

Figure 4.3. Second level neighbourhood of the concept "Mast cell". The user selects the link between the concepts "Osteocytes" and "Connective tissue cells". The tooltip shows information about the selected link (predicate label and author).

Since the graphical visualization of a level of neighbours greater than two may be complex, COWB supports the visualization of a pre-calculated clustering of nodes with a high number of connections as depicted in Figure 4.4. Here, the numbers on brackets indicate the existence of clusters and detail the number of hidden connections. A double click on the cluster centre explodes the cluster and visualizes the original network with all its nodes and arcs.

COWB enables manipulation of the network and interaction with dynamic data. Therefore, users can move the network, reduce/enlarge the size, and zoom in on selected portions. These features allow for exploring large amount of data effectively when users investigate a well-defined network portions. To produce readable views of the network, a force-direct algorithm models arcs as springs that pull linked nodes together and attempts to place nodes so that all forces are in equilibrium. This process is visually animated.

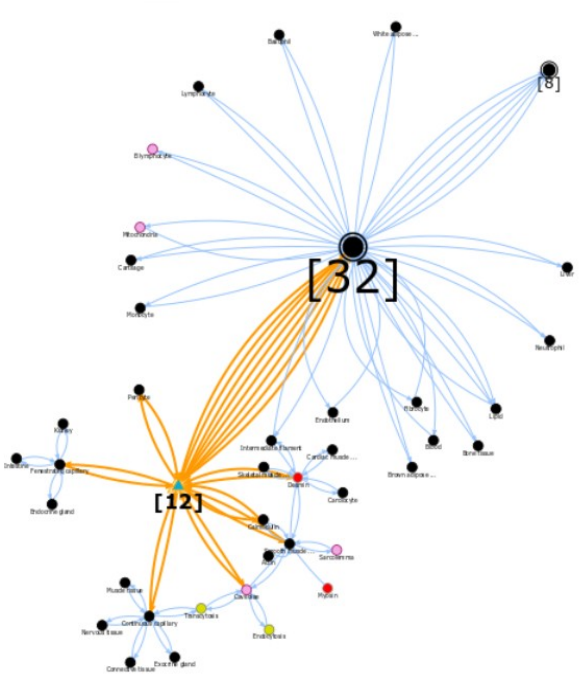

Figure 4.4. Third level neighbourhood of the concept "mast cell". A pre-calculated clustering is visualized.

4.2.2 Network Editing

Editing functionalities are strictly connected with the model of collective knowledge presented in Section 4.1. Within this model, let us consider a possible scenario where Alice is a researcher interested in representing knowledge about system biology.

At the beginning, she requires the private user privileges. Next, when the COWB application manager gives her these privileges, she starts creating from scratch her private workspace. To create a node, Alice accesses the network editing menu from the COWB main page, chooses to work on a new empty whiteboard which represents her private workspace and clicks in an empty space of the whiteboard. A wizard helps her to choose the appropriate biomedical ontology for defining the concept of interest. Figure 4.5 shows an example of such interaction where Alice creates a new node which represents the concept "plasma" and COWB captures, in *pay-as-you-go* manner, information about this concept from Bioportal. Here, the concept "plasma" is defined in four ontologies: NCIT, MESH, CRISP, and PMA. Since these definitions may differ, Alice must select the ontology which defines better this concept according to her opinion (e.g., MESH). After

her selection, a new object is automatically stored into the knowledge base. Hence, this object represents the concept "plasma" and its functional annotations; that is, the concept "plasma" is created by Alice (author) and is tied to the selected ontology.

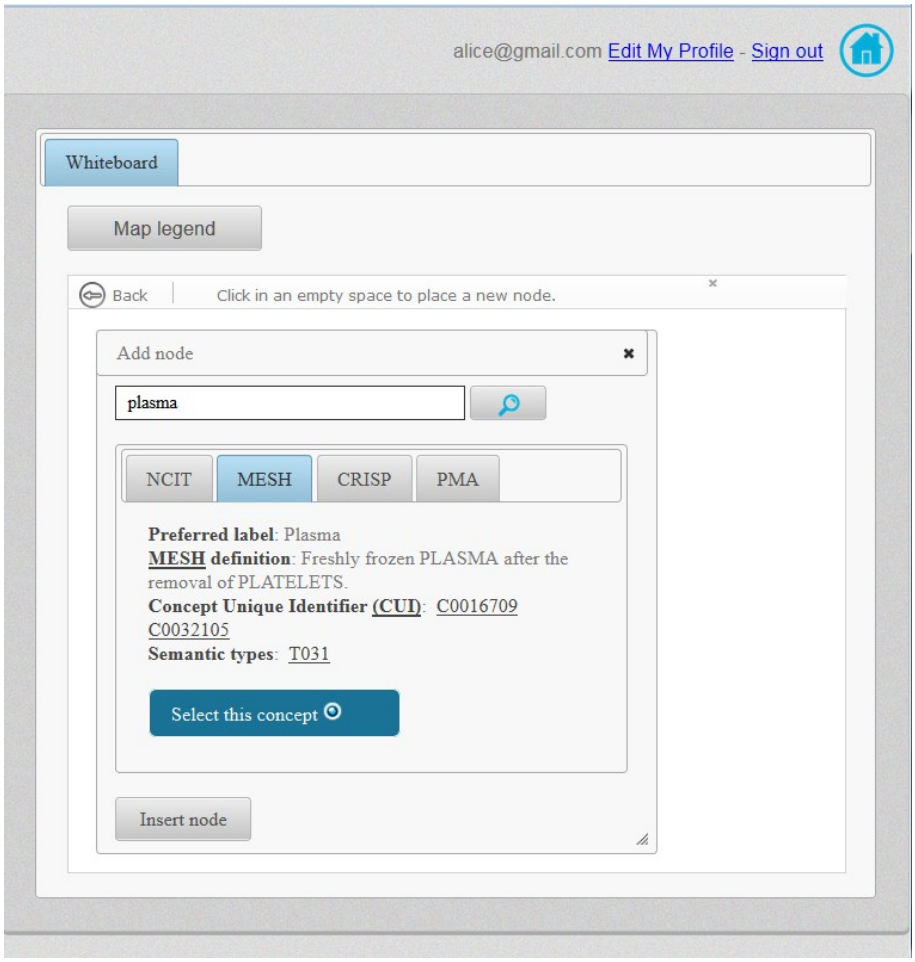

Figure 4.5. Alice creates the node "plasma". The wizard helps Alice to choose the proper ontology for defining the concept of interest.

In addition to nodes, Alice can link nodes by selecting an existing node and dragging an edge from this node to another node of the visualized network. Then, a window appears which contains the preferred label of the two nodes and a list of predicates. Specifically, each predicate is associated with a specific URI and a label which describe the relationship between the connected nodes (i.e., the subject and the object). When Alice selects a predicate from the list, the new drawn link is mapped into a triple (i.e., a *subject-predicate-object* expression) which is automatically stored into the knowledge base with the annotated functional knowledge. Alice can now browse her network and modify nodes and relationships by clicking on their graphical representations.

Now, let us suppose that Alice invites Bob, a colleague, to join the community. Bob agrees, fills in an on-line form and receives the role of private user by the COWB application manager. Just as Alice did, Bob can now draw on the whiteboard his own network. If Bob adds to his network a node which represents a concept already stored by another user (e.g., the concept “plasma” that was stored by Alice), COWB warns Bob about this fact and visualizes the node (in this case the Alice's node) on the whiteboard. Furthermore, clicking on that node, Bob can obtain the annotations about the node. For example, clicking on the node “plasma” which was created by Alice, Bob can visualize the Alice’s profile including her photo, her e-mail address, her linkedIn page and her twitter account (if declared), as shown in Figure 4.6. Besides, Bob can link the node "plasma" with a node of his own network.

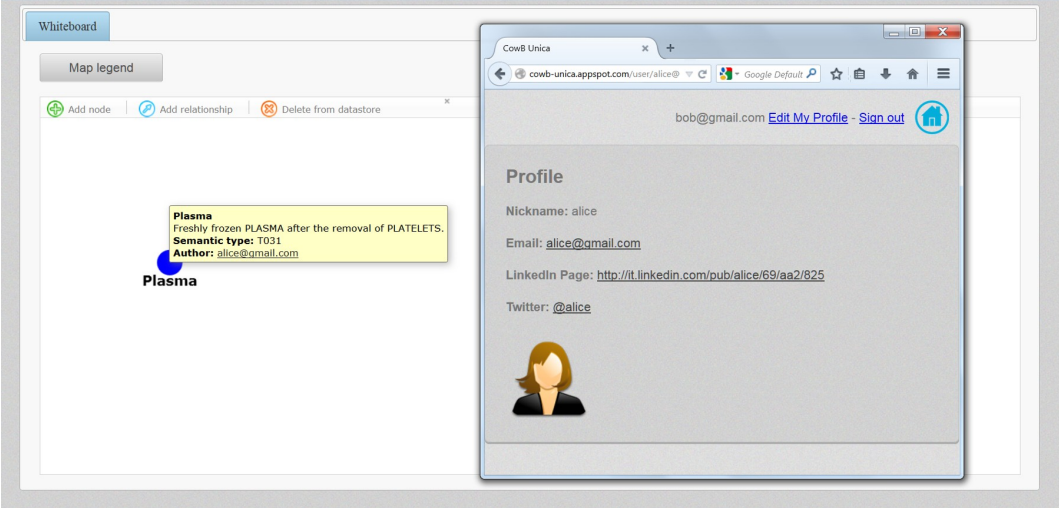

Figure 4.6. Bob visualizes the node “plasma” created by Alice. By clicking on the node, Bob visualizes the Alice’s profile.

Finally, Carol, an American biologist, visits by chance the public workspace provided by COWB. Being a public user, she can only browse the knowledge related to the public workspace. When she searches for the concept “mast cell”, COWB presents the network linked to this concept, irrespective of its authors as depicted in Fig 4.3. Note that multiple semantic networks can be presented within a single workspace; thus, COWB provides different views on the same knowledge base.

As described so far, while a user manages his personal workspace some information is stored into the network and made available for community participants to further exploration and download. However, a researcher may be curious; thus, he/she can wonder if there are other users who have connected nodes to his/her sub-network. To face up to this problem, COWB defines a special category of nodes, namely the boundary nodes. For example, given a node N and a graph G, let us assume that the node N was created by Alice, whereas the graph G was created by Bob. The node N is a boundary node in respect to the graph G, if N participates in one or more triples of the graph G; that is, if at least one triple, which belongs to G, has N as subject or object. Figure 4.7 shows a user sub-network with its boundary nodes. Nodes that belong to the user are red coloured, while boundary nodes and external links (i.e., nodes and arcs created by other users) are visualized in blue.

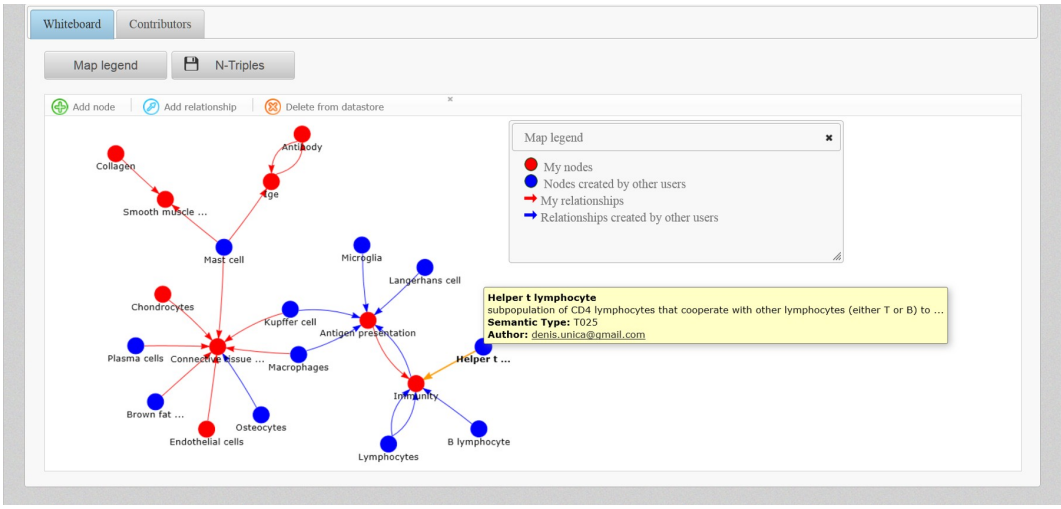

Figure 4.7. Example of a private workspace. Nodes created by the workspace owner are visualized in red, whereas nodes created by other users are depicted in blue.

4.3 Implementation

COWB is built and run on GAE [AE14] (see Appendix A). At the time of writing, it manages a manually-curated semantic network in medical biology which is stored into a schemaless NoSQL database.

For implementing COWB's core functionalities, I used Phyton, JavaScript/AJAX/jQuery and Django [Dja15]. The *pay-as-you-go* approach is supported by exploiting REST services provided by Bioportal and Europe PubMed Central.

I developed the Graphical User Interface using JQuery UI [JQ14], a set of dynamic user interactions, effects, widgets, and themes. Besides, COWB takes advantage of customized functions that I added into *vis.js* [Vis14], an open source JavaScript library that is specifically suitable for handling large amounts of dynamic data, enabling users to manipulate and interact with the data. I implemented these custom functions in order to save modifications into the database at runtime. The "graph component" made available by *vis.js* also includes a force-directed algorithm which was used to generate graphs where nodes have a minimum distance from each other.

4.4 Related Work

Related work affects several different aspects. The first aspect concerns how to model and manage scientific collaborative knowledge. It has been observed [LKN+13] that the proposed models are quite verbose; as a result, they are not very suitable for implementing data management systems. Recently, some platforms, which implement collaborative environments, have been developed in biomedicine. However, they are based on a client-server model of computing and do not include semantic features. By way of example, I cite WikiPathways [PKM+08] that is a public, collaborative platform dedicated to curation of biological pathways (i.e., networks), and BioUML [BUW14] which allows users to collaborate and draw biological maps in a similar way to Google Docs [Doc15].

The second aspect regards the architecture which supports the collaborative knowledge management. Previous research [ZL07, Zhu09] has presented some paradigms, based on P2P architecture, to build a semantic network among peers by establishing relations between semantic nodes. As it happens in COWB, a semantic link represents a semantic relationship between semantic nodes such as *similar-to*, *cause-effect* and so on. Differently from COWB, a semantic node can be an entity, a concept, a schema, or a semantic community. Analogous P2P solutions [ETB+03, BBM+02] show that this approach

requires specific algorithms in order to provide consistency among replicas; on the other hand, it is difficult to guarantee consistency in large-scale dynamic systems. In COWB, cloud technologies avoid these difficulties.

Another central aspect regards the need of promoting interoperability between different tools that has inspired the creation of graphical standards such as the SBGN notation for biological diagrams [LNH+09]. However, it has been observed [KCP+13] that there is a severe lack of adequate software for navigating and querying maps created according to the systems biology standards, as well as for collecting the user feedbacks about the map's content in an interactive manner. A number of recently tools attempt to meet this requirement. In particular, NaviCell [KCP+13] relying on Google Maps supports user-friendly exploration of large-scale maps at different scales. Similarly, CellPublisher [FLM+10], Pathway Projector [KAO+09] and PathVisio [IKP+08] exploit the geographical metaphor for navigating within the maps.

Many tools have been developed for visually exploring networks [SH07, PWS08, PHS+11]. Some of them are general-purpose; therefore, they can be used to cope with a wide range of problems. In contrast, some others are specialized for specific applications such as protein-protein interactions, pathways analysis, and gene networks. Cytoscape [LFK+10] is a case in point; in fact, it is currently a golden standard for large scale network visualization. It can support directed, undirected and weighted graphs and provides customizable layouts that allow the user to change the properties of nodes or edges. Furthermore, it incorporates statistical analysis as well as network filtering capabilities. A broad variety of additional features are made available as plug-ins (i.e., apps) mainly developed by high-experienced users.

The closest platform to COWB is a collaborative Web service platform for gene-regulatory and biochemical pathway model curation called Payao [MGK+10]. This platform combining Web 2.0 technologies and online model visualization functions enables a community to work on biological models simultaneously. Specifically, Payao reads the models in Systems Biology Markup Language (SBML) format, displays them with a process diagram editor and provides access-controlled community members with an interface for adding tags and comments to specific parts of the models. The model owner specifies the basic information about the model and indicates users who have the privileges

to view, add tags, add comments to its model. However, since PAYAO was not designed to handle semantic knowledge, I think that the layered organization of knowledge and the role-based model exploited by COWB are more suited to guarantee both the autonomy of users and the coordination of their actions.

4.5 Concluding Remarks

In this chapter, I presented a comprehensive overview and a first implementation of the Semantic Web community scenario enabled by COWB. I tried to highlight distinct challenges I tackled in the context of the Web 2.0 and Semantic Web paradigm. By showing some practical examples from COWB, I illustrated how problems may be resolved within these challenges. Compared to current centralized approaches, the COWB framework presents an alternative way to knowledge management and exploits a cloud platform to share the knowledge collectively created by a community of researchers.

This case study highlights three points.

- Currently, it is feasible to reformulate the common biomedical investigation practices by combining different technologies, including Semantic Web technologies, NoSQL databases, and cloud computing. Domain experts, rather than programmers, can be supported to model, store and explore the domain knowledge in a simple, integrated and intuitive way.

- The proposed ontology-centric approach, which deals with discovering, importing and publishing new knowledge from biomedical ontologies, allows for supporting effectively collective intelligence within biomedical communities. The integration of external services permit users to capture semantic information in a *pay-as-you-go* fashion.

- According to the Web 2.0, the visualization of semantically structured information is supported by dynamic interfaces which assist users for interactively visualizing, editing and exploring the semantic contents. Thus, users have a global view of the knowledge that can embrace interdisciplinary areas.

COWB contrasts sharply with traditional centralized computing platforms which are not only costly, but risk to become more and more inadequate to meet the applications requirements. Being a SaaS, COWB runs in a physical location of the infrastructure which is determined by the provider (usually, the server that is closest to the user). In addition, the cloud infrastructure enables the transparent storage of the knowledge base across many machines. This is beneficial to scale processing tasks when many users join the community to concurrently share information. Finally, since COWB is provided as SaaS, users avoid the installation and management of software on their own computers and benefit from software which is always up-to-date.

Although COWB is the first attempt to address the challenges discussed in this chapter, preliminary results demonstrate that it is adequate for managing a collaborative environment. The proposed framework sets out to pave the way development of future projects and systems that combine the flexibility of the cloud computing approach with the knowledge provided by ontologies.

In future, COWB should be extended in order to capture additional domain knowledge not only from ontologies but also from alternative resources such as well-curated texts as investigated in recent research [DPP14].

Finally, COWB is not an alternative and complementary tool but occupies an its own niche, being the proposed approaches interesting per se. Indeed, COWB provides the users with practical solutions which tackle some Semantic Web open research issues such as representing and managing knowledge, extracting information from ontologies, integrating data from different web resources and so on. Although the above solutions can hardly be considered as suitable for all the possible collaborative scenarios, the collaborative workspaces provided by COWB might allow biomedical researchers to formulate new insights about the Semantic Web opportunities.

Chapter 5

Knowledge Extraction: a Case Study

Microarray technology allows for collecting large-scale gene expression data and is currently used in medical diagnosis in order to identify genes that play an important role in the pathogenesis of complex diseases. Such identification requires facing the challenge of handling datasets where the number of genes, namely features, is much larger than the number of samples. Even though thousands of genes are usually investigated only a very small number of them show a correlation with the disease in question. Although many machine learning methods have been developed, it is still difficult to train and test general classification methods.

Decision trees are among the popular machine learning methods. Being produced by a greedy algorithm, a single tree may generate an unstable classification model with poor generalization accuracy; indeed, a small change to the data can result in a very different model. The proposal of random forests [Bre01], a method for classification based on the repeated growing of trees through the introduction of a random perturbation, tries to counteract such instability averaging the outcome of a great number of models fitted to the same dataset. At each node of the trees, a small subset of randomly selected features, instead of all features, are considered to split the node. As a sub-product of this technique, the identification of variables that are important in a great number of models provides suggestions in terms of variable selection.

Good generalization performance is critical for many learning algorithms, in general, and for microarray data classification in particular, since it remarks on the performance of the algorithm on new data. As demonstrated in previous research [Bre01], the generalization error is influenced by two factors: the correlation between the random trees and their individual strengths. Breiman [Bre01] further derives that as the number of random trees becomes large (i.e., it tends to infinity), the generalization error converges to a limit.

Random forests have been applied, with promising results, in analysing datasets with large dimensionality. Extending these studies to develop random forests for microarray data analysis presents an interesting research goal [ACL08]: random forest performance tends to decline when the number of features is huge and the proportion of truly informative features is small, such as with microarray data. Therefore, the effectiveness of a random forest classification process is largely dependent on its capability in facing the curse of dimensionality of gene expression data.

This case study evaluates the effects of a filtering process on the predictive performance of a random forest classifier as well as on the choice of its critical parameters. Using two popular microarray datasets, I carried out a series of classification experiments by growing random forests both on the whole set of features and on different subsets of pre-filtered features. Specifically, different parameter settings were explored in order to investigate the optimal trade-off between the number of trees and the number of variables randomly chosen at each split. The results suggest that growing few trees on small subsets of pre-filtered features, with only one variable randomly chosen at each split, presents results which compare very well with state-of-the-art studies in literature.

5.1 Background

Given a training set with N cases and M features, a random tree is built as follows:

1. N cases are randomly sampled with replacement from the original data. These N cases, which represent the new training set, are used to construct a single tree.

2. A number *mtry*, which is is held constant during the growth of the forest, is specified. Then, each node is split using the best split among a randomly selected subset of *mtry* features. Note that *mtry* is a number much smaller than M (number of features).

3. Each tree is built to the largest extent possible without pruning.

The random forest, in most cases, results more difficult to understand for humans than a single decision tree [BBH+10] because of its complexity. On the other hand, this algorithm presents several advantages that make it suitable for analysing microarray data. According to previous research [Bre01, SLT+03 , SWA08, UA06], it presents the following features:

- It can be used for both binary and multi-category classification.
- It can manage thousands of input features (without feature selection) even when there are a few cases.
- It runs efficiently on high-dimensional datasets.
- It is relatively insensitive to non-informative features.
- It makes available an embedded measure of feature importance.
- It is robust against overfitting.

In more detail, random forests can be trained in less time than a single decision tree because the method tests only *mtry* features (i.e., a small subset of the original features) and it does not do any pruning [SLT+03].

As previous mentioned, the critical parameters of a random forest are the number of trees, namely *ntree*, and number of random features to split each node of a tree, namely *mtry*. The value of *mtry* can range from 1 to M and common default values are $\sqrt{M}$ or log(M) [CWZ11].

Breiman [Bre01] states that parameters with default values often lead to excellent performance, but recent studies suggest a fine-tuning of the parameters. As demonstrated by Zhang and Wang [ZW09], it is not necessary to use the whole forest in order to reach satisfying prediction performance. In their study the size of the optimal sub-forest is in the

range of tens and some sub-forests can even overcome the original forest in terms of prediction accuracy on a breast cancer prognosis dataset. The case study presented in this chapter sets out to validate this research using both different datasets (e.g., a diagnostic dataset) and an alternative approach.

According to Genuer et al. [GPT08], applying random forests to high-dimensional classification problems, *mtry* needs to be sufficiently large for capturing important features (i.e., variables highly related to the class). As a consequence, if the number of genes is large and the percentage of meaningful information is small, it is necessary to choose large values of *mtry* in order to get better performance [LW02]. However, trees which made random splits (i.e., *mtry* equals to 1) can give very good performance for some datasets. Amaratunga et al. [ACL08] have proposed a filtering approach to decrease the contribution of trees whose nodes are populated by non-informative features. Specifically, they choose the splitting subset at each node by using a weighted random sampling instead of a simple random sampling.

5.2 Experiments

I carried out a series of experiments investigating two public microarray datasets: *Colon* [ABN+99] and *Leukemia* [GST+99]. Specifically, the *Colon* dataset is made up of 2000 genes which were measured on 62 patients. Among them, 40 samples come are from tissues of patients with colon-cancer and 22 come are from healthy parts of the colons of the same patients. *Colon* dataset is considered as one of the noisiest microarray benchmarks. As regards the *Leukemia* dataset, it consists of 7129 genes and 72 samples. These samples belong to 47 patients with acute lymphoblastic leukemia (ALL) and 25 patients with acute myeloid leukemia (AML).

The overall analysis was performed using the Weka data mining software [MEG+09]. I used a leave-one-out cross-validation procedure (LOOCV), a well-known and popular procedure in literature for performance estimation, though it has been observed that a cross-validation setting can yield overoptimistic results on small sample size domains [ND02]. The performance of the method was evaluated using the value of area under the

curve (AUC) of the receiver operating characteristic (ROC) curve in order to synthesize the information of sensitivity and specificity. Note that AUC metric is not sensitive to unbalanced distributions and is more discriminative than the accuracy metric [Faw04].

The experiments were divided into two broad classes:

- *Tuning on the original dataset.* I build different random forests using the following parameters values: (i) *ntree* = 10, 20, 30, 50, 100, 200, 300, 500, 1000, 1500; (ii) *mtry* = 1, 2, 3, 5, 10, 20, 30, 40, 50, 80. Both the choices (i) and (ii) intend to finely investigate parameters values smaller than the common default values.

- *Tuning on filtered subsets.* First, I ranked the features of the original dataset using two popular ranking methods: *Information Gain* (IG) and *Chi Squared* (χ^2). Based on their outputs, I extracted different subsets of highly-ranked features indicated as TOP10 (i.e., the first 10 top-ranked features), TOP20 (i.e., the first 20 top-ranked features) and so on. Then, I used these subsets for constructing random forests within the following parameter configurations: (i) *ntree* = 10, 20, 30, 50, 100, 200, 300; (ii) *mtry* = 1, 2, 3, 5, 10, 20, 30.

The others parameters of the algorithm not mentioned above were used with their default value [MEG+09].

5.2.1 Results About Tuning on the Original Dataset.

Figure 5.1 and Figure 5.2 depict for different values of *mtry*, the effects of changes in the parameter *ntree* on the AUC. According to Breiman [Bre01], the behaviour of AUC is asymptotic; that is, as the number of trees increases, the AUC value converges to a limit. Interestingly, in both *Leukemia* and *Colon,* I observed this asymptotic trend for *ntree* $\geq$ 100, while previous studies [SWA08, UA06] on microarray datasets made use of *ntree* values in three order of magnitude. Globally, results in Figure 5.1 and Figure 5.2 suggest that, even on high-dimensional domains, the choice *ntree* = 100 can be quite adequate, with

further increases having negligible effects and smaller values leading to more unstable AUC performance.

As regards the influence of *mtry* parameter on random forests performance, Figure 5.1 and Figure 5.2 show that, for small values (i.e., values smaller than 50) of *ntree*, the choice of high values of *mtry* (*mtry* ≥ 30 for *Leukemia* and *mtry* ≥ 5 for *Colon*) results in higher values of AUC. This seems to suggest that, when I choose to grow a forest with a small number of trees, I need to set higher values for *mtry* in order to rise the probability of randomly selecting informative variables. On the other hand, if the forest is sufficiently large (*ntree* ≥ 100), the influence of *mtry* parameter declines. In particular, no improvement in AUC performance can be observed when setting values of *mtry* > 20 and *mtry* > 10 for *Leukemia* and *Colon* respectively. Therefore, as previously observed for the *ntree* parameter, the common default setting of *mtry* = √M [SWA08, UA06], where M is the total number of features, seems to be unnecessary large since smaller values ensuring a good predictive performance at a lower computational cost.

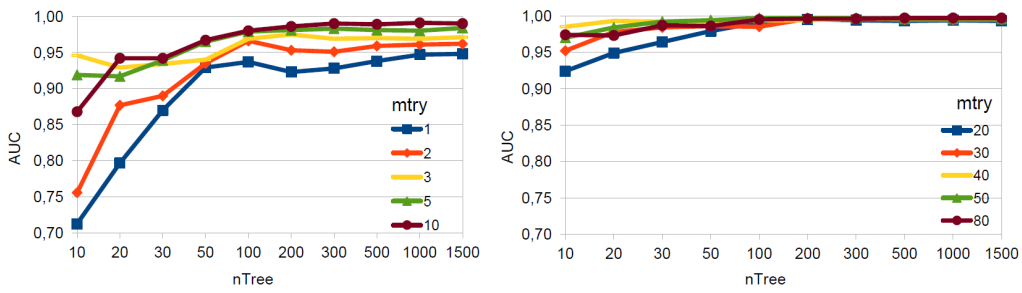

Figure 5.1. Tuning on Leukemia dataset: AUC versus ntree for mtry equal to 1, 2, 3, 5, 10 (left) and mtry = 20, 30, 40, 50, 80 (right).

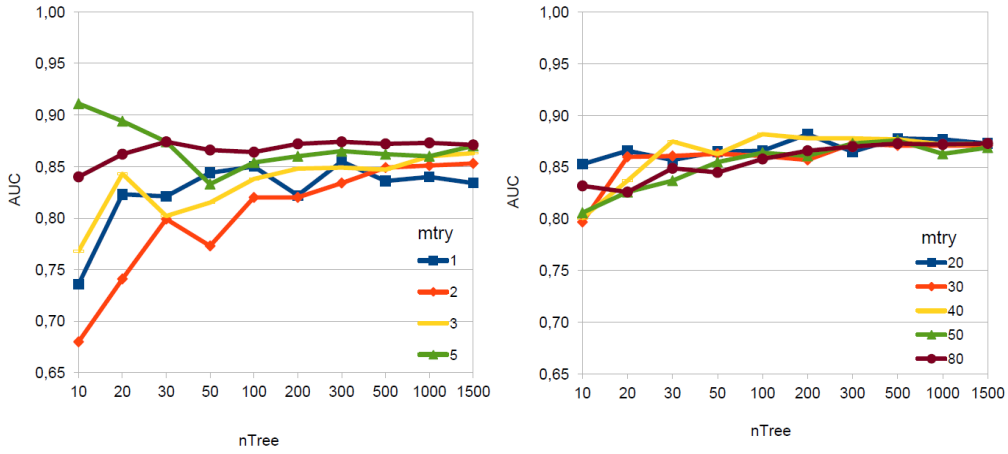

Figure 5.2. Tuning on Colon dataset: AUC versus ntree for mtry = 1, 2, 3, 5, 10 (left) and mtry = 20, 30, 40, 50, 80 (right).

5.2.2 Results About Tuning on Filtered Subsets

As above mentioned, I applied two ranking methods (IG and χ^2) and, for each ranking method, I performed tuning experiments on pre-filtered subsets of increasing size (TOP10, TOP20, etc.). Table 5.1 summarizes the "optimal" values of both parameters *ntree* and *mtry*; that is, the lowest values leading, on a given subset, to the best AUC result. As shown in Table 5.1, in most cases, the value *mtry* = 1 is sufficient to maximize the predictive performance of random forests. The optimal number of trees is also quite low, especially for *Leukemia*, where the AUC is maximized with at most 30 random trees. More trees (a few hundred at most) can be needed for *Colon* which is recognized to be a more noisy dataset. Results in Table 5.1 globally confirm what previously observed on the overall datasets: parameter values lower than common default values can lead to effective and more parsimonious classification models. Although surprising, the goodness of the choice *mtry* = 1 is also supported (for datasets of low-moderate dimensionality, as the pre-filtered datasets here considered) by some considerations reported in [Bre01].

Pre- filtered subset	Leukemia				Colon			
	IG		χ^2		IG		χ^2	
	mtry	ntree	mtry	ntree	mtry	ntree	mtry	ntree
TOP10	1	30	1	20	1	30	10	20
TOP20	1	10	1	10	1	10	1	200
TOP30	1	10	1	10	1	20	1	10
TOP50	1	10	1	20	10	10	1	10
TOP100	1	20	1	20	1	100	1	200
TOP300	1	30	1	20	1	100	1	300
TOP500	1	10	1	20	1	200	3	50

Table 5.1. Optimal values of mtry and ntree for pre-filtered subsets of increasing size, as obtained by IG and χ2 ranking methods, for both Leukemia and Colon datasets.

In addition, the pre-filtering process significantly improves the predictive performance. As regards *Leukemia,* the experiments presented in this case study gave excellent outcomes in all the subsets from TOP10 to TOP500. Only for larger subsets (e.g., TOP1000), the AUC declines if the number of random trees is not sufficiently large, as shown in Figure 5.3.a, where the AUC behaviour is shown for some subsets filtered by IG (an analogous trend has been registered for χ^2) within the "optimal" setting *mtry* = 1.

Figure 3.3 points up the asymptotic behaviour of AUC. The effectiveness of pre-filtering process is considerable as the random forests built on the selected subsets greatly outperform the random forests grown on the original dataset. However, the setting *mtry* = 1, optimal for the filtered subsets, is not so optimal for the whole dataset, where the best AUC performance is registered for *mtry* ≥ 30, as shown in Figure 5.1. Therefore, a further demonstration of the effectiveness of the pre-filtering process is given in Figure 5.3.b where the performance on the TOP20 subset (*mtry* = 1) is compared with the performance on the whole dataset (*mtry* = 40). Note that *mtry* = 40 corresponds to the "best" AUC curve in Figure5.1. The advantages deriving from pre-filtering are confirmed by the analysis on *Colon* dataset as shown in Figure 5.4.

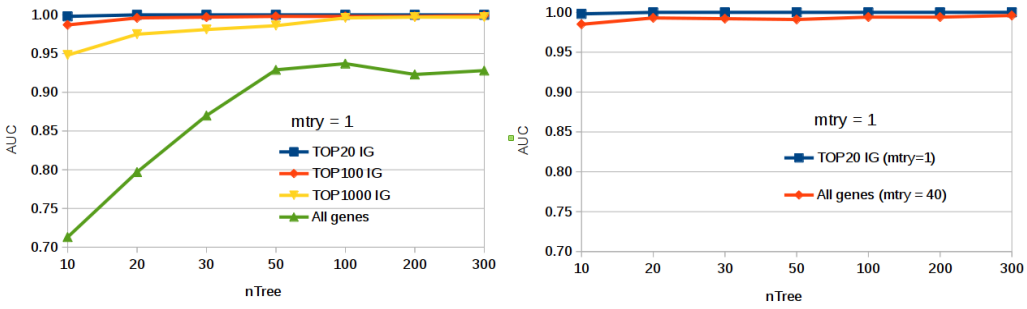

Figure 5.3. Leukemia dataset: (a) AUC versus ntree for some pre-filtered subsets and for the whole dataset (mtry = 1 for all the curves); (b) AUC versus ntree for the subset TOP20 (mtry = 1) and for the whole dataset (mtry = 40).

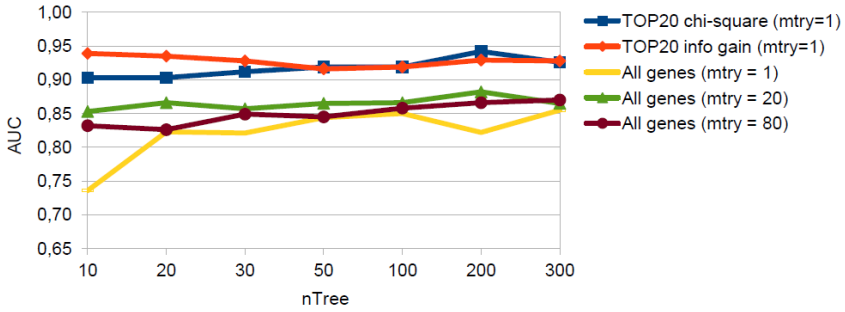

Figure 5.4. Colon dataset: AUC versus ntree for some pre-filtered subsets and for the whole dataset.

Finally, Table 5.2 shows the effectiveness of the proposed approach when compared to the most cited works in literature that applied random forests to microarray data [SWA08, UA06]. Specifically, Dìaz-Utiarte and Alvarez de Andrés [UA06] report an error rate of 0,051 for the *Leukemia* dataset (in a slightly different version) using the random forest method with *mtry* = √M and *ntree* = 5000 and without a preliminary feature selection. Within the same settings, the error rate reported for *Colon* is 0.127. By integrating a variable selection approach, the best error rates given in [UA06] for *Leukemia* and *Colon* are 0,075 and 0,159, respectively. In the research study of Alexander Statnikov et al. [SWA08] the AUC performance for *Colon* is 0.867 on the whole dataset and 0,917 with gene selection; here, the best-performing configuration is selected among the following values of parameters: *ntree* = 500, 1000, 2000 and *mtry* = 0,5·√M, 1·√M, 2·√M.

Dataset	On the full set of genes		Using a filtered subset	
	AUC	*Accuracy*	*AUC*	*Accuracy*
Leukemia	0,997	0,986	1,00	1,00
Colon	0,911	0,855	0,939	0,903

Table 5.2. Best results on Leukemia and Colon, both in terms of AUC and accuracy

5.3 Concluding Remarks

This case study presented an approach to microarray data classification that builds upon the well-known strengths of the random forests. The proposed method attempts to eliminate irrelevant variables by pre-filtering. Results on two public microarray datasets (*Colon* and *Leukemia*) confirm what expected on the basis of similar studies on filtering methods when applied to microarray data classification. The experimental analysis reveals that a pre-filtering process positively impacts both on random forest performance and on its optimal parameterization, leading to very effective and more parsimonious classification models.

Chapter 6

Conclusions

The technological advancements of the last decade have yielded a data deluge in bioinformatics. Accordingly, data integration and scientific collaboration requirements are dramatically changed. Technological challenges, which range from architectural principles to the implementation details, call for a complete re-examination about the design of Web applications and databases in bioinformatics. The work in this thesis focuses on some aspects of current issues and challenges in bioinformatics. Within the above mentioned challenges, this thesis aims at giving a contribution about the following topics:

- **Development of bioinformatics applications within a PaaS**: Currently, few bioinformatics applications are built on PaaS. This thesis outlines the benefits provided by exploiting Platform as a Service (PaaS) in order to make available scalable Web applications to biomedical community (Chapter 2).

- **Integrating information from different Web resources**: An approach is proposed which grounds on the dataspace paradigm, a new abstraction for integrating information from the Web in a *pay-as-you-go* fashion. That paradigm is exploited in the context of Biocloud Search EnGene (BSE) [DPM+13], a cloud-based application for surfing web resources. This application harnesses several technologies, including cloud computing, NoSQL databases, and Web services, in order to address dataspace requirements in terms of flexibility and scalability (Chapter 3).

- **Scientific collaboration**: A framework is proposed to support the collaborative management of shared digital resources in building a semantic network. A major role, within this framework, is played by formal semantic representations of information objects which are built up by groups of users working together in collaborative workspaces. Therefore, the framework relying on ontologies provides a collaborative knowledge management solution for biomedical communities. The proposed framework is enabled by a cloud-based application developed at PaaS layer and using a NoSQL database [DDM+14, DMP+15] (Chapter 4).

- **Knowledge extraction**: The effectiveness of random forest method has been evaluated in extracting knowledge from datasets which are affected by the so-called course of dimensionality; that is, when the number of features is huge and the proportion of truly informative features is small, as it happens with gene expression data. Thus, applying random forests in microarray data analysis presents an interesting research goal due to the additional issue of reducing the contribution of trees whose nodes are populated by non-informative features [DMP12, DMP13] (Chapter 5).

Bibliography

[ABN+99] Uri Alon, Naama Barkai, Daniel A. Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS* 96:6745-6750, 1999.

[ABR14] Paolo Atzeni, Francesca Bugiotti, Luca Rossi. Uniform access to NoSQL systems. *Information Systems* 43:117-133, 2014.

[AC15] Apache Cassandra. Retrieved January, 2015. *http://cassandra.apache.org.*

[ACL08] Dhammika Amaratunga, Javier Cabrera, and Yung-Seop Lee. Enriched random forest. *Bioinformatics* 24:2010-2014, 2008.

[AD11] Maurizio Atzori and Nicoletta Dessì. Dataspaces: Where Structure and Schema Meet. *Studies in Computational Intelligence* 375:97-119, 2011.

[AE14] App Engine. Retrieved November, 2014. *https://cloud.google.com/appengine*.

[AED15] App Engine Datastore. Retrieved January, 2015. *https://cloud.google.com/appengine/docs/python/datastore.*

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. *Technical Report* No. UCB/EECS-2009-28, 2009.

[AGT+08] Maria Susana Avila-Garcia, Anne E. Trefethen, Mike Brady, and Fergus Gleeson, Daniel Goodman, Lowering the barriers to cancer imaging. In *eScience '08*: *Proceedings of the 4th IEEE International Conference on eScience*, pp. 63-70, 2008.

[AHB15] Apache HBase. Retrieved January, 2015. *http://hbase.apache.org.*

[Ama14] Amazon EC2. Retrieved November, 2014. *http://aws.amazon.com/ec2*.

[AWS14] Amazon Web Services. Retrieved November, 2014. *https://aws.amazon.com*.

[BBH+10] Michael R. Berthold, Christian Borgelt, Frank Höppner, and Frank Klawonn. Guide to Intelligent Data Analysis. *Springer*, 2010.

[BBM+02] Matteo Bonifacio, Paolo Bouquet, Gianluca Mameli, and Michele Nori. KEx: A peer-to-peer solution for distributed knowledge management. In *Proceedings. of PAKM'02*, pp. 490–500, 2002.

[BDP11] Andrea Bosin, Nicoletta Dessì, and Barbara Pes. Extending the SOA paradigm to e-Science environments. *Future Generation Computer Systems* 27:20-31, 2011.

[BHK+14] Garth R. Brown, Vichet Hem, Kenneth S. Katz, Michael Ovetsky, Craig Wallin, Olga Ermolaeva, Igor Tolstoy, Tatiana Tatusova, Kim D. Pruitt, Donna R. Maglott, and Terence D. Murphy. Gene: a gene-centered information resource at NCBI. Nucleic Acids Research, 2014.

[BHL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems* 5(3), 2009.

[Bio14] Biopython. Retrieved November, 2014. *http://biopython.org/wiki/Biopython*.

[Bre00] Eric A. Brewer. Towards Robust Distributed Systems. *PODC Keynote*, available from *http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf*, 2000.

[Bre01] Leo Breiman. Random forests. *Machine Learning* 45:5-32, 2001.

[Bur11] Greg Burd. NoSQL. Available from *https://www.usenix.org/legacy/publications/login/2011-10/openpdfs/Burd.pdf , 2011.*

[BUW14] BioUML wiki Home Page. Retrieved December, 2014. *http://wiki.biouml.org/index.php/BioUML_wiki*.

[BYV+09] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25(6):599-616, 2009.

[CDG+06] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *OSDI'06: Seventh Symposium on Operating System Design and Implementation*. Available from http://research.google.com/archive/bigtable.html

[CDG+13] Jon Chambers, Mark Davies, Anna Gaulton, Anne Hersey, Sameer Velankar, Robert Petryszak, Janna Hastings, Louisa Bellis, Shaun McGlinchey, and John P. Overington. UniChem: A Unified Chemical Structure Cross-Referencing and Identifier Tracking System. *Journal of Cheminformatics*, 5(1):3, 2013.

[CJL+13] Heejoon Chae, Inuk Jung, Hyungro Lee, Suresh Marru, Seong-Whan Lee, and Sun Kim. Bio and health informatics meets cloud: BioVLab as an example. *Health Information Science and Systems* 1:6, 2013.

[Cou15] Apache CouchDB. Retrieved January, 2015. *http://couchdb.apache.org.*

[COW+11] David Croft, Gavin O'Kelly, Guanming Wu, Robin Haw, Marc Gillespie, Lisa Matthews, Michael Caudy, Phani Garapati, Gopal Gopinath, Bijay Jassal, et al. Reactome: a database of reactions, pathways and biological processes. *Nucleic Acid Res.* 39:D691-D697, 2011.

[CQY+13] Jiajia Chen, Fuliang Qian, Wenying Yan, and Bairong Shen. Translational Biomedical Informatics in the Cloud: Present and Future. *BioMed Research International* Volume 2013, 2013.

[CSP13] Dunren Che, Mejdl Safran, and Zhiyong Peng. From Big Data to Big Data Mining. Challenges, Issues, and Opportunities. *Database Systems for Advanced Applications* 7827:1-15, 2013.

[CWZ11] Xiang Chen, Minghui Wang, and Heping Zhang. The use of classification trees for bioinformatics. *Wiley Interdisciplinary Rev. Data Mining Knowledge Discov.*, 1(1):55–63, 2011.

[CZ14] C.L. Philip Chen, Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences* 275:14–347, 2014.

[DDB15] Amazon DynamoDB. Retrieved January, 2015. *http://aws.amazon.com/dynamodb.*

[DdLM14] Yuri Demchenko, Cees de Laat, and Peter Membrey. Defining architecture components of the Big Data Ecosystem. *2014 International Conference on Collaboration Technologies and Systems*, pp. 104-112, 2014.

[DDM+14] Nicoletta Dessì, Giacomo Diaz, Gabriele Milia, and Emanuele Pascariello. A Cloud-based Approach to a Semantic Network in Cell Biology. *BITS 2014:99-100*, available from http://bits2014.uniroma2.it/Abstract_book.pdf, 2014.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Cluster. *Sixth Symposium on Operating System Design and Implementation (OSDI'04), 2*004.

[DG10] Jeffrey Dean and Sanjay Ghemawat. MapReduce: a flexible data processing tool. *Communications of the ACM* 53(1), pp. 72-77, 2010.

[DGG+12] Lin Dai, Xin Gao, Yan Guo, Jingfa Xiao, and Zhang Zhang. Bioinformatics clouds for big data manipulation. *Biology Direct* 7:43, 2012.

[DGdL+13] Yuri Demchenko, Paola Grosso, Cees de Laat, and Peter membrey. Addressing big data issues in Scientific Data Infrastructure. *International Conference on Collaboration Technologies and Systems (CTS)*, pp. 48-55, 2013

[DH07] Xin Dong and Alon Halevy. Indexing dataspaces. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data, SIGMOD'07, ACM*, pp. 43-54, 2007.

[DHZ12] Anhai Doan, Alon Halevy, and Zachary Ives. Principles of Data Integration. *Morgan Kaufmann*, 1st edition, chapter 15, 2012.

[Dja15] Django Web Framework. Retrieved February, 2015. *https://www.djangoproject.com.*

[DMP12] Nicoletta Dessì, Gabriele Milia, and Barbara Pes. Pre-filtering Features in Random Forests for Microarray Data Classification. In Proceedings of the Workshop on *New Frontier in Mining Complex Patterns (NFMCP 2012)*, 2012.

[DMP13] Nicoletta Dessì, Gabriele Milia, and Barbara Pes. Enhancing Random Forests Performance in Microarray Data Classification. *AIME* 2013, pp. 99-103, 2013.

[DMP+15] Nicoletta Dessì, Gabriele Milia, Emanuele Pascariello, and Barbara Pes. COWB: a cloud-based framework supporting collaborative knowledge management within biomedical communities. Submitted to *Future Generation Computer System* on 12/12/2014. Accepted

[Doc15] Google Docs. Retrieved February, 2015. *http://www.google.com/intl/en-GB/docs/about.*

[DPM+13] Nicoletta Dessì, Emanuele Pascariello, Gabriele Milia, and Barbara Pes. BioCloud Search EnGene: Surfing Biological Data on the Cloud. *CIBB 2013*, *Lecture Notes in Computer Science,* pp. 33-48, 2013.

[DPP14] Nicoletta Dessì, Emanuele Pascariello, and Barbara Pes. Integrating Ontological Information About Genes. *In Proceedings of the 2014 IEEE 23rd International WETICE Conference*, pp. 417-422, 2014.

[DW12] Jürgen Dönitz and Edgar Wingender. The ontology-based answers (OBA) service: a connector for embedded usage of ontologies in applications. *Front Genet.* 3, article 197, 2012.

[EAH14] Ensembl Annotated Human Genome Data (MySQL Release 73). Retrieved November, 2014. *https://aws.amazon.com/datasets/2315*.

[EBI14] EBI Home Page. Retrieved November, 2014. *http://www.ebi.ac.uk*.

[EMB13] EMBL-European Bioinformatics Institute Annual Scientific Report 2013.

[EMR15] Amazon EMR. Retrieved February, 2015. *http://aws.amazon.com/elasticmapreduce/*

[ETB+03] Marc Ehrig, Christoph Tempich, Jeen Broekstra, Frank van Harmelen, Marta Sabou, Ronny Siebes, Steffen Staab, and Heiner Stuckenschmidt. SWAP- Ontology-

based knowledge management with peer-to-peer technology. In *Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services*, pp. 557-562, 2003.

[Faw04] Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. *Technical Report*, 2004.

[FHM05] Michael Franklin, Alon Halevy and David Maier: From Databases to Dataspaces: A New Abstraction for information Management. *Sigmod Record* 34(4):27-33, 2005.

[Fie00] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine, 2000.

[FLM+10] Lope A. Flórez, Christoph R. Lammers, Raphael Michna, and Jörg Stülke. Cell Publisher: a web platform for the intuitive visualization and sharing of metabolic, signalling and regulatory pathways. *Bioinformatics* 26:2997-2999, 2010.

[FNS11] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data Linking for the Semantic Web. *International Journal on Semantic Web and Information Systems* 7(3), 2011.

[FPG+11] Vincent A. Fusaro, Prasad Patil, Erik Gafni, Dennis P. Wall, and Peter J. Tonellato. Biomedical cloud computing with Amazon Web Services. *PLoS computational biology* 7(8), 2011.

[GCP14] Google Cloud Platform. Retrieved December, 2014. *https://cloud.google.com*.

[GD15] Google Drive. Retrieved February, 2015. *https://www.google.com/intl/en-GB/drive*.

[GKD+08] Stefan Günther, Michael Kuhn, Mathias Dunkel, Monica Campillos, Christian Senger, Evangelia Petsalaki, Jessica Ahmed, Eduardo Garcia Urdiales, Andreas Gewiess, Lars Juhl Jensen, et al. *Nucleic Acids Res.* 36(Database issue):D919-22, 2008.

[GMP15] MapReduce for App Engine. Retrieved February, 2015. *https://cloud.google.com/appengine/docs/python/dataprocessing/*

[GO14] Gene Ontology. Retrieved December, 2014. *http://www.geneontology.org*.

[GPA14] 1000 Genomes Project and AWS. Retrieved November, 2014. *http://aws.amazon.com/1000genomes*.

[GPT08] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau. Random Forests: some methodological insights. *Tech rep, INRIA,* 2008.

[GRR14] Venkat N. Gudivada, Dhana Rao, and Vijay V. Raghavan. NoSQL Systems for Big Data Management. *2014 IEEE World Congress on Services*, pp. 190-197, 2014.

[GST+99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science* 286:531-537, 1999.

[GTB+08] Maria Susana Avila Garcia, Anne E. Trefethen, Mike Brady, Fergus Gleeson, and Daniel Goodman. Lowering the barriers to cancer imaging. In *Proceedings of the 4th IEEE International Conference on eScience*, pp. 63-70, 2008.

[Had15] Apache Hadoop. Retrieved February, 2015. *http://hadoop.apache.org*.

[HBP+11] Cornelia Hedeler, Khalid Belhajjame, Norman W. Paton, Alvaro A.A. Fernandes, Suzanne M. Embury, Lu Mao, and Chenjuan Guo. Pay-as-you-go mapping selection in dataspaces. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD'11, pp. 1279-1282, 2011.

[HDI15] Microsoft Azure - HDInsight. Retrieved February, 2015. *http://azure.microsoft.com/en-us/services/hdinsight.*

[Her14] Heroku home page. Retrieved November, 2014. *https://www.heroku.com*.

[HFM06] Alon Halevy, Michael Franklin, and David Maier: Principles of dataspace systems. In *Proceedings of PODS'06*, ACM, pp. 1-9, 2006.

[HGV+09] Brian D. Halligan, Joey F. Geiger, Andrew K. Vallejos, Andrew S. Greene, Simon N. Twigger. Low cost, scalable proteomics data analysis using Amazon's cloud computing services and open source search algorithms. *Journal of proteome research* 8(6):3148-53, 2009.

[HGP12] Tony Hey, Dennis Gannon, and Jim Pinkelman. The Future of Data-Intensive Science. *IEEE Computer* 45(5):81-82 , 2012.

[HHL+11] Jing Han, E. Haihong, Guan Le, and Jian Du. Survey on NoSQL database. *In 6th International Conference on Pervasive Computing and Applications (ICPCA)*, pp. 363-366, 2011.

[HJ13] Christian Theil Have and Lars Juhl Jensen. Are graph databases ready for bioinformatics?. *Bioinformatics* 29(24):3107,3108, 2013.

[HKP11] Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. *A volume in The Morgan Kaufmann Series in Data Management Systems*. Third Edition, 2011.

[HMP14] Human Microbiome Project. Retrieved November, 2014. *https://aws.amazon.com/datasets/1903160021374413*.

[HMR+08] Bill Howe, David Maier, Nicolas Rayner, and James Rucker. Quarrying dataspaces: Schemaless profiling of unfamiliar information sources. In *Proceedings of ICDEW'08*, pp. 270-277, 2008.

[HTT09] Anthony J. G. Hey, D. Stewart W. Tansley, and Kristin M. Tolle. Jim Gray on eScience: a transformed scientific method. *The Fourth Paradigm*, 2009.

[HTT11] Anthony J. G. Hey, D. Stewart W. Tansley, and Kristin M. Tolle. The Fourth Paradigm: Data-Intensive Scientific Discovery. *Proceedings of the IEEE* 99(8), 2011.

[HWC+14] Han Hu,Yonggang Wen, Tat-Seng Chua, and Xuelong Li. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *Access, IEEE* 2:652 - 687, 2014.

[IKP+08] Martijn P. van Iersel, Thomas Kelder, Alexander R. Pico, Kristina Hanspers, Susan Coort, Bruce R Conklin and Chris Evelo. Presenting and exploring biological pathways with PathVisio. *BMC Bioinformatics* 9:399, 2008.

[ISB14] Institute for Systems Biology Home Page. Retrieved December, 2014. *http://www.systemsbiology.org*.

[Jef08] Shawn Jeffery. Pay-as-you-go Data Cleaning and Integration. Doctoral dissertation, University of California, Berkeley, 2008.

[JFH08] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. Pay-as-you-go user feedback for dataspace systems. *SIGMOD*, 2008.

[JKS+09] Lars J. Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8--a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Res.*, 37(Database issue):D412-6, 2009.

[JQ14] jQuery User Interface Home Page. Retrived December, 2014. *http://jqueryui.com*.

[KAO+09] Nobuaki Kono, Kazuharu Arakawa, Ryu Ogawa, Nobuhiro Kido, Kazuki Oshita, Keita Ikegami, Satoshi Tamaki, and Masaru Tomita. Pathway projector: web-based zoomable pathway browser using KEGG atlas and Google Maps API. *PLoS One* 4(11):e7710, 2009.

[KCP+13] Inna Kuperstein, David P.A. Cohen, Stuart Pook, Eric Viara, Laurence Calzone, Emmanuel Barillot, and Andrei Zinovyev. NaviCell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC Systems Biology* 7:100, 2013.

[KGS+12] Minoru Kanehisa, Susumu Goto, Yoko Sato, Miho Furumichi, and Mao Tanabe. KEGG for integration and interpretation of large-scale molecular datasets. *Nucleic Acids Res.* 40:D109-D114, 2012.

[KJD+12] Insik Kim, Jae-Yoon Jung, Todd F. De Luca, Tristan H. Nelson, and Dennis P. Wall. Cloud Computing for Comparative Genomics with Windows Azure Platform. Evolutionary *Bioinformmatics* 8:527-534, 2012.

[KPH+09] Thomas Kelder, Alexander R. Pico, Kristina Hanspers, Martijn P. van Iersel, Chris Evelo, and Bruce R. Conklin. Mining Biological Pathways Using WikiPathways Web Services. *PLoS ONE*, 4(7):e6447, 2009.

[Lea10] Neal Leavitt: Will NoSQL Databases Live Up to Their Promise?. *IEEE Computer* 43(2): 12-14, 2010.

[LFK+10] Christian T. Lopes, Max Franz, Farzana Kazi, Sylva L. Donaldson, Quaid Morris, and Gary D. Bader. Cytoscape Web: an interactive web-based network browser. *Bioinformatics* 26(18):2347-8, 2010.

[LFZ+09] Geng Lin, David Fu, Jinzy Zhu, Glenn Dasmalchi. Cloud Computing: IT as a Service. I*T Professional*, Vol. 11, Issue 2: 10-13, 2009.

[LKN+13] Yuan-Fang Lia, Gavin Kennedyb, Faith Ngoranb, Philip Wud, and Jane Hunter. An ontology-centric architecture for extensible scientific data management systems. *Future Generation Computer Systems* 29(2):641-653, 2013.

[LNH+09] Nicolas Le Novère, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, et al. The systems biology graphical notation. *Nat Biotechnol 27:*735-741, 2009.

[LP10] Nicholas J. Loman and Mark J. Pallen. EntrezAJAX: direct web browser access to the Entrez Programming Utilities. *Source Code for Biology and Medicine* 5(1):6, 2010.

[Lyn08] Clifford Lynch, "Big data: How do your data grow?", *Nature* 455:28-29, 2008.

[LW02] Andy Liaw and Matthew Wiener. Classification and Regression by random forest. *R News* 2:18-22, 2002.

[MA14] Microsoft Azure. Retrieved November, 2014. *http://azure.microsoft.com*.

[Mar13] Vivien Marx. The big challenges of big data. *Nature* 498:255-260, 2013

[MBB01] Alexa T. McCray, Anita Burgun, Olivier Bodenreider. Aggregating UMLS semantic types for reducing conceptual complexity**.** *Stud Health Technol Inform.* 84(Pt 1):216-20, 2001.

[McK98] Victor A. McKusick. Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders. *Johns Hopkins University Press*, 1998.

[MD99] David Maier and Lois Delcambre, Superimposed information for the internet. In *Proceedings of WebDB*, pp. 1-9, 1999.

[MEG+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11(1), 2009.

[MG11] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. *NIST Special Publication* 800-145, 2011.

[MGK+10] Yukiko Matsuoka, Samik Ghosh, Norihiro Kikuchi, and Hiroaki Kitano. Payao: a community platform for SBML pathway model curation. *Bioinformatics* 26(10):1381-1383,2010.

[MHF06] David Maier, Alon Y. Halevy, and Michael J. Franklin. Dataspaces: Co-existence with heterogeneity. In *KR*, page 3, 2006.

[Mon15] Mongo DB home page. Retrieved January, 2015. *http://www.mongodb.org.*

[MOP+11] Donna Maglott, Jim Ostell, Kim D. Pruitt, and Tatiana Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res.*, 39(Database Issue):D52–D57, 2011.

[MPR+12] Ganiraju Manyam, Michelle A. Payton, Jack A. Roth, Lynne V. Abruzzo, and Kevin R. Coombes. Relax with CouchDB - Into the non-relational DBMS era of bioinformatics. *Genomics* 100:1-7, 2012.

[Mur07] San Murugesan. Understanding Web 2.0. *IT Professional* 9(4):34:41, 2007.

[NCB14] National Center for Biotechnology Information Home Page. Retrieved November, 2014. *http://www.ncbi.nlm.nih.gov*.

[ND02] Ulisse M. Braga-Neto and Edward R. Dougherty. Is cross-validation valid for small-sample microarray classification?. *Bioinformatics* 20:374-380, 2004.

[Neo15] Neo4j Home Page. Retrieved January, 2015. *http://neo4j.com.*

[NIST15] National Institute of Standards and Technology. Retrieved February, 2015. *http://www.nist.gov.*

[Nos14] NoSQL databases. Retrieved November, 2014. *http://www.nosql-database.org.*

[NT14] W3C Recommendation. Retrieved December, 2014. *http://www.w3.org/TR/n-triples*.

[ODS13] Aisling O'Driscoll, Jurate Daugelaite, and Roy D. Sleator. 'Big Data', Hadoop and Cloud Computing in Genomics. *Journal of Biomedical Informatics* 46, pp. 774-781, 2013.

[OLe98] Daniel E. O'Leary. Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems* 13 (3):34-39, 1998.

[OMN10] Brian D O'Connor, Barry Merriman, and Stanley F. Nelson. SeqWare Query Engine: storing and searching sequence data in the cloud. In P*roceedings of the 11th Annual Bioinformatics Open Source Conference (BOSC)*, 2010.

[Ora15] Oracle NoSQL database. Retrieved January, 2015. *http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html*

[Ori15] OrientDB. Retrieved January, 2015. *http://www.orientechnologies.com.*

[OWL15] Web Ontology Language (OWL). Retrieved February, 2015. *http://www.w3.org/2001/sw/wiki/OWL.*

[Pas08] Claude Pasquier. Biological data integration using Semantic Web technologies. *Biochimie* 90:584:594, 2008.

[PC14] Pubcrawl. Retrieved December, 2014. *https://code.google.com/p/pubcrawl*.

[PHS+11] Georgios A. Pavlopoulos, Sean D. Hooper, Alejandro Sifrim, Reinhard Schneider, and Jan Aerts. Medusa: A tool for exploring and clustering biological networks, *BMC Research Notes* 4:384, 2011.

[PKM+08] Alexander R. Pico, Thomas Kelder, Martijn P. van Iersel, Kristina Hanspers, Bruce R. Conklin, Chris Evelo. WikiPathways: pathway editing for the people. *PLoS Biology* 6:e184, pp. 1403-1407, 2008.

[PMC15] Europe PMC. Retrieved February, 2015. *http://europepmc.org/restfulwebservice.*

[Pos15] PostgreSQL. Retrieved February, 2015. *http://www.postgresql.org.*

[PTB+12] Kim D. Pruitt, Tatiana Tatusova, Garth R. Brown, and Donna R. Maglott. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic Acids Res.*, 40(Database issue):D130-135, 2012.

[PTPT+13] Pablo Pareja-Tobes, Eduardo Pareja-Tobes, Marina Manrique, Eduardo Pareja, and Raquel Tobes. Bio4J: An Open source biological data integration platform. *IWBBIO*, 2013

[PWS08] Georgios A. Pavlopoulos, Anna-Lynn Wegener, Reinhard Schneider. A survey of visualization tools for biological network analysis. *BioData Mining* 1:12, 2008.

[QEG+10] Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Tak-Lon Wu, Yang Ruan, Saliya Ekanayake, et al. Hybrid cloud and cluster computing paradigms for life science applications. *BMC bioinformatics* 11(Suppl 12):S3, 2010.

[Rav15] RavenDB. Retrieved January, 2015. *http://ravendb.net.*

[RDF15] Resource Description Framework (RDF). Retrieved February, 2015. *http://www.w3.org/RDF.*

[RE14] Regulome Explorer. Retrieved December, 2014. *https://code.google.com/p/regulome-explorer*.

[RSK+11] Shoba Ranganathan, Christian Schönbach, Janet Kelso, Burkhard Rost, Sheila Nathan, and Tin Wee Tan. Towards big data science in the decade ahead from ten years of InCoB and the 1st ISCB-Asia Joint Conference. *BMC Bioinformatics*, 12 (suppl 13): S1, 2011.

[RWE13] Ian Robinson, Jin Webber, and Emil Eifrem. Graph databases. *O'Reilly Media*, 2013.

[SAR+07] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, *et al*. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology* 25:1251-1255, 2007.

[Say10] Eric Sayers. A General Introduction to the E-utilities. In Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US), available from *http://www.ncbi.nlm.nih.gov/books/NBK25497*, 2010.

[Say13] Eric Sayers. E-utilities Quick Start. In Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US), available from *http://www.ncbi.nlm.nih.gov/books/NBK25500*, last update: August 9, 2013.

[SH07] Matthew Suderman and Matthew Hallet. Tools for visually exploring biological networks. *Bioinformatics* 23 , pp. 2651-2659, 2007.

[SIB14] Swiss Institute of Bioinformatics Home Page. Retrieved November, 2014. *http://www.isb-sib.ch*.

[SK10] Pradeep Kumar Sreenivasaiah and Do Han Kim. Current trends and new challenges of databases and web application for systems driven biological research. *Front Physiol* 1:147, 2010.

[SLB+11] Sherif Sakr, Anna Liu, Daniel M. Batista, and Mohammad Alomari. A Survey of Large Scale Data Management Approaches in Cloud Environments. *Communications Surveys & Tutorials*, *IEEE* 13(3), 2011.

[SLT+03] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson , Robert P. Sheridan, and Bradley P. Feuston. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. J*ournal of Chemical Information and Computer Sciences* 43:1947-1958, 2003.

[SOH14] Hugh P. Shanahan, Anne M. Owen, Andrew P. Harrison. Bioinformatics on the Cloud Computing Platform Azure. *PLoS ONE* 9(7), 2014.

[SQV+14] Quan Z. Shenga, Xiaoqiang Qiaob, Athanasios V. Vasilakosc, Claudia Szaboa, Scott Bournea, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences* 280:218-238, 2014.

[SR13] Michael Stonebraker and Judy Robertson. Big data is 'buzzword du jour;' CS academics 'have the best job'. *Magazine Communications of the ACM* 56(9):10-11, 2013.

[SRG13] Xosé M. Fernández-Suárez, Daniel J. Rigden, and Michael Y. Galperin. The 2014 Nucleic Acids Research Database Issue and an updated NAR online Molecular Biology Database Collection. *Nucleic Acids Research*, 2013.

[Sur12] Surajit Chaudhuri. What Next? A Half-Dozen Data Management Research Goals for Big Data and the Cloud. In *Proceedings of the 31st symposium on Principles of Database System*s (PODS), pp. 1-4, 2012.

[SWA08] Alexander Statnikov, Lily Wang, Constantin F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics* 9:319, 2008.

[TCB+13] Robi Tacutu, Thomas Craig, Arie Budovsky, Daniel Wuttke, Gilad Lehmann, Dmitri Taranukha, Joana Costa, Vadim E Fraifeld, and João Pedro de Magalhães. Human Ageing Genomic Resources: Integrated databases and tools for the biology and genetics of ageing. *Nucleic Acids Research* 41(D1):D1027-D1033, 2013.

[TOG+10] Paolo Di Tommaso, Miquel Orobitg, Fernando Guirado, Fernado Cores, Toni Espinosa, and Cedric Notredame. Cloud-Coffee: implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud. *Bioinformatics* 26:1903-4, 2010.

[UA06] Ramón Díaz-Uriarte1 and Sara Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics* 7:3, 2006.

[UPA14] Uniprot Programmatic Access. Retrieved November, 2014. *http://www.uniprot.org/help/programmatic_access*.

[Vis14] Vis.js Home Page. Retrieved December, 2014. *http://visjs.org*.

[Vol15] Project Voldemort. Retrieved January, 2015. *http://www.project-voldemort.com/voldemort/*

[WHF+13] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* 41(Webserver-Issue):557-561, 2013 .

[WKF+10] Dennis P. Wall, Parul Kudtarkar, Vincent A. Fusaro, Rimma Pivovarov, Prasad Patil, and Peter J. Tonellato. Cloud computing for comparative genomics. *BMC bioinformatics* 11:259, 2010.

[WMS+13] Chunlei Wu, Ian MacLeod, Andrew I. Su. BioGPS and MyGene.info: organizing online, gene-centric information. *Nucl. Acids Res.* 41(Database issue):D561-D565, 2013.

[WN11] Paweł Widera and Natalio Krasnogor. Protein Models Comparator: Scalable Bioinformatics Computing on the Google App Engine Platform. *The Computing Research Repository (CoRR),* 2011.

[WNS+11] Patricia L. Whetzel, Natalya F. Noy, Nigam H. Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. *Nucleic Acids Res.* 39:W541-5, 2011.

[WO15] W3C Standards – Ontology. Retrieved February, 2015. *http://www.w3.org/standards/semanticweb/ontology.*

[WOB+09] Chunlei Wu, Camilo Orozco, Jason Boyer, Marc Leglise, James Goodale, Serge Batalov, Christopher L. Hodge, James Haase, Jeff Janes, Jon W. Huss and Andrew I. Su. BioGPS: an extensible and customizable portal for querying and organizing gene annotation resources. *Genome Biology* 10:R130, 2009.

[WS14] Semantic Web. Retrieved December, 2014. *http://www.w3.org/standards/semanticweb*.

[XMR+11] Zuoshuang Xiang, Chris Mungall, Alan Ruttenberg, Yongqun He. Ontobee: A Linked Data Server and Browser for Ontology Terms. In *Proceedings of the 2nd International Conference on Biomedical Ontologies (ICBO),* pp. 279-281, 2011.

[YSP+13] Wanjuan Yang, Jorge Soares, Patricia Greninger, Elena J. Edelman, Howard Lightfoot, Simon Forbes, Nidhi Bindal, Dave Beare, James A. Smith, I. Richard Thompson, et al. Genomics of Drug Sensitivity in Cancer (GDSC):a resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Res.* 41(D1):D955-D961, 2013.

[ZCB10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications* 1(1):7-18, 2010.

[ZGL+11] Lu Zhang, Shengchang Gu, Yuan Liu, Bingqiang Wang, and Francisco Azuaje. Gene set analysis in the cloud. *Bioinformatics* 28:294-295, 2011.

[Zhu09] Hai Zhuge. Communities and emerging semantics in semantic link network: Discovery and learning. *IEEE Transactions on Knowledge and Data Engineering* 21(6):785-799, 2009.

[ZL07] Hai Zhuge and Xiang Li. Peer-to-peer in metric space and semantic space. *IEEE Transactions on Knowledge and Data Engineering* 19(6):759-771, 2007.

[ZW09] Heping Zhang and Minghui Wang. Search for the smallest random forest. *Stat Interface* 2:381, 2009.

Appendix A: Google App Engine

Google App Engine (GAE) [AE14] is a fully-managed Platform as a Service which allows developers to build, deploy and run applications on Google's Infrastructure. In details, GAE is part of the Google Cloud Platform [GCP14]. GAE is currently used to develop a wide range of applications such as enterprise applications, scalable web and mobile applications, and games.

GAE supports applications written in Python, Java, PHP, and GO. In addition, it makes available several existing frameworks, including Django, Flask, Spring and webapp2.

As regards the storage, GAE provides a schemaless NoSQL database. In particular, this NoSQL database is an object-oriented database called Datastore. Unlike traditional relational databases, this database makes use of a distributed architecture to automatically manage scaling to very large amount of data. In addition, it guarantees atomic transactions and high availability of reads and writes.

According to the GAE documentation [AED15]:

> *The Datastore holds data objects known as entities. An entity has one or more properties, named values of one of several supported data types [...] Each entity is identified by its kind, which categorizes the entity for the purpose of queries, and a key that uniquely identifies it within its kind. [...] Entities of the same kind can have different properties, and different entities can have properties with the same name but different value types.*

The Datastore interface provides a rich set of API for modeling data. In addition, this interface also include a SQL-like query language called GQL for retrieving

objects (i.e., entities) or keys from the App Engine datastore. The Datastore uses by default a configuration called High Replication Datastore (HRD). This configuration implies that data is replicated across multiple datacenters exploiting a system based on the Paxos algorithm in order to guarantee a high level of availability for reads and writes. Note that most queries present a weak consistency; specifically, they are eventually consistent.

Finally, it is worth highlighting that GAE runs the apps in a secure sandboxed environment; therefore, an application exploits a reliable environment independent of the infrastructure (i.e., physical location of the server) or the operating system. This "sandbox" environment guarantees not only security but also automatic scaling and load balancing across multiple servers in order to meet peaks demand.