Expert Systems with Applications 40 (2013) 3943–3952

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology

Daniela Xavier [a], Federico Morán [a], Rubén Fuentes-Fernández [b], Gonzalo Pajares [b,*]

[a] Department of Biochemistry and Molecular Biology I, Universidad Complutense de Madrid, Avd. Complutense s/n, 28040 Madrid, Spain
[b] Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, C/Profesor José García Santesmases s/n, 28040 Madrid, Spain

ARTICLE INFO

Keywords:
CommonKADS
Expert system
DNA sequence
Functional annotation
Bioinformatics

ABSTRACT

Finding the genes that exist within a DNA sequence and assigning them biological features and functions is one of the biggest challenges of Genomics. This task, called *annotation*, has to be as accurate and reliable as possible, because this information will be applied in other researches. Ideally, each sequence should be annotated and validated by a human expert, who has the knowledge to infer the most appropriate annotation. Nevertheless, the huge amount of genomic data produced by the new sequencing technologies prevents this practice. Developing expert systems that are able to annotate sequences automatically and emulate the expert involvement in certain key points of the process would enhance the annotation quality. In this work, the CommonKADS methodology is innovatively applied for this purpose. It is used to structure and model the knowledge required to build an expert system able to deal with the functional part of sequence annotation, i.e. establishing the biological purpose of the sequence. This approach provides the first general framework for the aforementioned problem, which can be easily extended to related issues.

1. Introduction

With the advent of new sequencing technologies, organisms and even complete ecosystems can be sequenced at low cost and in short periods of time. Generating genomic data is no longer a problem, but to process, analyse and apply these data in a significant and useful way still are (Friedberg, 2006). The huge amount of genomic data slows down or even prevents the execution of whatever process that needs the human intervention during processing or analysis stages.

Gene annotation is one of the major challenges of the Genomics. This task consists in finding the genes that exist within a DNA sequence and assign them biological features, such as the name of the protein they code for or the biological process they are involved in. The *structural annotation* discovers where are the known genes, genetic markers and other landmarks. The *functional annotation* aims to predict the biological function of genes and proteins. This paper takes into account only the functional annotation.

The annotation assigned to a sequence should be as accurate and reliable as possible, since it could be used in further biological, medical, or pharmaceutical researches. Moreover, uploading annotated sequences to public databases and using this information to annotate other sequences is a common practice in the Bioinformatics community. Therefore, a miss-annotation could be propagated to future annotations (Friedberg, 2006).

The process to annotate a sequence involves the execution of pipelines composed by many Bioinformatics programs. Highly skilled professionals analyse their outputs and, based on their Biological and Biochemistry knowledge, infer the most appropriate information for each sequence. In spite of the trustful and accurate character of the manual annotation, this process is extremely time consuming, labour-intensive, and expensive, and therefore, it is not suitable for great volumes of data. In order to avoid the drawbacks of manual annotation, automated annotation methods need to be employed for the vast majority of genomes (Edwards, Stajich, & Hansen, 2009).

This approach sacrifices some quality in the annotation to get results faster and cheaper by removing the experts. However, given the potential impact of miss-annotations, its results has to be manually revised (or *curated*), recreating again the manual bottleneck.

One possible solution to this problem is to design an Expert System (ES) that is capable of emulating the human expertise during the annotation process. However, develop such system is a laborious and complex task that requires a deep comprehension of the applications domain. Moreover, it requires being able to elicit the relevant knowledge and providing it in a suitable format for automated processing.

Obtaining the knowledge required to solve the studied problem is a crucial phase in the development of any Knowledge-Based

* Corresponding author. Tel.: +34 1 3947546; fax: +34 1 3947547.
E-mail address: pajares@fdi.ucm.es (G. Pajares).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.088

System (KBS), and in particular ES. The success of the system depends to a large extent on the accuracy of the information acquired. There are knowledge elicitation techniques that can facilitate this process, such as structured interviews, protocol analysis, and laddered grid (Burton, Shadbolt, Rugg, & Hedgecock, 1990).

The development of Knowledge-Based Systems (KBS) requires the ability to understand and structure the knowledge in order to incorporate it into the ES. Knowledge Engineering provides methodologies that facilitate this process and consequently helps the systems design and implementation. CommonKADS (Schreiber et al., 2000) is a leading methodology to support structured knowledge. This methodology is a flexible and powerful tool that can be employed in any context-based problem.

This work consists in using CommonKADS approach to analyze and structure the knowledge required to develop an ES for functionally annotating DNA sequence without taking into account the genomic context. As far we can ascertain, this is the first formal description of the application of this methodology in the bioinformatic field. Therefore, here is presented a novel general framework to the functional annotation problem that can be adapted for different pipelines and extended to related matters. The knowledge employed here was obtained using the knowledge elicitation technique *structured interview* and was extract from experts in Biology and Bioinformatics.

The paper is organised as follows. Section 2 provides the background needed to understand the annotation problem, and describes the current state of the art in tools for annotation. The presentation of the proposed approach starts in Section 3 with an overview of the CommonKADS methodology and the knowledge elicitation techniques applied. Section 4 describes the knowledge modelling process for the aforementioned problem, while Section 5 characterises the main requirements for the related system. Finally, Section 6 discusses the conclusions and future work.

2. Biological background

The hereditary information of all living being, with exception of virus, is stored in a macromolecule called Deoxyribonucleic Acid (DNA). This molecule consists of two complementary long strands, linked through hydrogen bonds, twisted around each other, forming a double helix shape. The strands are mainly composed of smaller molecules called nucleotides. Each nucleotide, in turn, consists of a deoxyribose sugar, one phosphate and one of the four nitrogen-rich bases: Adenine (A), Guanine (G), Thymime (T) and Cytosine (C). A DNA sequence is formed by the combination of these four bases (Fig. 1a) and its length is measured in base pairs (bp), that is, according to the number of bases in the string. The complete sequence along each strand is called genome.

Some of these sequences contain the genetic information required to produce other cellular components, such as proteins. These sequences are the genes and are responsible for life maintenance and transmission of genetic traits to the descendants. In most organisms, genes consist of exons and introns, that is, coding and non-coding sequences respectively. During the transcription (see Fig. 1a), exons are transcribed into Messenger Ribonucleic Acid (mRNA), which is further translated (see Fig. 1a) into a sequence of amino acids.

Proteins are present in every cell and are responsible for all metabolic processes produced in an organism and for the life maintenance. In a very simplistic way, a protein can be viewed as a long chain of thousands units called amino acids held together by peptide bonds. Due to a biological phenomenon called alternative splicing, where different combinations of exons can be used to create a mRNA, one gene is capable of coding different proteins (Fig. 1b).

The role of a protein in the metabolic processes depends on its structure. The protein sequence is its primary structure, which can be written as a combination of the twenty amino acids letters (see Fig. 1c.1). This structure sometimes folds, creating turns, helixes or sheets, which form a two-dimensional organisation called secondary structure (see Fig. 1c.2). The set of secondary structures folded in the space forms the tertiary structure (see Fig. 1c.3). This three-dimensional structure is determined by its amino acids sequence and is directly related to the protein function. The prediction of the tertiary structure based on its primary organisation has been shown to be a NP-complete problem (Berger & Leighton, 1998; Crescenzi, Goldman, Papadimitriou, Piccolboni, & Yannakakis, 1998).

As depicted in Fig. 1d, the protein architecture can be composed of one or more *families* and *domains*. There are some parts of a protein that can evolve, function and exist independently of the whole protein chain. These regions are denominated *protein domains*, and are usually high-density zones due to the accumulation of folds. A *protein family* is a group of evolutionary-related proteins that have similar primary and/or tertiary structures and resembling functions. In general, proteins that share domains are grouped in the same family. The members of a protein family are known as homologous proteins. If two homologs are present in the same specie, they are called paralogs, whereas, if they are in different species, they are ortologs.

The process of tracing evolution involves first identifying suitable families of homologous proteins and then using them to reconstruct evolutionary paths (Lehninger, Nelson, & Cox, 2008). Comparing DNA is based on the fundamental assumption that if two DNA sequences are similar, they probably share the same function, even if they occur in different parts of the genome or across two or more genomes (Keedwell & Narayanan, 2005). On the other hand, protein comparison starts from the premise that the linear sequence determines the tertiary structure which, in turn, determines the function.

There are many algorithms that align and compare two or more sequences (nucleotide/amino acids) or do searches through genetic databases in order to find the sequence's evolutionary relatives or to infer the sequence function based on known ones. BLAST (Altschul et al., 1997) is one of the most widely algorithms used to do homology searches.

In order to extract genetic information from a DNA sample, this molecule has to be partial or completely sequenced. Next Generation Sequencing (NGS) technologies allow sequencing huge volumes of DNA in short periods and at low cost. However, this high throughput methods generates a massive amount of data that should be processed and analysed.

Interpreting the genomic data and assigning it biological meaning is one of the major challenges of the post-genomic era (Gouret et al., 2005). Annotation is the process by which the landscape of genomic DNA is surveyed, and key features of the sequence are described (Pevsner, 2009). The *structural annotation* aims to find genes, their location in the sequence, the intron/exon structure and predict the protein sequences they encode (Gouret et al., 2005), while the *functional annotation* is centred in discovering the biological function of the sequence.

Gene Ontology (GO) (The Gene Ontology Consortium, 2000) was created in order to standardising the representation of the functional annotation across species and databases. This controlled vocabulary of terms has a graph structure and describes the gene biological process, molecular function and cellular component.

The annotation process can be carried out manual or automatically. The first clearly guarantees highest-quality data and most accurate gene structures, but this process is slow and can produce conflicting interpretations of the analysis (Potter et al., 2004). On the other hand, the automated annotation is faster, does not

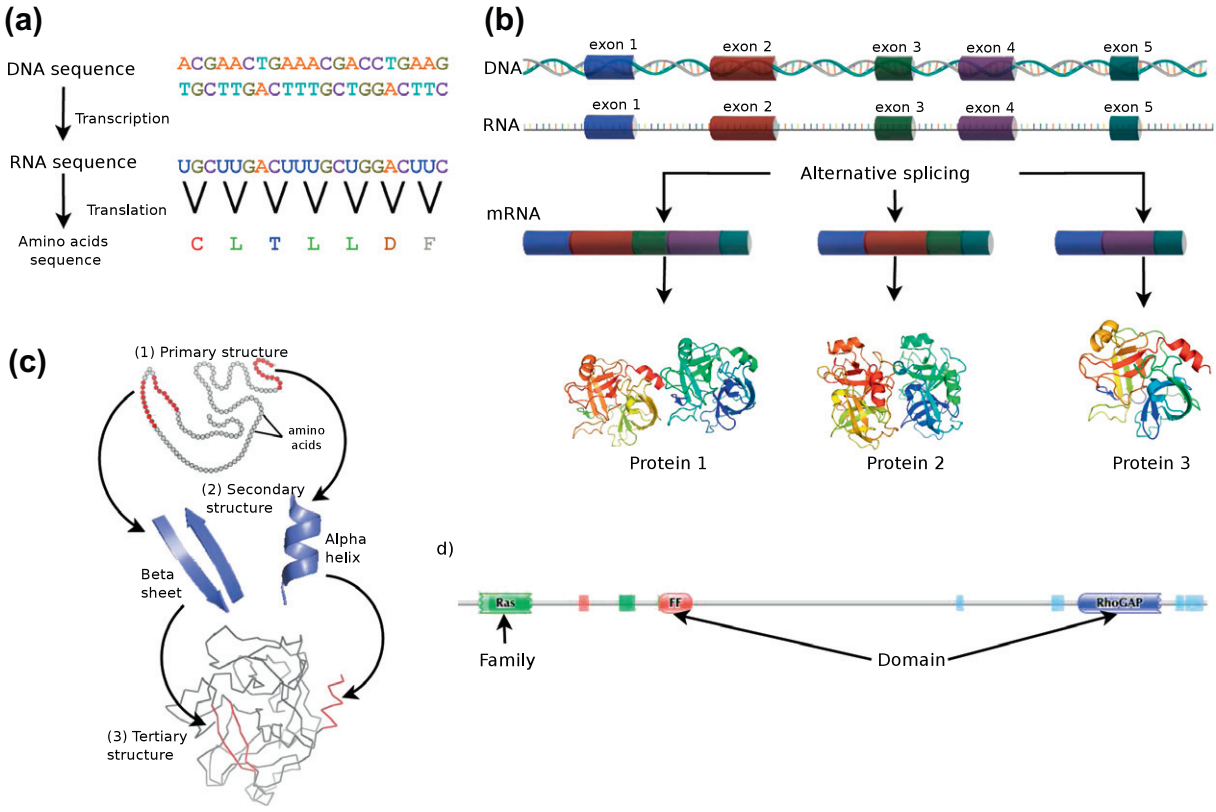

Fig. 1. (a) Simplified central dogma of molecular biology. (b) Alternative splicing schema: one gene can code more than one protein. (c) Protein structure. (d) The protein B2RTN5_MOUSE predicted architecture consists of one family (RAS) and two domains (FF and RhoGAP).

require an expert team, and maintains the consistency of the analysis, but it is less accurate and the results can be less specific. Although, ideally, the annotation should be done by experts who have biological and biochemistry knowledge to infer the features that suit better a certain DNA sequence, the advent of high throughput sequencing methods prevents this practice. Therefore, there is need to develop more accurate automated annotation methods. Moreover, since the functional annotation is mainly carried out through homology techniques, the genetic databases have an important role in this task, as they will be the source of the information for the new annotations. This fact reinforces the necessity of really accurate annotations in order to avoid error propagation.

A great variety of genetic information is publicly available and can aid during the annotation and other processes as well. There are databases for genes and genomes (e.g. GenBank (Benson, Karsch-Mizrachi, Lipman, Ostell, & Sayers, 2011) and Ensembl (Flicek et al., 2011)), proteins (e.g. NCBI Entrez Protein Database (Sayers et al., 2012) and UniProt (UniProt Consortium, 2012)), domains and families (e.g. NCBI's Conserved Domains (CDD) (Marchler-Bauer et al., 2011), InterPro (Hunter et al., 2012), and Pfam (Finn et al., 2010)), gene ontologies (e.g. GO (The Gene Ontology Consortium, 2000)), tertiary structure (e.g. PDB (Bernstein et al., 1977)), and so on. Even though some of these databases are linked to each other through cross-references, in general, each one has its own format and sometimes it is difficult to gather related information.

Currently, some ES annotators, such as Ensembl (Curwen et al., 2004) and FIGENIX (Gouret et al., 2005), have been developed. Nevertheless, these systems have some limitations: the first one was designed to deal with complete genomes, while the second is only available to online execution. Regarding the first issue, there are many scientific researches, such as gene discovery, that do not need to sequence and annotate the whole genome of an organism, as they are focused on specific sequences. Thus, creating an ES capable of annotating DNA sequences without considering the genome context will aid them. This would contribute to enhance the quality of the genomic data uploaded into the public databases and consequently, contribute to improve future annotations. The second issue is related to the fact that online systems oblige users to submit their data to external servers. This is an important drawback, as the annotation frequently works on sensitive information.

3. CommonKADS and knowledge elicitation

3.1. CommonKADS

CommonKADS (Schreiber et al., 2000) is a flexible methodology that offers a set of tools to model KBS regarding not only the knowledge needed but also its context and proposal. Through this methodology, it is possible to identify the strategy that best fits the problem to solve. Apart from that, it establishes the methodological bases to tackle the problem in a general way, allowing these bases to be applied to any similar problem, independently of its complexity. Other benefits of using CommonKADS include improved communication, standardization, technology support, and availability of reusable components (Akerkar & Sajja, 2010).

The CommonKADS process is organised around three analysis activities that specify six models. Fig. 2 summarises it.

In the *Context* analysis, an *Organisation Model* is used to understand the organisational context and environment where the problem lays. This supports the evaluation of the feasibility and benefits of employing a KBS to solve the problem. During this phase, the organisational task layout and the agents that perform theses tasks are also analysed, through a *Task* and an *Agent Model*, respectively.

The second analysis, the *Concept*, is focused on the conceptual description of the knowledge applied in the task. The *Knowledge*

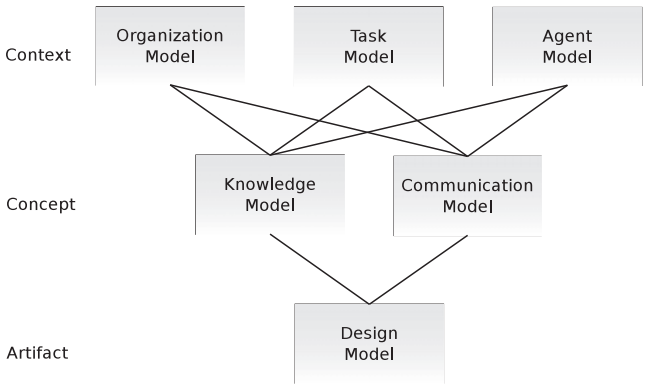

Fig. 2. CommonKADS models (Schreiber et al., 2000).

Model takes into consideration the knowledge and reasoning requirements to develop the system, while the *Communication Model* deals with the communicative transactions between the agents involved.

The main focus of the last analysis, i.e. *Artifact*, is the technical aspects of the computer implementation. This phase culminates in the *Design Model*, where the technical system specifications needed to implement the functions proposed in the knowledge and communication models are specified. The construction of all models is not mandatory. It depends on the goals of the project as well as the experiences obtained during its development.

The current work is centred in describing the knowledge model. Its major goal is analysing and structuring the knowledge required to develop an ES for solving the annotation problem. For this purpose, it performed complete context and concept analyses, and partially the artifact one.

3.2. Knowledge elicitation

Before modelling the knowledge, it is essential to extract it from the sources that contain this information. Knowledge Elicitation (KE) provides a set of techniques to facilitate the acquisition of the material required to structure a more formal description of the problem. It generates organised data such as diagrams, lists, informal rules, and formulae. This task is the main bottleneck of the KBS developing, since this activity demands much time and its results should be accurate and robust. If the knowledge is not correctly obtained or contains errors, the system will be doomed to fail.

Information can be extracted from different kinds of sources related to the domain. Experts are one of the most important sources of knowledge and sometimes they have information that was not previously catalogued. However, this information is structured and stored in a complex cognitive way, what hinders its acquisition (Alonso & Guijarro, 2004). Therefore, to be successful in this task, it should be focused and structured, but also as open as possible. Moreover, it is important to take into account which technique fits better in the problem-solving domain, the type of expert from whom the knowledge will be obtained, and the type of knowledge to be extracted.

Many techniques can be employed, separately or complementarily, to extract knowledge. The interview is the most popular of these techniques and has the benefit that it is a low cost method that does not require any special training from the knowledge engineer. Other advantage is that this technique allows the obtainment of different types of knowledge at distinct levels of the KBS developing process, independently of the application domain. The interview can vary from completely unstructured to formally planned. The unstructured interview has no detailed planning and, consequently, few constrains. In contrast, the structured interview is the formal version of the first. In this case, the knowledge engineer plans and manages the session aiming to extract specific knowledge from the expert.

The type of expert who facilitates the information is crucial for the success of the KE process as well. Aspects such as communication and verbalization skills, and attitude towards the domain have a great influence in the way the knowledge engineer should conduct the information extraction. Experts can be classified as academics, practitioners or samurais (Schreiber et al., 2000). The formers are accustomed to verbalize their knowledge and structure it in a logical way, but they do not apply it in a regular basis. The second kind deals with the problem-solving daily, and since their theoretical knowledge about the subject and the domain is not as deep as that of academic experts, they are used to apply heuristics in their decision process. The latter carry out their job in an automatic way due to their lacking of theoretical training.

The KE method is directly linked to the type of knowledge to be obtained. For that reason, it is essential to apply the technique that best suits the type of information regarding the domain. According to Awad and Ghaziri (2004) the knowledge can be procedural, declarative, semantics or episodic, according to the depth (from shallow to deep) of understanding of the problem area.

Procedural knowledge is the knowledge applied to carry out a specific procedure. It is the most shallow and is related to skills that demand the repetition of the knowledge over and over again, such as psychomotor tasks or learning a language. It is so highly automated that becomes a natural part of the person. Since this knowledge is so rooted in the expert, it is difficult to be verbalized.

On the other hand, the declarative knowledge is the one that could be verbally expressed in a simple way. This type is often in the short-term memory, and for that reason it is related to uncomplicated information that is ready to be recalled. This knowledge is useful for early stages of the KE and it can be properly acquired through structured interviews.

Semantic knowledge is a deeper kind of knowledge, which resides in the long-term memory, and reflects the cognitive structure, representation and organisation of the expert. It includes major concepts, vocabulary, facts, and relationships.

Episodic knowledge is the one based on the experimental episodes the expert faced during her/his life. However its application is automated for the expert, this knowledge is so chunked in the long-term memory that the expert experiment difficulty in remembering and explaining it.

This work aims to comprehend how experts solves the annotation problem and, from this understanding, extract the rules and heuristics used by them during this process. The experts interviewed have an academic/practitioner profile, as they have both theoretical and practical knowledge, and are very familiar with the domain. The type of knowledge to be acquired is mainly procedural, since it is related to how a task is carried out, but it has declarative, semantic, and episodic characteristics as well. This makes its extraction even more complex.

The KE technique employed was the structured interview because it perfectly suits the type of knowledge we intend to obtain. Moreover, this approach produces structured data that are easier to analyse, facilitating the process of extracting usable knowledge.

The interviews held with different experts gave a global vision of the annotation process and helped the engineer to obtain specific rules employed by these specialists. At this stage, the main conclusion is that each expert employs her/his own annotation pipeline, what makes very important to design a flexible annotation model that can be modified according to the expert's knowledge and needs.

Moreover, after comparing the interviews, it is possible to summarise the following findings: (1) different paths can lead to the same solution; (2) BLAST (Altschul et al., 1997) is used by all experts, despite of the pipeline; (3) some output attributes of programs, such as e-value, are key to guide decisions in the annotation process; and, (4) each program consults different databases, for instance, Ensembl's (Flicek et al., 2011) *protein databases* and NCBI's *non-redundant protein database* (NR) are processed by BLASTx, while CDD (Marchler-Bauer et al., 2011) database is handle by RPS-BLAST (Altschul et al., 1997).

4. Knowledge model

The main focus of this work is to organise and structure the knowledge needed to design the ES proposed. Since one of the major challenges of Knowledge Engineering is to discover structures that are capable to model the knowledge in a schematic way, methodologies that are able to accomplish this goal excel in this field.

CommonKADS, as depicted in Fig. 2, supports a Knowledge Model (KM) that facilitates the structure of a knowledge-intensive information-processing task. This model is structured similar to traditional analysis models in Software Engineering. It is composed of three knowledge categories, hierarchically arranged, which are discussed in next sections for the annotation problem (see Fig. 3).

4.1. Domain Knowledge

The Domain Knowledge includes the leaves of the tree (see Fig. 3c), and covers the domain-specific knowledge and the types of information tackled in the application. This category has two basic elements: Domain Schema and Knowledge Base (KB).

The domain schema describes the domain-specific knowledge and information schematically, by means of the association of attributes and values to certain domain instances that share similar features. These instances are called *concepts*, and each one of their attributes requires a *type* (e.g. concepts, relations, and rule types) to specify the values accepted by these attributes. Fig. 4a depicts the concept of sequence and some of its attributes and values. Concepts can be linked through relationships, as the one in Fig. 4b, which indicates that a DNA sequence codes zero or more proteins. Complex dependencies between concepts are represented by logical relationships in rules. Fig. 4c shows an example of a rule related to the analysed domain: *if a sequence is a pseudo-gene then it cannot code a protein*. The domain schema can also contains super/subtypes (see Fig. 4d), and can be described at different levels of abstraction.

In order to understand the scope of the application is important to be able to schematize it and to create a KB containing the instances of knowledge types existents in the Domain Schema. The KB built in this work consist of the information obtained during the KE process together with biological data from public genetics databases, such as GenBank (Benson et al., 2011), CDD (Marchler-Bauer et al., 2011), and Ensembl (Flicek et al., 2011).

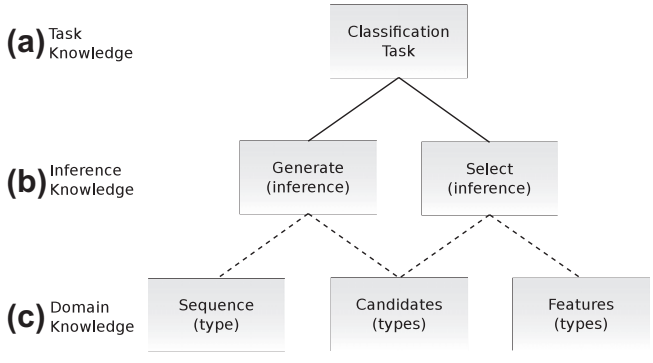

Fig. 3. Knowledge categories schema for a DNA sequence annotation application.

4.2. Inference knowledge

The Inference Knowledge is the middle knowledge category of the tree (see Fig. 3b), and describes the basic steps required to use the Domain Knowledge. This knowledge mainly consists of *inferences*, *knowledge roles*, and *transfer functions*.

Inferences apply the knowledge existent in the KB to deduce new information from an input. This element can be understood as a block that represents the machine reasoning, that is, actions to verify, order, generate, predict, and evaluate. The inference is described as a function of its input and output, without explaining the process employed to generate the output. These input and output, in turn, are structured in terms of the roles they play in the reasoning process. A knowledge role is called dynamic when it varies according to the invocation of the inference, and static if it stays to a certain extent stable during the process. Static roles generally make part of the Domain Knowledge employed to make the inference. In Fig. 4.e, the static information of the role DB (member of the KB) is applied to generate protein candidates to the input sequence.

The transfer function is responsible for the communication with the external world. It transfers information between the reasoning process and the environment that is outside of the system, such as another system or a user. There are four types of transfer function and they vary according with who has the initiative, the system or the exterior, and where the information resides, inside or outside the system. The functions *obtain* and *receive* are the most used in knowledge models (Pajares & Santos, 2005).

Fig. 5 depicts a general vision of the proposed application that can be adapted to different annotation pipelines. The system *receives* the dynamic role *sequence* from the *user* and passes it to a *program*, which using a specific *database* (an static role) *generates candidates* and *selects* the best ones based on *rules* (another static role). The rules change according to the program and the approach used. They are based on attribute features of the output, such as BLAST's e-value or bitscore. The *results* of the rule filtering are represented by a list of data and its respective attributes and values.

The set of inferences *generate*/*select* can be applied for n programs, creating a flexible design that can be adapted to other pipelines. Each one of the n programs runs in parallel using its own database and rules. After the execution of these programs, the content of their output (*Results* in the Fig. 5) is inserted into a database.

It is also possible to run programs that depends on the output of others, as occurs in Fig. 5.a. In this case, the output obtained is employed as input of another program in order to generate complementary data. Even though these data are not essential for the decision process, they can reinforce the annotation inferred, raising its reliability.

Once all programs are finished, the transfer function *present* retrieves *all* [*the*] *results*, and these are compared based on *rules* of the KB. The final result, the dynamic role *protein*, is *presented* by the system to the *user* together with a log file that details all the reasoning process carried out.

Fig. 6 exemplifies the application of the general schema previously proposed to a system that contains three programs ($n = 3$): BLASTx (Altschul et al., 1997) against Ensembl (Flicek et al., 2011) *protein*, BLASTx against NR and RPS-BLAST (Altschul et al., 1997) against CDD (Marchler-Bauer et al., 2011). In this example, the pipeline were designed to annotate data from *Sparus aurata* (sea bream fish), thus a protein database containing all model

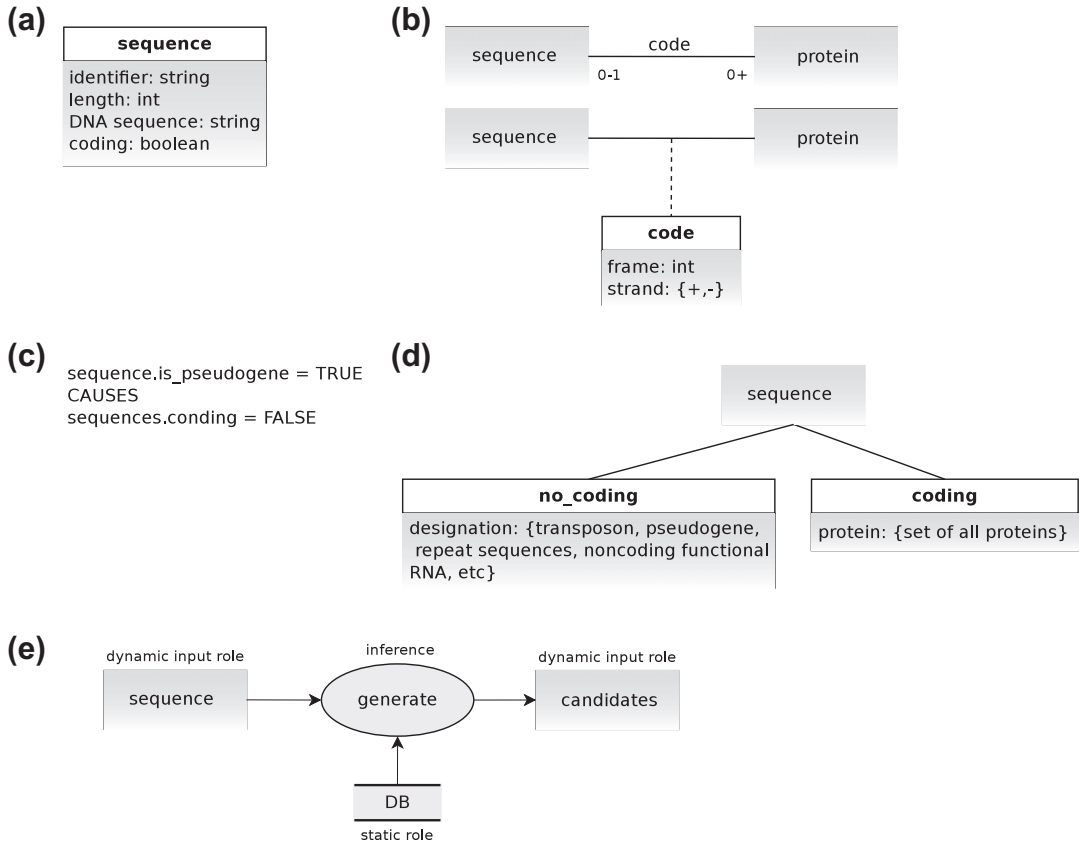

Fig. 4. Different abstractions of the Domain schema: (a) concept, (b) relationship, (c) rule, (d) subtype and, (e) knowledge roles for the inference generate.

fishes from Ensembl were created in order to complement, in a more specif way, the information of the NR database.

Due to the model modularity and the reusability of the approach, it is quite simple to alter the pipeline. Programs and databases can be changed according to the problem to solve and the strategy traced. Moreover, since the rule descriptions also follow a modular structure, it is easy to add or remove rules without causing changes in the main code.

4.3. Task knowledge

The Task Knowledge is located in the root of the knowledge category tree, as shown in Fig. 3a. It is crucial for the system design, as it describes the objectives to be achieved through the knowledge and the strategies to be implemented in order to reach them. This knowledge describes a recursive decomposition of a top-level task in sub-tasks, which in turns are decomposed into simpler subtasks, as depicted in Fig. 7. At the lowest level of this decomposition, the tasks are linked to one or a series of inferences and transfer functions in the Inference Knowledge category.

The main elements of this category are the *task* and the *task method*. The former is related to the question "what can be done?", and defines a complex reasoning goal in terms of input and output roles. Each task has a task method that describes how this task is carried out through its decomposition into sub-functions. The task method answers the question "how can it be done?", and is composed of a set of sub-functions and a control structure that specifies the order in which these functions are carried out.

When modelling the task, a straightforward approach is to base it upon a generic task pattern from CommonKADS Template Knowledge Models library (Schreiber et al., 2000). These templates specify the steps to be taken as part of a specific task, preventing the engineer from "reinventing the wheel". They have proved to be useful in developing a range of common systems for many projects (Schreiber et al., 2000). The templates can be used in different domains without significant changes and be reused regardless of the application, what is one of the greatest advantages of this methodology.

Task templates are divided according to the system the task operates on. They can be analytic, if representing an existing system that is not completely characterised. In this case, the objective of the task is producing more information about the system. In contrast, the synthetic approach is used when the system does not exist and the purpose is constructing its first description.

Since the goal of this work is to describe a system that is not completely known, the analytical approach is the one that fits best. This category specifies tasks such as classification, diagnosis, assessment, monitoring, and prediction.

The task of the functional annotation of DNA sequences can be viewed in an objective way as finding the protein "class" that best characterises a certain sequence. This premise fits perfectly the well-known analytical task of classification. This task usually involves objects from the nature, such as animals and plants, and its objective is to discover the association between the features of an object and a class from a predefined set of classes.

Each generic task template has four basic elements that together facilitate the task specification: the *general characterisation*, the *default method*, the *variation method*, and the *typical schema of domain*.

The first, the general characterisation, describes the features of the typical task. This part defines the *objective* of the task. It also specifies the *terminology* applied to describe the object to be classified in function of its *class*, *attributes*, and *features*. The object *input*, the class *output*, and a typical *example* are given as well. Table 1 shows an example of this characterisation.

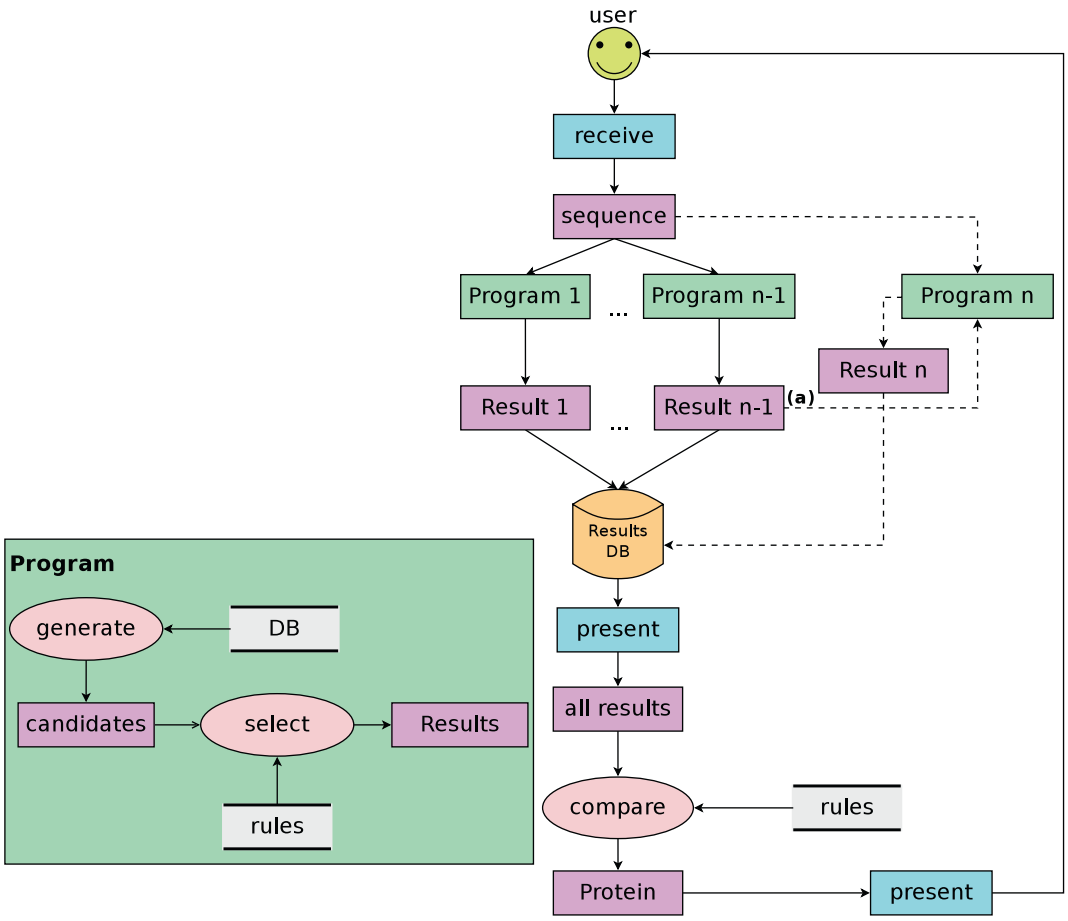

Fig. 5. The general schema for the application: inferences, knowledge roles and transfer functions.

The default method represents the first decision to be made. It can be data-driven, when the object's initial features are used to generate a set of candidate solutions, or solution-driven, if the starting point is the set of all possible solutions and this set is reduced based on the information acquired. In the case that the default method cannot be used, method variations can be applied to reach the solution.

In this system, the candidate solution set is obtained from the initial data through alignments against different databases, therefore the default method is data-driven. Moreover, all input objects can be processed by the default method, so there is no need to use method variations. The schema for the sequence classification task and its inference structure is very similar to the one depicted in Fig. 5, but without the static roles.

5. Analysis and implementation

The system for functional annotation will be a rule-based ES according to the schema proposed in Fig. 6. The previous sections describe the acquisition of the knowledge about the domain of the application, and model the generic task to be carried out by that system. Its rules were obtained through interviews held with experts, and complemented with the previous knowledge in the domain of the engineer. The development of the system also requires specifying additional requirements such as its execution environment and user interface, and defining some basic components of an ES (Giarratano & Riley, 1998), such as its KB, working memory, and inference engine.

The ES will be first developed for Ubuntu 10.04 and then extended to other operating systems. The user interface will be text-based and graphical as well, in order to be as much flexible and friendly as possible.

The system's KB comprises external databases from different public sources and a collection of rules compiled based on the knowledge of the experts. These rules will be described according to the production rule system Drools (The JBoss Drools Team, 2012). Table 2 describes the KB elements and their respective formats.

The active memory, where the input and intermediate data are kept, will be stored in a physical database (see Fig. 6.a). If needed, at the end of the annotation process the user can dump these data to further analysis.

Since Drools will be used as the rule management system, the main inference engine will have a forward chaining approach. This engine, represented by the inference *compare* (see Fig. 6b), is fed by the KB and its own agenda, and it infers the final annotation, i.e. *Protein*. Apart from that, each program will have its own inference engine in order to achieve the best results.

One advantage of rule systems, such as Drools, is that they already provide explanation facilities. In this application, the reasoning process will be summarised in a log file, which will be available for the user under request.

In a first version, the knowledge acquisition facility will not be implemented, as this implies a relevant effort on interface design and validation. Nevertheless, it could be easily added to the system in the future thanks to its modularity.

The overall system has been designed as a collection of pipelines that can be carried out in parallel. Its single input is a DNA sequence file in FASTA format, as shown in the example of Table 1. After the execution of all the pipelines, the results are gathered

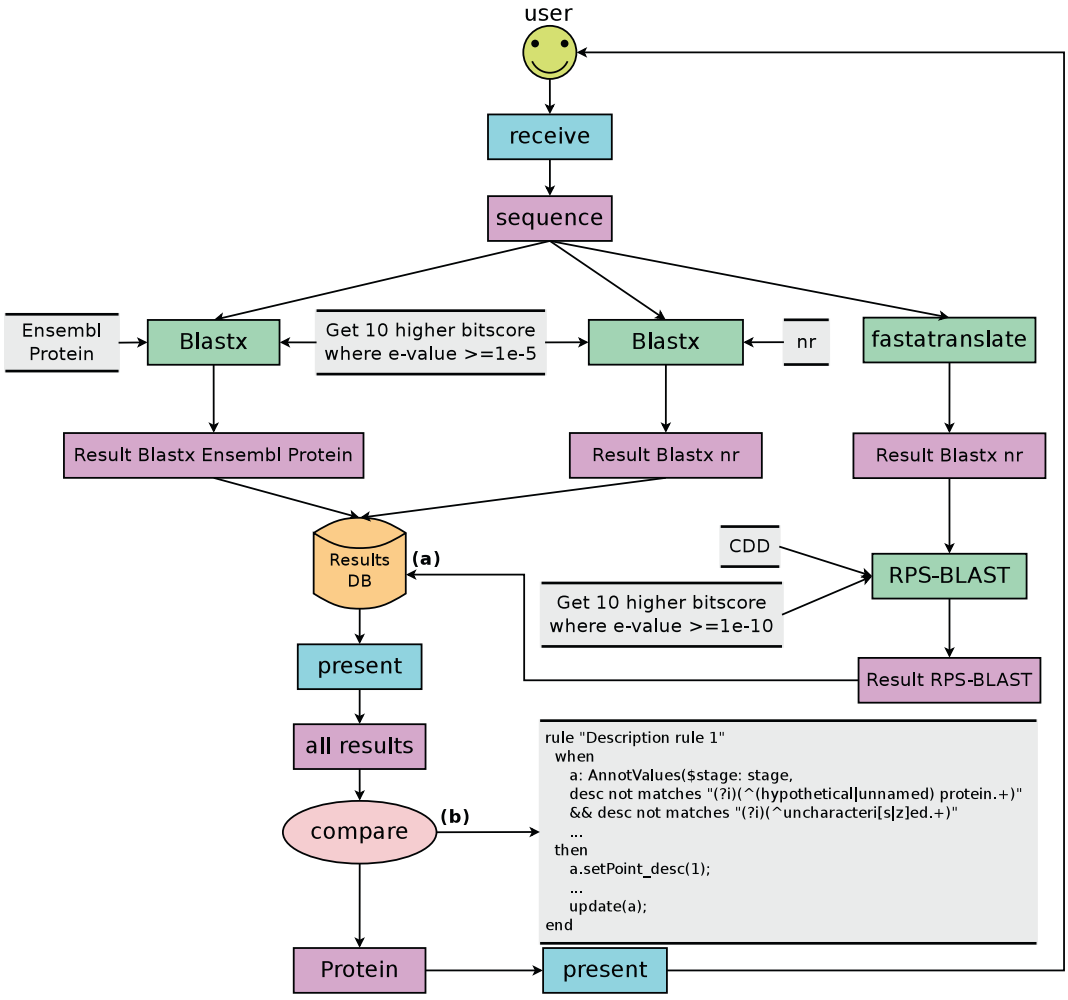

Fig. 6. General schema applied to a concrete pipeline: BLASTx against Ensembl Protein and NR databases and RPS-BLAST against CDD.

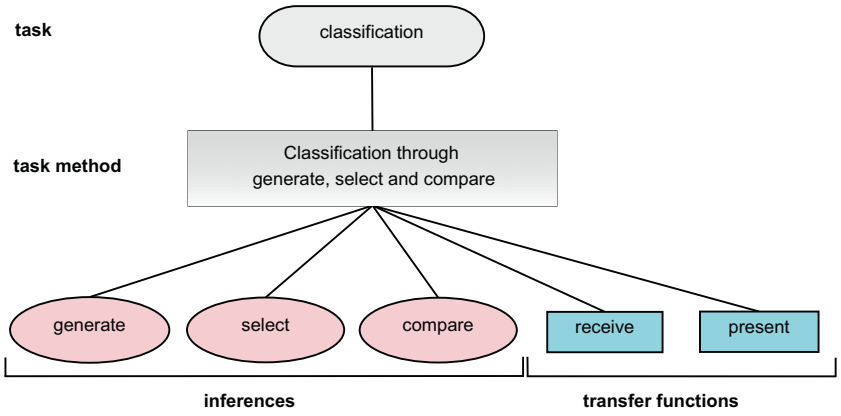

Fig. 7. Diagram for decomposition of the classification task.

and analysed, generating a GFF (Durbin, Haussler, Stein, Lewis, & Krog, 2011) file, an annotated FASTA file, and a reasoning trace file.

The implementation will be mainly in Java due to its communication with Drools, its portability and its facilities to build user interfaces. At first, some reused Perl scripts will complement this implementation, but they may be replaced by Java code in the future.

Perl scripts will be used in the implementation of the inference select, during the *Program* stage (see Fig. 5). BioPerl (Stajich et al., 2002) module Bio::SearchIO, which handles Bioinformatics formats such as the BLAST output, aids the processing and parsing of this inference outcome, facilitating the application of filtering rules. The selected information will be then written in a intermediary file, through Bio::Tools::GFF module, and finally loaded into the Results DB by *bp_load_gff.pl*, a Perl script from the GBrowse (Stein et al., 2002) packge that uses the module Bio::DB::GFF.

Java framework for handling biological data, BioJava (Holland et al., 2008), and its extension, BioJavax, will be employed as well.

Table 1
General characterisation for the system classification task.

Objective	Infer the protein corresponding to a given sequence. This classification is based on values and features obtained through local alignments of the object sequence with sequences from different databases
Example	>gi|34783212 Homo sapiens mitochondrial ribosomal protein S33 AGGAACTCTAACTCGTTGCCACTCAAGAAATGTCGTCCGTTTCAGAATATGGCTTCCGCATGTCTGGTCT CAGTGCCCGGCTATTTGGTGAAGTCACCAGGCCTACTAATTCCAAGTCTATGAAAGTGGTGAAACTGTTT AGTGAACTGCCCTTGGCCAAGAAGAAGGAGACTTATGATTGGTATCCAAATCACCACACTTACGCTGAAC TCATGCAGACGCTCCGATTTCTTGGACTCTACAGAGATGAGCATCAAGGATTTTATGGATGAGCAAAAACG ACTAAAGAAGCTTCGTGGAAAGGAGAAACCAAAGAAAGGAAGAGGCAGCAAAAGGAAATAG TGTTGGTCCGTCAAGAGGGAGACTTTCTTCGTCAGTGGCGGAGAAGAAAGTGCATTTATTGTCTTTCC ACATATTGGAGGAATGTCATCTTCCTAAATGAAGTTTATTTGGAGGAACACAGTCATCTCCTTGGTGAAA TCTAATCCGGTTACATTGTGGCTGGTTTCTTGAACACATTCTAACTGTGCAAAATTATCGTTGGCCTTGGC CGTGTAATGTGAGGTTTACCTGATTCTCTAATGAAATAAATACCTAAGTTATTGTTTGAAAAAAAAAAA AAA
Terminology	Object: sequence object Class: an element of the class protein Attributes: identifier, nucleotide sequence, sequence length
Features	• The identifier is a string of alphanumeric characters that can also contain:"|","_" and "." characters. In the example: gi|34783212 • The nucleotide sequence is a string composed only of "A", "C", "G", "T" and "N" characters. • The sequence length is a natural number that shows the number of base pairs in the sequence. In the example: 563 bp
Input	DNA sequence
Output	A specific protein

Table 2
Knowledge elements for the system proposed in Fig. 6.

KB element	Description	Format
NR	Non-redundant protein database	Binary indexed files, created by formatdb algorithm (Altschul et al., 1997) for protein databases
CDD	Conserved domains database	
Ensembl	Model fishes database	
Rules	Collection of rules	Drools

The library *org.biojavax.bio.seq* aims to describe the rich implementations of sequences, locations and features, whereas, *org.biojava.bio.program.gff* is responsible for GFF manipulation. Since these frameworks are very powerful and can provide many functionalities, others libraries will be added in future updates.

6. Conclusion and future works

The technological advances in the last ten years have increased the volume of biological data generated all over the world. Nowadays, the challenge is how to process, analyse, store and use these data in an efficient way. A key example of this situation is the gene annotation process, one of the most difficult tasks of Genomics, which has to identify and characterise the genes in a DNA sequence.

Ideally, the gene annotation process should be carried out by human experts, who evaluate each sequence and, based on their biological and biochemistry knowledge, infers its proper features. However, the huge amount of biological data makes of human intervention a bottleneck in this task. The automatic annotation alleviates this bottleneck, but it is less accurate and reliable than the manual approach. Moreover, it still requires human experts to validate (i.e. *curate*) its results given the potential impact of any misprediction in future research and its applications.

One possible solution is the creation of ES that are capable to emulate the expert reasoning in key steps of the annotation process. The development of ES is a complex task and requires extensive knowledge of the application domain and the expert reasoning process. Nevertheless, there are methodologies for modelling ES, such as CommonKADS, which facilitate the development and can be employed in different scopes.

This work proposes a knowledge model for functional annotation system regardless the DNA sequence context and the structured interview as KE technique. The development of the knowledge model applied the CommonKADS methodology. Through this methodology, a general framework was created, using the classification task as a cornerstone. Due to the high generalisation level achieved, the model designed here can be applied for different kinds of pipelines and any DNA functional annotation task.

Thanks to the efforts carried out in this work, it was possible to verify that the use of CommonKADS can be extended to different bioinformatics problems, such as the classification of sequences from metagenomic data or the prediction of protein three-dimensional structure.

This work also intends to highlight the importance of the application of Knowledge Engineering methodologies in systems development for Bioinformatics. These techniques provide the required tools to organise and structure the knowledge in a way it can be understood by all the people involved in the systems development. The KE techniques have a great value as well. They are a powerful mean to obtain the information that is essential for the comprehension of the applications domain.

With the purpose of evaluating the accuracy of the system, 624 annotated sequences of *Sparus aurata* (sea bream) were retrieved

from Entrez Protein Database (Sayers et al., 2012), translated to nucleotide (through *backtranseq* from Emboss package (Rice, Longden, & Bleasby, 2000)) and processed by the pipeline proposed in Fig. 6. Since NR contains all non-redundant protein data from Entrez Protein and the sequences used to test the performance of the system are from the same source, all sequences from this specie were removed from NR database used in this pipeline. This measure prevented the possibility to have the same sequence in both sets (query and hit) and consequently, guarantees a more realistic result. Ensembl (Flicek et al., 2011) fishes database used in this test has not any sequence from sea bream, and therefore it was not modified. Then, each annotation inferred by the system was compared with the annotation in Entrez. The implemented prototype annotated satisfactorily more than 70% of the sequences.

The work discussed in this paper is part of an ongoing work with several open lines for improvement. Even though the prototype got a high percentage of correct annotations, the number of involved rules and programs still need to be increased in order to create more accurate inferences. As part of the future work, we also intend to extend the implementation of this system to a multi-agent system and model it using MAS-CommonKADS (Iglesias, Garijo, González, & Velasco, 1997). Finally, it is also interesting to get a higher involvement of annotation experts in the development of the system, in particular in the refinement of the knowledge the system uses. Here, improvements in the tools to explain the actual annotation process followed by the system for a sequence, and the interactive participation of experts during the process, can be of great help.

Acknowledgements

This work has been supported by Grant CSD2007-00002 Consolider-Ingenio 2010, MICINN (Spain).

Thanks are due to the anonymous referees for their very valuable comments and suggestions.

References

Akerkar, R., & Sajja, P. (2010). *Knowledge-based systems*. Jones & Bartlett Learning.

Alonso, A., & Guijarro, B. (2004). *Ingeniería del conocimiento. Aspectos Metodológicos* (1 ed.,). Pearson Prentice Hall.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., et al. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research, 25*, 3389–3402.

Awad, E. M., & Ghaziri, H. M. (2004). *Knowledge management*. Pearson Education India.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2011). GenBank. *Nucleic Acids Research, 39*, 32–37.

Berger, B., & Leighton, T. (1998). Protein folding in the hydrophobic–hydrophilic (HP) is NP-complete. In *Proceedings of the second annual international conference on computational molecular biology. RECOMB '98* (pp. 30–39). New York, NY, USA: ACM.

Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Brice, M. D., Rodgers, J. R., et al. (1977). *European Journal of Biochemistry, 80*, 319–324.

Burton, A., Shadbolt, N., Rugg, G., & Hedgecock, A. (1990). The efficacy of knowledge elicitation techniques: A comparison across domains and levels of expertise. *Knowledge Acquisition, 2*, 167–178.

Crescenzi, P., Goldman, D., Papadimitriou, C. H., Piccolboni, A., & Yannakakis, M. (1998). On the complexity of protein folding. In *Proceedings of the second annual international conference on computational molecular biology. RECOMB '98* (pp. 61–62). New York, NY, USA: ACM.

Curwen, V., Eyras, E., Andrews, T. D., Clarke, L., Mongin, E., Searle, S. M., et al. (2004). The ensembl automatic gene annotation system. *Genome Research, 14*, 942–950.

Durbin, R., Haussler, D., Stein, L., Lewis, S., & Krog, A. (2011). *GFF (general feature format) specifications document*. <http://www.sanger.ac.uk/resources/software/gff/spec.html>.

Edwards, D., Stajich, J. E., & Hansen, D. (2009). *Bioinformatics tools and applications*. Springer.

Finn, R., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J., et al. (2010). The Pfam protein families database. *Nucleic Acids Research, 38*, D211–D222.

Flicek, P., Amode, M. R., Barrell, D., Beal, K., Brent, S., Chen, Y., et al. (2011). Ensembl 2011. *Nucleic Acids Research, 39*, D800–D806.

Friedberg, I. (2006). Automated protein function prediction – The genomic challenge. *Briefings in Bioinformatics, 7*, 225–242.

Giarratano, J. C., & Riley, G. (1998). *Expert systems: Principles and programming. Computer science series*. PWS Publishing Company.

Gouret, P., Vitiello, V., Balandraud, N., Gilles, A., Pontarotti, P., & Danchin, E. G. (2005). Figenix: Intelligent automation of genomic annotation: Expertise integration in a new software platform. *BMC Bioinformatics, 6*, 198.

Holland, R. C. G., Down, T. A., Pocock, M., Prlic, A., Huen, D., James, K., et al. (2008). Biojava: An open-source framework for bioinformatics. *Bioinformatics, 24*, 2096–2097.

Hunter, S., Jones, P., Mitchell, A., Apweiler, R., Attwood, T. K., Bateman, A., et al. (2012). InterPro in 2011: New developments in the family and domain prediction database. *Nucleic Acids Research, 40*, D306–D312.

Iglesias, C. A., Garijo, M., González, J. C., Velasco, J. R. (1997). *MAS-CommonKADS: A comprehensive agent-oriented methodology* (an extended version of this paper has been published in the Journal Mathematical Modelling and Scientific Computing, 8).

Keedwell, E., & Narayanan, A. (2005). *Intelligent bioinformatics: The application of artificial intelligence techniques to bioinformatics problems*. Wiley.

Lehninger, A., Nelson, D. L., & Cox, M. M. (2008). *Lehninger principles of biochemistry* (5th ed.). W.H. Freeman.

Marchler-Bauer, A., Lu, S., Anderson, J., Chitsaz, F., Derbyshire, M., Deweese-Scott, C., et al. (2011). Cdd: A conserved domain database for the functional annotation of proteins. *Nucleic Acids Research, 39*, 225–229.

Pajares, G., & Santos, M. (2005). Inteligencia Artificial e Ingeniería del Conocimiento, Ra-Ma, Librería y Editorial Microinformática.

Pevsner, J. (2009). *Bioinformatics and functional genomics*. Wiley Blackwell.

Potter, S. C., Clarke, L., Curwen, V., Keenan, S., Mongin, E., Searle, S. M., et al. (2004). The ensembl analysis pipeline. *Genome Research, 14*, 934–941.

Rice, I., Longden, P., & Bleasby, A. (2000). Emboss: The european molecular biology open software suite. *Trends in Genetics, 16*, 276–277.

Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., Bryant, S. H., Canese, K., et al. (2012). Database resources of the national center for biotechnology information. *Nucleic Acids Research, 40*, 13–25.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Walter, V. d. V., et al. (2000). *Knowledge engineering and management: The CommonKADS methodology*. Cambridge, MA: MIT Press.

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., et al. (2002). The bioperl toolkit: Perl modules for the life sciences. *Genome Research, 12*, 1611–1618.

Stein, L. D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., et al. (2002). The generic genome browser: A building block for a model organism system database. *Genome Research, 12*, 1599–1610.

The Gene Ontology Consortium, 2000. Gene ontology: Tool for the unification of biology. *Nature Genetics 25*, 25–29.

The JBoss Drools Team. Drools expert user guide, 2012.

UniProt Consortium (2012). Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Research 40*, D71–D75.