Universidade Estadual de Campinas
Instituto de Computação

Juan Sebastián Beleño Díaz

WorkflowHunt: a hybrid search mechanism for scientific workflow repositories

WorkflowHunt: um mecanismo de busca híbrida para repositórios de workflows científicos

CAMPINAS
2018

Juan Sebastián Beleño Díaz

WorkflowHunt: a hybrid search mechanism for scientific workflow repositories

WorkflowHunt: um mecanismo de busca híbrida para repositórios de workflows científicos

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientadora: Profa. Dra. Claudia Maria Bauzer Medeiros

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES
ORCID: https://orcid.org/0000-0001-8252-742

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

B411w Beleño Díaz, Juan Sebastián, 1992-
WorkflowHunt : a hybrid search mechanism for scientific workflow repositories / Juan Sebastián Beleño Díaz. – Campinas, SP : [s.n.], 2018.

Orientador: Claudia Maria Bauzer Medeiros.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Fluxo de trabalho - Processamento de dados. 2. Sistemas de recuperação da informação. 3. Ontologias (Recuperação da informação). I. Medeiros, Claudia Maria Bauzer, 1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: WorkflowHunt : um mecanismo de busca híbrida para repositórios de workflows científicos
Palavras-chave em inglês:
Workflow - Data processing
Information storage and retrieval systems
Ontologies (Information retrieval)
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Claudia Maria Bauzer Medeiros [Orientador]
Benilton de Sa Carvalho
Julio Cesar Dos Reis
Data de defesa: 03-05-2018
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Juan Sebastián Beleño Díaz

WorkflowHunt: a hybrid search mechanism for scientific workflow repositories

WorkflowHunt: um mecanismo de busca híbrida para repositórios de workflows científicos

Campinas, 03 de maio de 2018

Acknowledgements

I would like to thank my advisor, professor Claudia Bauzer Medeiros for her support in the development of my path in academia.

I would like to thank my family for all the support that they have provided to me.

I would like to thank members of LIS (including professors André and Julio) for helping me to enjoy my time in Brazil. I will always remember the parties that ended up producing GIFs.

I would like to thank professor Cid Carvalho de Souza. Although my grade in the subject of Computational Complexity was just a C, I learned a lot in his classes.

I would like to thank people at my job in Evoluservices for teaching me good practices in software development and sharing good moments in Happy Hours.

This research was partially supported by Brazilian financing agencies CNPq, CAPES (07/2016 - 1629106) and FAPESP CCES (2013/08293-7).

Resumo

Os experimentos científicos e os conjuntos de dados gerados a partir deles estão crescendo em tamanho e complexidade. Os cientistas estão enfrentando dificuldades para compartilhar esses recursos e permitir a reprodutibilidade do experimento. Algumas iniciativas surgiram para tentar resolver esse problema. Uma delas envolve o uso de *workflows científicos* para representar a execução de experimentos científicos. Existe um número crescente de *workflows* que são potencialmente relevantes para mais de um domínio científico. Criar um *workflow* leva tempo e recursos e sua reutilização ajuda aos cientistas a criar novos *workflows* de forma mais rápida e confiável. No entanto, é difícil encontrar *workflows* adequados para reutilização. Geralmente, os repositórios de *workflows* possuem mecanismos de busca com muitas limitações, o que afeta negativamente a descoberta de *workflows* relevantes para um cientista ou seu time. Esta dissertação apresenta WorkflowHunt, uma arquitetura híbrida para busca e descoberta de *workflows* em repositórios genéricos, combinando busca baseada em palavras-chave e busca semântica para encontrar *workflows* relevantes usando diferentes métodos de busca. Ao contrário da maioria das pesquisas correlatas, nossa proposta e sua implementação são genéricas. Nosso sistema de indexação e anotação é automático e independe de domínio ou ontologia específica. A arquitetura foi validada por meio de um protótipo que usa *workflows* e metadados reais do myExperiment, um dos maiores repositórios de workflows científicos. Nosso sistema também compara seus resultados com o mecanismo de busca do myExperiment para analisar em que casos um sistema supera o outro.

Abstract

Scientific experiments and the datasets generated from them are growing in size and complexity. Scientists are facing difficulties to share those resources in a way that allows reproducibility of the experiment. Some initiatives have emerged to try to solve this problem. One of them involves the use of scientific workflows to represent and enact the execution of scientific experiments. There is an increasing number of workflows that are potentially relevant for more than one scientific domain. Creating a workflow takes time and resources, and their reuse helps scientists to build new workflows faster and in a more reliable way. However, it is hard to find workflows suitable for reuse for an experiment. Usually, workflow repositories have search mechanisms with many limitations, which affects negatively the discovery of relevant workflows. This dissertation presents WorkflowHunt, a hybrid architecture for workflow search and discovery for generic repositories, which combines keyword and semantic search to find relevant workflows using different search methods. Unlike most related work, our proposal and its implementation are generic. Our indexing and annotation mechanism are automatic and not restricted to a specific domain or ontology. We validated our architecture creating a prototype that uses real workflows and metadata from myExperiment, one of the largest online scientific workflow repositories. Our system also compares its results with myExperiment's search engine to analyze in which cases one retrieval system outperforms the other.

List of Figures

2.1 Graph-based representation of a ontology. 14
2.2 Hierarchical representation of a ontology. 15
2.3 Workflow structure and components. 18
2.4 Query "chromosomes". 23
2.5 Query "chromosome". 23

3.1 WorkflowHunt Architecture. 28
3.2 Example of Keyword Index. 35
3.3 Example of Hybrid Text. 36
3.4 Example of Hybrid Index. 37

4.1 Database Schema for the Prototype. 40
4.2 Home Page of WorkflowHunt. 42
4.3 Keyword search results. 43
4.4 Hybrid search results. 43
4.5 Workflow metadata in myExperiment. 44
4.6 Semantic annotations for a workflow in WorkflowHunt. 45
4.7 Draft of a Venn diagram comparing different search approaches for the query "q". 46
4.8 Venn diagram comparing different search approaches for the query "chromosomes". 47
4.9 Venn diagram comparing different search approaches for the query "ecology". 48
4.10 Venn diagram comparing different search approaches for the query "pathway simulation". 49

List of Tables

2.1 Related work comparison 25

Contents

1 Introduction **12**

2 Related Work and Basic Concepts **14**
- 2.1 Ontologies . . . 14
- 2.2 Semantic Annotations . . . 16
- 2.3 Workflows . . . 17
- 2.4 Workflow Repositories . . . 18
  - 2.4.1 MyExperiment . . . 19
  - 2.4.2 CrowdLabs . . . 20
  - 2.4.3 Galaxy . . . 20
  - 2.4.4 CloudFlows . . . 21
  - 2.4.5 PBase . . . 21
  - 2.4.6 OPMW Workflow Repository . . . 22
- 2.5 Workflow Retrieval . . . 22
  - 2.5.1 Keyword-based Search . . . 22
  - 2.5.2 Structure-based Search . . . 24
  - 2.5.3 Semantic-based Search . . . 24
- 2.6 Final Remarks . . . 25

3 Architecture of WorkflowHunt **27**
- 3.1 Architecture . . . 27
- 3.2 Persistence Layer . . . 28
- 3.3 Pre-processing Layer . . . 29
  - 3.3.1 Semantic Annotations . . . 29
  - 3.3.2 Indices . . . 33
- 3.4 Search Layer . . . 38
- 3.5 Final Remarks . . . 38

4 Implementation Aspects **39**
- 4.1 Implementing the Persistence Layer . . . 39
- 4.2 Implementing the Preprocessing Layer . . . 39
- 4.3 Implementing the Search Layer . . . 41
- 4.4 User Interface . . . 42
- 4.5 Case Study . . . 44
- 4.6 Final Remarks . . . 49

5 Conclusions and Future Work **51**
- 5.1 Conclusions . . . 51
- 5.2 Future Work . . . 52

Chapter 1

Introduction

There is a reproducibility crisis in science [37]. The complexity of handling datasets is growing exponentially and there is a lack of tools to allow reproducibility and communication of data analysis. Some efforts are aimed at solving these problems, e.g., The Open Science Movement [24]. One particular approach involves the use of scientific workflows.

A *scientific workflow* is a step-by-step description of a scientific process for achieving a scientific objective using elements as inputs, outputs, and tasks [30]. Usually, scientists publish and share workflows with their peers in academia and industry via workflow repositories. These repositories contain workflows and metadata associated with them.

There is an increasing number of scientific workflows that are potentially relevant to one or more scientific domains [2]. The access to those workflows is usually open to anyone interested in their reuse, re-purpose, or experiment replication. Given such availability of workflows, how to find and choose the one(s) more suitable for reuse by a given experiment?

The problem handled in this work is the search and discovery of relevant workflows according to a scientist's needs. This problem is important because creating new workflows can be expensive in terms of time and resources. Reuse of relevant workflows helps researchers to build new workflows in a faster and more reliable way.

There are mainly three approaches for search in scientific workflow repositories: keyword-based search, structure-based search, and semantic-based search [2, 40, 39, 6]. Keyword-based search uses the workflow metadata to return workflows associated with the exact words used by scientists in their queries. Structure-based search uses the workflow topology to find workflows with similar structure. Finally, semantic-based search uses semantic annotations to find workflows where annotations are similar in meaning to the scientist's query. Ideally, search mechanisms should combine these approaches, but they are seldom supported due to the technical challenges of balancing advantages and disadvantages of different search approaches.

For example, keyword-based search is easy to use because interfaces and functionality are similar to modern search engines like Google. Nevertheless, this is a limited approach because of the heterogeneity in the terms that are used to refer to the same scientific concept. This causes problems when a scientist wants to search for scientific workflows related to a scientific concept, but s/he knows just a subset of the terminology used to refer to that concept and gets partially relevant results. Structure-based search is computationally complex, and hard to perform with heterogeneous sources. While semantic-based search presents more advantages, it often requires knowing languages with complex syntax like SPARQL, a graph query language [27].

This work presents the design and implementation of WorkflowHunt: a retrieval system for scientific workflow repositories, which uses keyword-based search and a hybrid-based search (keyword and semantic search) to find relevant results in scientific workflow repositories. This work was validated with data from myExperiment [11], which is one of the largest scientific workflow repositories at the moment. Moreover, EDAM and CHEMINF ontologies were used to create semantic annotations that works as inputs for the hybrid search. EDAM is an ontology of bioinformatics information, including operations, types of data, topics, and formats [25]. CHEMINF is an ontology for chemical information, including terms, and algorithms used in chemistry [22].

The main contributions of our work are:

- A study comparing different scientific workflow repositories and their retrieval systems;
- The design of an architecture for workflow retrieval combining keyword and semantic search;
- The implementation of a prototype for WorkflowHunt;
- A case study with detailed comparison of the retrieval capabilities of WorkflowHunt and myExperiment search mechanism

From this work, we published a paper, parts of which are reproduced in this text:

- J. S. B. Diaz and C. B. Medeiros, "WorkflowHunt: Combining Keyword and Semantic Search in Scientific Workflow Repositories," *2017 IEEE 13th International Conference on e-Science (e-Science)*, Auckland, 2017, pp. 138-147. doi: 10.1109/e-Science.2017.26 [12]

The rest of this work is organized as follows: Chapter 2 presents the theoretical foundations and related work; Chapter 3 presents the architecture of WorkflowHunt; Chapter 4 presents details about the implementation and a case study. Finally, Chapter 5 presents the conclusions and future work.

Chapter 2

Related Work and Basic Concepts

This chapter shows general concepts needed to understand semantic retrieval systems for workflow repositories such as ontologies (Section 2.1), semantic annotations (Section 2.2), and workflows (Section 2.3) . Sections 2.4 and 2.5 describe some retrieval systems for scientific workflow repositories.

2.1 Ontologies

An ontology is “an explicit specification of a conceptualization" [21], where a conceptualization is an abstract perspective about something (e.g., objects, concepts, and relationships). An explicit specification means the definition of a formalism to map the abstract meaning of the conceptualization into something concrete. Ontologies are expressed and stored in many ways, always associated with some sort of mechanism to allow inferences.

Figures 2.1 and 2.2 respectively show a graph-based and a hierarchical representation of a small subset of the EDAM ontology. EDAM is an ontology of bioinformatics information, including operations, types of data, topics, and formats [25]. These figures were generated via Ontology Lookup Service [8], a service that allows querying, browsing, and navigating over a database that integrates several biomedical ontologies and related vocabulary.

[1]http://www.ebi.ac.uk/ols/ontologies/edam/terms/graph?iri=http://edamontology.org/topic_0654

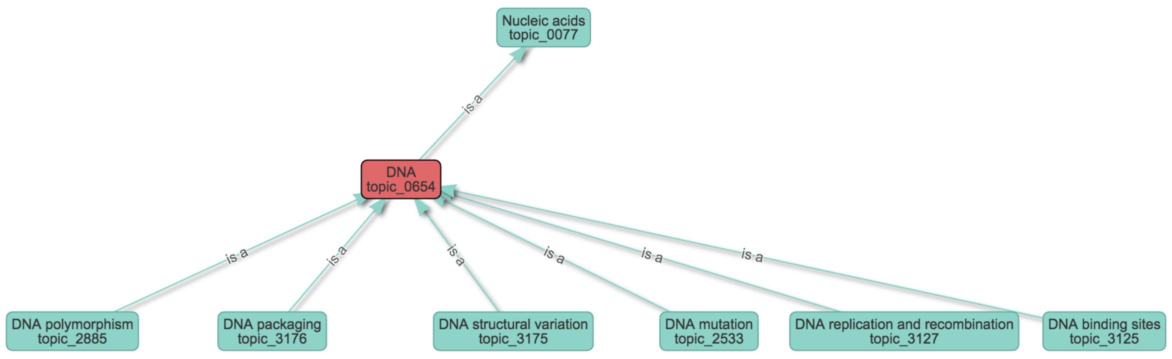


Figure 2.1: Graph-based representation of a subset of the EDAM Ontology[1].

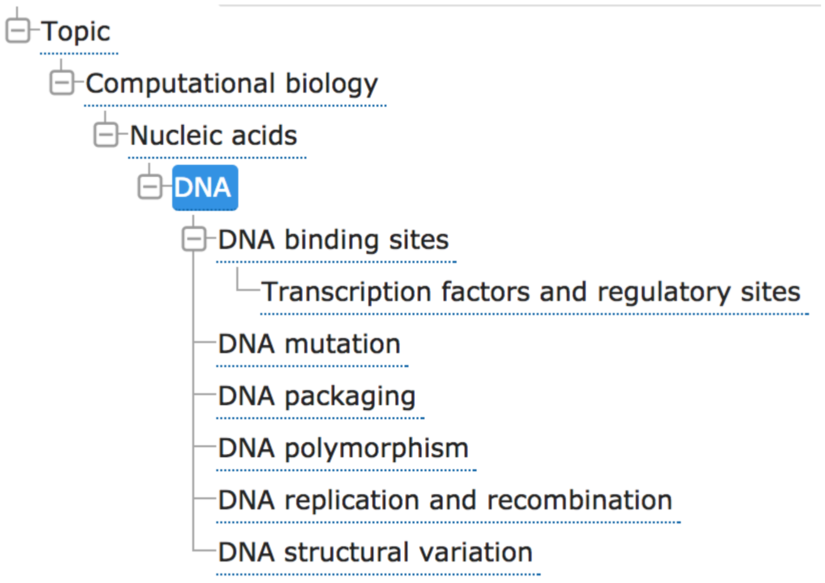


Figure 2.2: Hierarchical representation of a subset of the EDAM Ontology[2].

Classes are key components in an ontology. An **ontology class** represents an object, concept or category that belongs to a domain [10]. Usually, classes are connected through relationships defined in the ontology. For example, in Figure 2.1, square boxes represent classes and arrows represent relationships. The kind of relationship between two classes is described by the orientation of the arrow and a label. Although the hierarchical representation seems easier for navigating among ontology classes, it does not show relationship labels, which is a big barrier to fully understand an ontology. Hence, we decided to focus on the graph-based representation, which allows us to introduce two concepts:

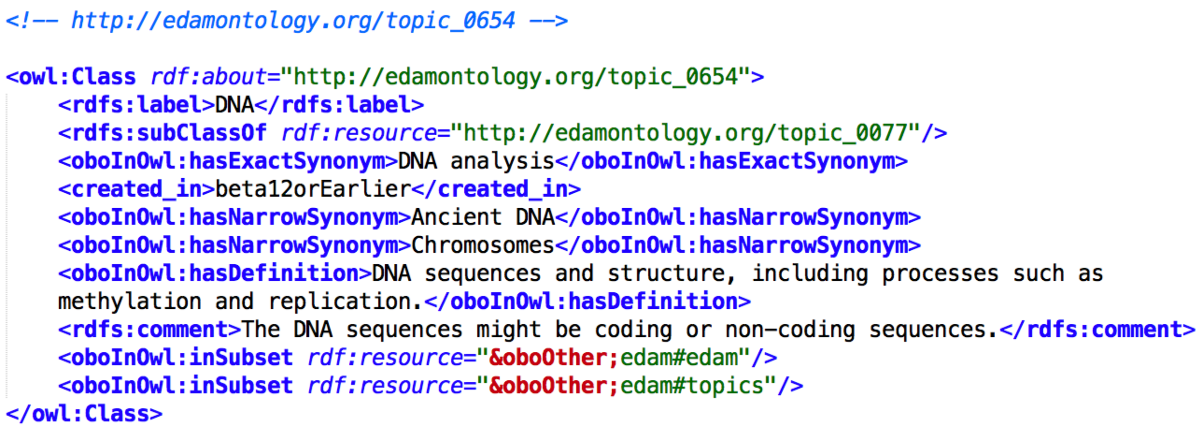
- **Superclasses:** allow representing high-level concepts [43]. Given a class C, S is a superclass of C if and only if there exist an *"is-a"* relationship from C to S – e.g., Nucleic acids (*http://edamontology.org/topic_0077*) is a superclass of DNA (*http://edamontology.org/topic_0654*) in the EDAM ontology (see Figure 2.1);

- **Subclasses:** allow refining concepts [43]. Given a class C, K is a subclass of C if and only if there exist a relationship *"is-a"* from K to C – e.g., DNA mutation (*http://edamontology.org/topic_2533*) is a subclass of DNA (*http://edamontology.org/topic_06* in the EDAM ontology (see Figure 2.1);

Ontology classes have properties to model their characteristics. **Properties** can be attributes or relationships among different classes [10]. Listing 2.1 shows the ontology class DNA *http://edamontology.org/topic_0654* in OWL format, which has properties that represent attributes (e.g., *rdfs:label*, *oboInOwl:hasExactSynonym*, *created_in*, or *rdfs:comment*). The class DNA also has properties that represent relationships (*rdfs:subClassOf* and *oboInOwl:inSubset*).

[2]http://www.ebi.ac.uk/ols/ontologies/edam/terms?iri=http://edamontology.org/topic_0654

Listing 2.1: Ontology class DNA[3]in OWL format.



```
<!-- http://edamontology.org/topic_0654 -->

<owl:Class rdf:about="http://edamontology.org/topic_0654">
    <rdfs:label>DNA</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://edamontology.org/topic_0077"/>
    <oboInOwl:hasExactSynonym>DNA analysis</oboInOwl:hasExactSynonym>
    <created_in>beta12orEarlier</created_in>
    <oboInOwl:hasNarrowSynonym>Ancient DNA</oboInOwl:hasNarrowSynonym>
    <oboInOwl:hasNarrowSynonym>Chromosomes</oboInOwl:hasNarrowSynonym>
    <oboInOwl:hasDefinition>DNA sequences and structure, including processes such as
    methylation and replication.</oboInOwl:hasDefinition>
    <rdfs:comment>The DNA sequences might be coding or non-coding sequences.</rdfs:comment>
    <oboInOwl:inSubset rdf:resource="&oboOther;edam#edam"/>
    <oboInOwl:inSubset rdf:resource="&oboOther;edam#topics"/>
</owl:Class>
```

Classes may have properties that represent terms semantically related to the class such as *synonyms*, also *homonyms*, *hypernyms*, *hyponyms*, *meronyms*, and *holonyms*. Two terms are synonyms if they are different and denote the same resource. Two terms are homonyms if they are the same and denote two different resources. Consider two different terms x and y, then according to Mangold [32]:

- x is a hypernym of y if x represents a resource that is a more general than the resource represented by y. If x is a hypernym of y, then y is a hyponym of x.
- x is a meronym of y if x represents a resource that is part of the resource represented by y. If x is a meronym of y, then y is a holonym of x.

EDAM [4] provides three properties to represent synonyms (*oboInOwl:hasExactSynonym*), hypernyms (*oboInOwl:hasBroadSynonym*), and hyponyms (*oboInOwl:hasNarrowSynonym*). Listing 2.1 shows that according to EDAM, *DNA Analysis* is a synonym of *DNA*; moreover, hyponyms of *DNA* are *Ancient DNA* and *Chromosomes*.

An **ontology term** is the value of an attribute that represents a label, synonym, hypernym, hyponym, meronym, or holonym of an ontology class [26].

2.2 Semantic Annotations

For Macario et al. [31], semantic annotations are annotations that use ontologies to eliminate ambiguities and promote a unified understanding of concepts, where an annotation is a particular kind of data that describe other data.

Oren et al. [36] give a more formal definition: a semantic annotation is a tuple $< s, p, o, c >$, where s is the subject of annotation, o is the object of annotation, p is the predicate (the relationship between s and o), and c is the context (provenance information) [36]. Consider the annotation of a workflow from myExperiment [17] – one of the largest scientific workflow repositories at the moment – using EDAM. Then,

[3]`http://edamontology.org/topic_0654`
[4]`https://github.com/edamontology/edammap/wiki`

<*http://edamontology.org/topic_0654, occurs-in, http://www.myexperiment.org/workflows/3905,* $\{date : 2017 - 07 - 16\}$> is an example of a semantic annotation, where s is a class of EDAM, p is the relationship $occurs-in$, o is a workflow, and c contains the date in which the annotation was collected.

There are research efforts to improve the annotation process using semantics – e.g., the Ontology Biomedical Annotator [26]. This is a semantic annotator service available on the web, which extracts ontology classes from free text, for biomedicine.

While many efforts try to design generic solutions for annotations, there are specific efforts to improve the annotation process of workflows. Garcia-Jimenez and Wilkinson [14] proposed a methodology to semantically annotate workflows with biomedical ontologies. This methodology was validated using data from myExperiment [17]. Such methodology comprises four steps:

1. **Filtering relevant workflows:** They used EDAM to create a dictionary of relevant terms in the domain of bioinformatics. They performed some modification to delete general terms and added some other relevant terms;

2. **Cleaning irrelevant services:** In the context of bioinformatics, they considered a service to be irrelevant when a workflow element that performs data transformations not associated with biological by meaningful analyses – e.g., merging, formating, or parsing;

3. **Retrieving service descriptions:** They collected the description of each workflow task, when available. However, just a low percentage of workflow tasks had a description;

4. **Entity extraction from descriptions to create semantic annotations:** They used the Open Biomedical Annotator [26] and thirteen biomedical ontologies to extract semantic annotations from workflow descriptions.

2.3 Workflows

A workflow is a step-by-step description of a process for achieving an objective, normally expressed in terms of inputs, outputs, and tasks [30]. Workflow tasks mainly refer to computational simulations and data analyses [2]. Although workflows are widely used in science and business [41], our work just considers scientific workflows.

Workflows are often represented as Directed Acyclic Graphs (DAG). Figure 2.3 presents a DAG representation of a workflow with title *"Get similar phenotypes for a disease and a gene"*, which was extracted from myExperiment repository. Workflow inputs are at the top of the figure, surrounded by a dashed box. Workflow outputs appear at the bottom of the figure, again surrounded by a dashed box. Other boxes represent workflow tasks (also known as activities). Finally, arrows represent the data flow.

Usually, scientific workflows are executed in Scientific Workflow Management Systems (SWMS) via a user-friendly interface. Most of the SWMS such as Kepler [29], Taverna

[5] `https://www.myexperiment.org/workflows/4796.html`

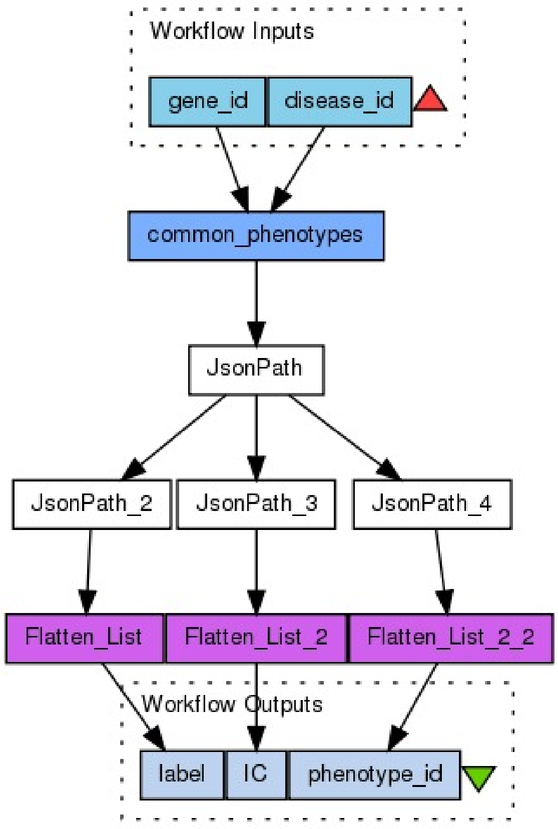


Figure 2.3: Workflow structure and components[5].

[42], VisTrails [5], and Wings [16] have functionalities as scheduling, logging, process control, provenance management, recovery mechanisms and parallelization of workflow execution [7].

An important issue is the capture of provenance. **Provenance** is a kind of metadata that explains how a data product was generated – e.g., origin, context, derivation, ownership, or history of some artifact. Provenance information allows transparency and helps to generate reproducible experiments.

2.4 Workflow Repositories

Workflow repositories emerged as a part of the solution for the reproducibility problem of experiments since they allow scientists to publish and share their workflows. Other scientists can download such workflows to reuse, re-purpose, or reproduce an experiment.

Usually, workflow authors upload their creations to repositories through a process that includes filling a form with some metadata that describe the workflow. Such metadata is often used for search and discovery of relevant workflows.

Assumption 2.4.1. For the purpose of our work, we assume that each metadata field is a tuple of type *(label, value)*, where the label is the name of the metadata field and value can be text, number, a list of text or a list of numbers.

Listing 2.2 shows the metadata in JSON format of the workflow presented in Figure 2.3. Fields *id*, *version*, *views*, and *downloads* have a number as value. Fields *repository*, *url*, *author*, *title*, *description*, *image*, and *license* have a text as value. Fields *tags*, *inputs*, *tasks*, and *outputs* have a list of texts as value. Finally, there are no fields with a list of number as value.

Listing 2.2: Workflow metadata for the workflow on Figure 2.3[6].

```
{
  "id": 4796,
  "repository": "myExperiment",
  "url": "https://www.myexperiment.org/workflows/4796.html",
  "author": "Kristina Hettne",
  "title": "Get similar phenotypes for a disease and gene",
  "description": "This workflow retrieves the similar phenotypes between a disease and
a gene based on the Monarch services at http: //monarchinitiative.org/page/services.
Phenotype similarity is calculated based on OwlSim, see http: //owlsim.org.",
  "tags": [
    "common",
    "disease",
    "gene",
    "monarch",
    "phenotype"
  ],
  "image": "https://www.myexperiment.org/workflows/4796/versions/1/previews/medium",
  "version": 1,
  "inputs": [
    "disease_id",
    "gene_id"
  ],
  "tasks": [
    "common_phenotypes",
    "JsonPath",
    "JsonPath_2",
    "Flatten_List",
    "JsonPath_3",
    "Flatten_List_2",
    "JsonPath_4",
    "Flatten_List_2_2"
  ],
  "outputs": [
    "label",
    "IC",
    "phenotype_id"
  ],
  "license": "Creative Commons Attribution-Share Alike 3.0 Unported License",
  "views": 178,
  "downloads": 49
}
```

There are many scientific workflow repositories referring to different scientific domains, SWMS, and retrieval methods. There follow examples of some workflow repositories.

2.4.1 MyExperiment

This is a collaborative platform to publish and share scientific workflows [11]. Most users of this platform are scientists in the life sciences; however, other communities (e.g., chemistry, social statistics, and music information retrieval) are expanding their participation

[6]`https://www.myexperiment.org/workflows/4796.html`

in myExperiment. It stores workflows, metadata, data inputs, provenance information, versions, among other assets [11]. Moreover, it supports workflows from different SWMS such as Taverna, RapidMiner, Galaxy, KNIME, Kepler, among others.

MyExperiment allows keyword-based search and filtering by some workflow metadata such as title, description, tags, user, license, etc. It also provides structure-based search for workflow discovery using workflows that share the same services as similarity measure.

The myExperiment project hosts 392 scientific groups, 10,501 registered users and more than 2,800 scientific workflows in its database (as of May, 2017). It has been used in several research projects [39, 40, 14].

2.4.2 CrowdLabs

CrowdLabs is a system that allows sharing datasets, computational pipelines (a.k.a. workflows), and provenance information of scientific experiments [34]. This system is based on social websites and it aims to offer a rich collaborative environment for scientists via a set of tools and scalable infrastructure. Although CrowdLabs can be integrated with any SWMS that runs in server mode with an open API, it works mainly with VisTrails [34]. It uses VishMashup, which is a VisTrails extension to allow user interaction with workflows using a web-based solution. Hence, this system provides workflow creation, modification, and execution without installing aditional software. The provenance information derived from workflows together with the results are easily shared on wikis and LaTeX via some plugins provided by the system.

The CrowdLabs architecture handles two types of resources: data analysis and visualization resources (workflows, visualizations, packages, and datasets) and social resources (profiles, projects, groups, and blogs). Moreover, this system provides a public RESTful HTTP API to access to such resources [34], which ease the integration of new functionalities by third-party developers.

Clowdlabs [34] by default shows all the workflows in the repository. It also provides keyword-based search and some filters to improve the precision of query results, where users can order such results by title, date, and rating.

2.4.3 Galaxy

Galaxy is a platform for performing computational analyses of genomic data via workflows, providing open, transparent, and reproducible results in genomic science [18]. Galaxy has four main components: workspace, workflows, data libraries, and user repositories. A workspace is where the user develops and deploy workflows; although Galaxy workspace is mainly a web platform, users can use the Galaxy application to set up local servers. Workflows presents the user list of workflow. Users can import theirs datasets (user repositories) or use dataset available in the platform (data libraries).

Galaxy also tracks the provenance information of each workflow and metadata collected via user annotations – e.g., title, description, and tags. Galaxy provides three ways for reproducibility and transparecy: a model for sharing items (datasets and workflows), a platform to display datasets and execute workflows, and wiki pages to communicate details about scientific experiments [18].

This platform provides keyword-based search, filtering the public repository by title, author, tag, and annotations to find items (datasets and workflows) of interest [18].

2.4.4 CloudFlows

Clowdflows is a web-based platform for creation, execution, and sharing of data mining workflows [28]. This platform provides a user interface that allows creating and editing workflows, using drag, drop, and connect operation over workflow components.

There exist a set of predefined workflow components, mostly created using Orange [7] and Weka[8]. Both are tools to interact with machine learning algorithms but Orange focus more in visualization and Weka in data mining tasks. Clowdflows users can also create custom workflow components as web services that should be stored on third-party servers [28].

Clowdflows provides remote execution and visualization, which implies that users do not require additional software to manipulate workflows [28]. Moreover, it allows sharing of workflows.

This platform does not provide mechanisms for workflow search but it allows workflow discovery by listing all the workflows in the platform for reuse, re-purpose, and experiment reproducibility [28].

2.4.5 PBase

PBase is a scientific workflow repository that also stores provenance information using ProvONE [9], which is a standard for modeling, representing, and sharing provenance information. This standard records provenance information of the workflow and its components. It is intrinsically graph oriented due to the DAG (Direct Acyclic Graph) representation of workflow components[9].

PBase uses Neo4j – a NoSQL graph database – to store ProvOne information [9]. Moreover, it uses Cypher – Neo4j's declarative graph query language – for queries. Neo4j and Cypher are a powerful combination for provenance information of workflows because it allows four types of queries [9]:

- **Lineage queries,** which deal with the derivation of data products – e.g., *"what are the datasets involved in the generation of this plot?"*;
- **Execution analysis queries** – e.g., *"find the processes in a workflow that were not completed"*;
- **Search queries** – e.g., *"find all the workflows that used a bilinear regrid module"*;
- **Statistical queries** – e.g., *"list the most used modules across all workflows"*;

However, Cypher is a complex query language for most scientists, a big barrier for its use.

[7]`https://orange.biolab.si/`
[8]`http://www.cs.waikato.ac.nz/ml/weka/`

2.4.6 OPMW Workflow Repository

The OPMW Workflow Repository [15] uses Linked Data principles [4] to enable direct access to workflows, workflow components and datasets identified with a unique URI and using RDF to represent them. Each resource is annotated with OPMW, a model to semantically annotate workflows derived from the Open Provenance Model (OPM) [35] but it includes terms for abstract workflows. Abstract workflows describe workflow components in a human readable way to make workflow more understandable; therefore, more reusable.

Workflows are accessible using semantic-based search via a SPARQL Endpoint [15]. Hence, other applications can use these workflows and link them using Linked Data. Nevertheless, this retrieval mechanism is hard to grasp for people who do not know about SPARQL and OPMW. For this reason, there is a module that allows keyword-based search on workflows, authors, and resources. The search mechanism looks for exact match, providing an auto-complete functionality.

This repository stores exclusively workflows implemented in Wings, a scientific workflow management system [15].

2.5 Workflow Retrieval

The retrieval process for workflow repositories can be performed via a wide range of methods; some of these methods are keyword-based search, structure-based search (also known as topology-based search), and semantic-based search.

2.5.1 Keyword-based Search

Keyword-based retrieval systems use the terms in a free text query to match with terms in a subset of metadata for each workflow in the repository.

Keyword-based methods are widely implemented on scientific workflow repositories – e.g., myExperiment [17].

These methods have good precision, retrieving relevant workflows that match the scientist's query. The downside is that in some cases it has a low recall because some relevant results are hidden. Scientists may use different words to refer to the same concepts and annotate a workflow according to distinct criteria. Hence, in such case, these methods just return a subset of all the relevant results for the user's query. For example, in Figures 2.4 and 2.5, we can see different results when we use different queries that are semantically related.

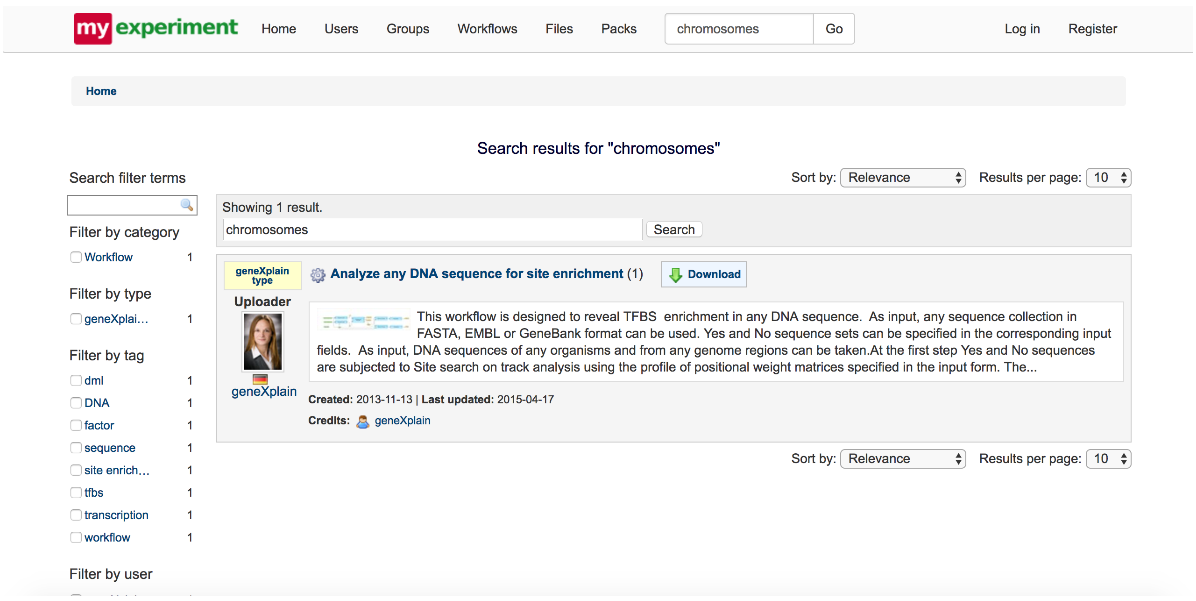
Figure 2.4 shows a search in myExperiment using the query "chromosomes"; myExperiment's retrieval system returns one result because the string "chromosomes" is contained in the value of metadata field *description* of the workflow with title *Analyze any DNA sequence for site enrichment*[9].

Figure 2.5 shows a search in myExperiment using the query "chromosome", which is semantically similar to "chromosomes". MyExperiment's retrieval system returns 84

[9]`https://www.myexperiment.org/workflows/3905.html`

results and none of them correspond to the one obtained with the query “chromosomes”.

Shao et al. present WISE [38], a workflow search engine that returns concise results on repositories of workflow hierarchies using keyword-based search. The hierarchy considers that each workflow task can be a workflow by itself recursively[38]. This search engine uses the metadata of components and subcomponents of each workflow and results must contain at least one match of each term in the query. Although it is possible to show a fine-grained graph with components and subcomponents of workflows given as results, this search engine shows the smallest graph that contains the query terms. Hence, it shows the workflow and presents a workflow component in detail (as another graph inside a box) when its subcomponents have metadata that match with terms in the query[38].


myexperiment Home Users Groups Workflows Files Packs chromosomes Go Log in Register

Home

Search results for "chromosomes"

Search filter terms

Sort by: Relevance Results per page: 10

Showing 1 result.

chromosomes Search

Filter by category
Workflow 1

Filter by type
geneXplai... 1

Filter by tag
dml 1
DNA 1
factor 1
sequence 1
site enrich... 1
tfbs 1
transcription 1
workflow 1

Filter by user

geneXplain type

Analyze any DNA sequence for site enrichment (1) Download

Uploader

geneXplain

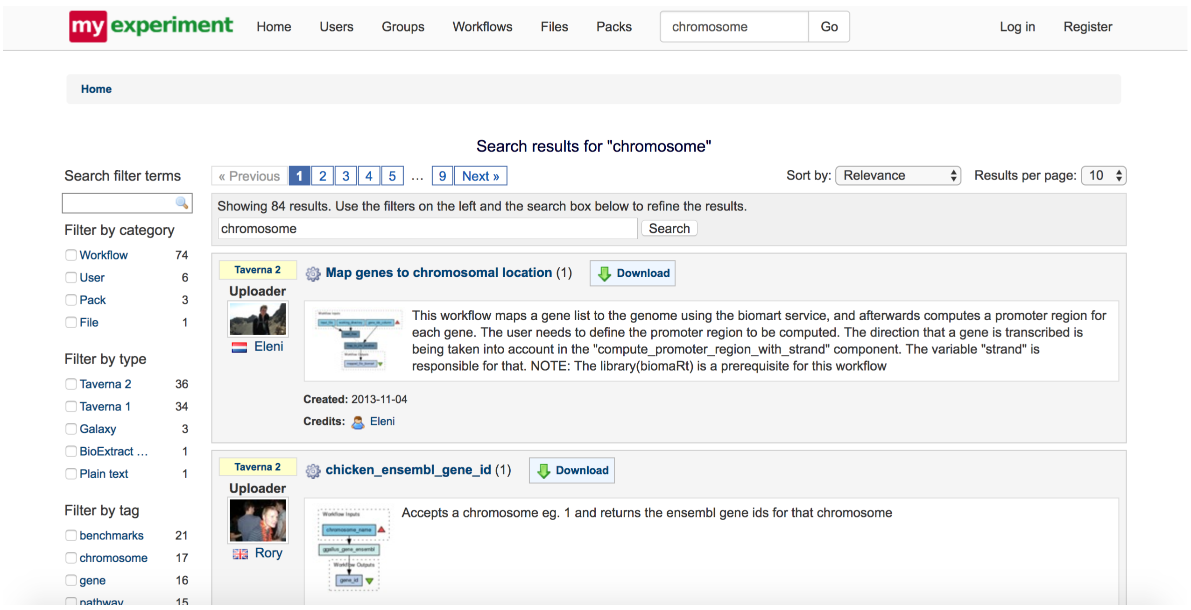
This workflow is designed to reveal TFBS enrichment in any DNA sequence. As input, any sequence collection in FASTA, EMBL or GeneBank format can be used. Yes and No sequence sets can be specified in the corresponding input fields. As input, DNA sequences of any organisms and from any genome regions can be taken.At the first step Yes and No sequences are subjected to Site search on track analysis using the profile of positional weight matrices specified in the input form. The...

Created: 2013-11-13 | **Last updated:** 2015-04-17

Credits: geneXplain

Sort by: Relevance Results per page: 10

Figure 2.4: Keyword-based search on myExperiment using the query “chromosomes”.


myexperiment Home Users Groups Workflows Files Packs chromosome Go Log in Register

Home

Search results for "chromosome"

Search filter terms

« Previous 1 2 3 4 5 … 9 Next »

Sort by: Relevance Results per page: 10

Showing 84 results. Use the filters on the left and the search box below to refine the results.

chromosome Search

Filter by category
Workflow 74
User 6
Pack 3
File 1

Filter by type
Taverna 2 36
Taverna 1 34
Galaxy 3
BioExtract ... 1
Plain text 1

Filter by tag
benchmarks 21
chromosome 17
gene 16
pathway 15

Taverna 2

Map genes to chromosomal location (1) Download

Uploader

Eleni

This workflow maps a gene list to the genome using the biomart service, and afterwards computes a promoter region for each gene. The user needs to define the promoter region to be computed. The direction that a gene is transcribed is being taken into account in the "compute_promoter_region_with_strand" component. The variable "strand" is responsible for that. NOTE: The library(biomaRt) is a prerequisite for this workflow

Created: 2013-11-04

Credits: Eleni

Taverna 2

chicken_ensembl_gene_id (1) Download

Uploader

Rory

Accepts a chromosome eg. 1 and returns the ensembl gene ids for that chromosome

Figure 2.5: Keyword-based search on myExperiment using the query “chromosome”.

2.5.2 Structure-based Search

Structure-based retrieval systems use the workflow structure to look for workflows with topology similar to an input workflow. There are three components in the DAG representation: the inputs, the outputs, and the processes. Data links represent the flow of data from one component to the next [40]. The elements are used individually or collectively to perform the search.

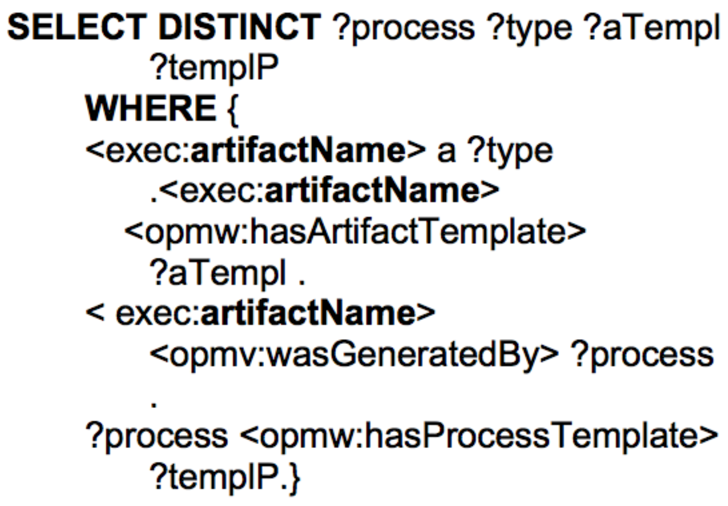
The work in [40] presents a hybrid architecture for a retrieval system using keyword and structure search. Most of the work was focused on the structure-based search. This work uses a topological sort of the DAG representation of workflows. Thus, the order of execution of components is important when a workflow is compared with others. A normalization of the similarity scores is performed to take into account the size of each workflow. Finally, workflows are ranked by the normalized scores. This work uses the metadata (e.g., label, URL, and type) of each component to calculate similarity scores of workflow components.

This search method is a good choice for workflow discovery. Nevertheless, it needs a workflow or a subset of the workflow components as input to return relevant results. Usually, it requires expensive preprocessing of workflows in a repository. Finally, topology alone is not a good choice in searching and has to be accompanied by, e.g., keyword-based search.

2.5.3 Semantic-based Search

Semantic-based retrieval systems use ontologies to match queries against the semantic annotations in the workflow repositories. This method provides the higher recall and precision among the search methods [27]. Nevertheless, it often needs languages with complex syntax (like SPARQL), which represents a barrier to the user [27].

Listing 2.3: Example of query in SPARQL [15].



```
SELECT DISTINCT ?process ?type ?aTempl
        ?templP
    WHERE {
    <exec:artifactName> a ?type
        .<exec:artifactName>
      <opmw:hasArtifactTemplate>
        ?aTempl .
    < exec:artifactName>
        <opmv:wasGeneratedBy> ?process
        .
    ?process <opmw:hasProcessTemplate>
        ?templP.}
```

Usually, repositories that support semantic-based search using SPARQL offer an alternative retrieval method to ease the search process for users. For example, OPMW

Table 2.1: Related work comparison.

Solution	Retrieval Approach			Multiple Repositories	SWMS
	Keyword	Semantics	Structure		
myExperiment [11]	YES	NO	YES	NO	ALL
CrowdLabs [34]	YES	NO	NO	NO	Vistrails
Galaxy [18]	YES	NO	NO	NO	Galaxy
ClowdFlows [28]	NO	NO	NO	NO	ClowdFlows
PBase [9]	YES	NO	YES	NO	ALL
OPMW [15]	YES	YES	NO	NO	Wings
Wise [38]	YES	NO	NO	YES	ALL
Starlinger, 2016 [40]	YES	NO	YES	YES	ALL
Bergmann, 2014 [2]	NO	YES	YES	YES	Wings
WorkflowHunt [12]	YES	YES	NO	YES	ALL

Workflow Repository [15] offers keyword-based and semantic-based search. Listing 2.3 shows an example of a query using SPARQL in this repository.

The work in [2] uses a hybrid approach between structure-based search and semantic-based search applied to workflows in the field of case-based reasoning (CBR). This is a method to solve problems using past experiences with similar problems. In this work, workflows are semantically annotated using ontologies. Such annotations are used to extract the semantics of each workflow component and this information is used to compute semantic similarity measures. This work uses the distance between ontology classes in the hierarchical representation of an ontology to extract semantic similarity measures from semantic annotations of workflows. Local similarity measures in each workflow element are aggregated to create a global similarity measure. Finally, this work was validated on a workflow repository using Wings (a workflow management system) and CAKE (a process-oriented system for CBR).

2.6 Final Remarks

This chapter presented general definitions of ontologies, workflows, and semantic annotations. Furthermore, this chapter described related work about retrieval mechanisms in scientific workflow repositories.

Table 2.1 presents fundamental characteristics of the reviewed research on retrieval mechanisms for scientific workflow repositories. We summarize architectures and papers in this chapter, considering the following criteria:

- Retrieval approach, which can be keyword, semantics, or structure;

- Multiple Repositories, which explains if the retrieval system looks for workflows in multiple repositories;
- SWMS (Scientific Workflow Management System), indicating the system(s) supported by the retrieval system.

In the next chapter, we will present the architecture of our retrieval system, WorkflowHunt.

Chapter 3

Architecture of WorkflowHunt

This chapter describes the architecture of WorkflowHunt, a hybrid retrieval system that combines keyword and semantic search for scientific workflow repositories. Section 3.1 presents a high-level explanation about the proposed architecture. Sections 3.3, 3.2, and 3.4 detail the main layers and algorithms that make up the architecture.

3.1 Architecture

WorkflowHunt aims to retrieve from arbitrary workflow repositories, the workflows that are relevant for users as long as the repositories provide the appropriate workflow metadata and all are available on the web (e.g., via URL). This helps to promote workflow reuse and re-purpose across different scientific fields. Our architecture focuses on improving the workflow interoperability among different scientific domains by offering semantic and keyword search for scientific workflow repositories.

Figure 3.1 shows the WorkflowHunt architecture, which comprises three main layers: *Persistence*, *Pre-processing*, and *Search*. The *Persistence Layer* is composed by four repositories: Local Workflow Repository, Metadata Repository, Ontology Repository, and Repository of Semantic Annotations. This layer also includes the keyword index and hybrid index (a combination among semantic and keyword indices). The *Pre-processing Layer* prepares the data for search, which is performed by the *Search Layer*. The latter provides all the services needed to find workflows using keyword and semantic search. These services can be accessed through the Web Interface by users (scientists who are looking for workflows of interest). Finally, solid arrows represent the data flow in the system and dashed arrows represent data connections among elements.

WorkflowHunt prepares the data offline using the Pre-processing Layer for subsequent processing. The preparation process starts when the Metadata Collector extracts metadata from external (1a) and local (2a) workflow repositories. External workflow data requires a Web Crawler; scientists can also take advantage of workflows created by their own research team, stored in the Local Workflow Repository. Extracted metadata is stored in the Metadata Repository (3a). The Semantic Annotator semantically annotates the metadata using the Dictionary of Ontology Terms, which maps ontology terms to ontology classes in the Ontology Repository (4a). These semantic annotations are stored in the Repository of Semantic Annotations (5a). The Index Generator then creates indices for metadata (keyword index) and a combination of keywords and semantic annotations (hybrid index) (6a-7a). We use an inverted index for keyword-based search and a modified version of the same index data structure for semantic search (see Subsection 3.3.2). Both indices refer to the data in Metadata Repository. Once repositories are created and indices are constructed, the system is ready to handle queries using keyword and semantic search. Each new workflow processed needs to go through this process. If new ontologies are added, the Dictionary of Ontology Terms is updated and steps (4a) onwards are executed.

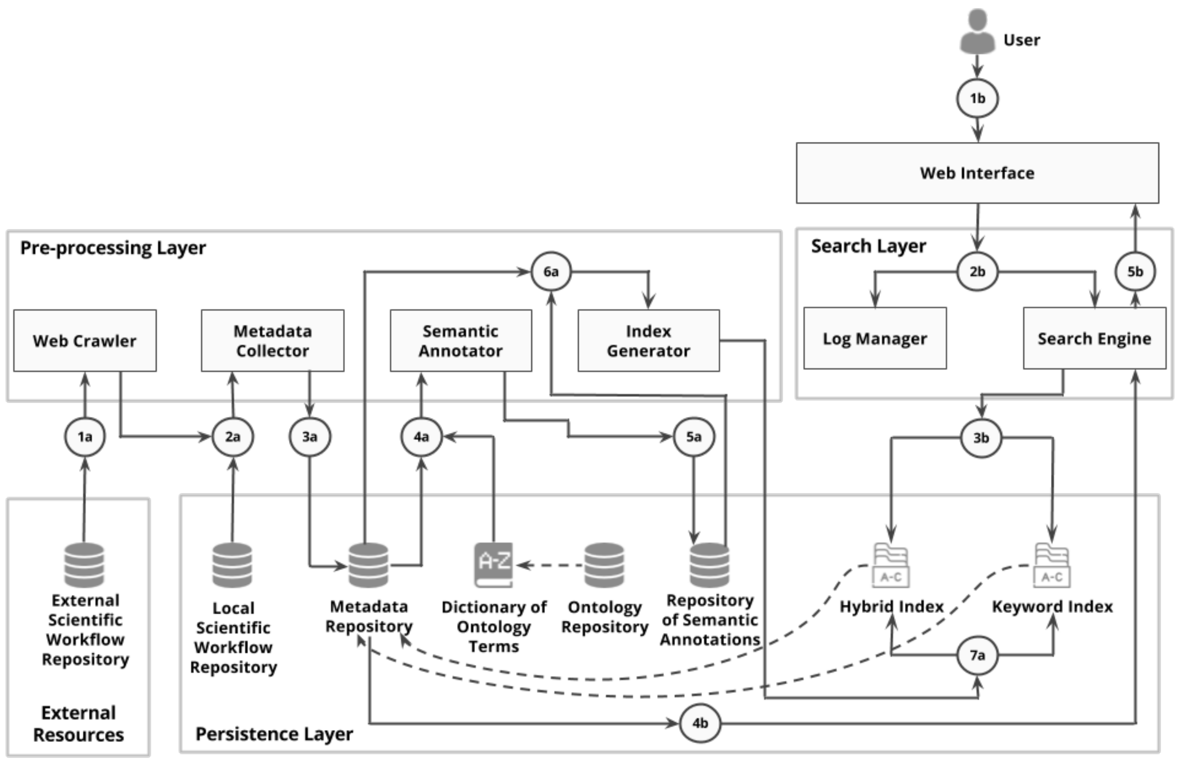


Figure 3.1: WorkflowHunt Architecture.

A usual retrieval scenario starts when a user poses a query through the Web Interface (1b). This interface redirects the query to the Search Engine and the Log Manager (2b). The Log Manager stores the query for performance management. The Search Engine executes the query using keyword or semantic indices (3b), depending on the user preferences. These indices are linked to the workflow metadata in the Metadata Repository. The result is a list of metadata with links pointing at relevant workflows, presented to the user at the interface (4b-5b).

3.2 Persistence Layer

The Persistence Layer comprises four repositories:

Local Workflow Repository: It is a private repository of the scientist's team, which stores abstract and executable workflows. Such workflows can be created using different Scientific Workflow Management Systems (SMWS).

Metadata Repository: It contains the metadata collected from Local and External workflow repositories. Such metadata includes a link to the actual workflows. Local repositories store workflow metadata directly in this repository. Metadata from external repositories are collected using a Web Crawler.

Ontology Repository: It contains all the ontology classes, ontology terms, and their relationships for different ontologies of different domains.

Repository of Semantic Annotations: It stores semantic annotations extracted from the data in the Metadata Repository, using the Dictionary of ontology Terms to link with ontology classes in the Ontology Repository. These annotations are important to create the hybrid index (see 3.3.2).

3.3 Pre-processing Layer

The Pre-processing Layer prepares data to be used by the Search Layer (Keyword and Hybrid indices, and Metadata Repository). The *Web Crawler* collects workflow metadata from external repositories. Usually, external workflow repositories provide the metadata of their workflows via various formats — e.g., HTML, RDF, JSON, XML, etc. Thus, this module may require customizations if a new repository is integrated with WorkflowHunt because it transforms these formats to a normalized format (see Assumption 2.4.1). The *Metadata Collector* receives metadata from the Local Scientific Workflow Repository or the Web Crawler and stores it in the Metadata Repository. Subsections 3.3.1 and 3.3.2 detail functionalities of *Semantic Annotator* and *Index Generator* respectively.

3.3.1 Semantic Annotations

The Dictionary of Ontology Terms associates ontology terms with ontology classes in the Ontology Repository. Algorithm 1 presents the BUILD_DICTIONARY algorithm to create this dictionary. The algorithm takes as input a list of ontologies (including ontology terms, ontology classes, and relationships) and the dictionary is created as result. It iterates over the ontology terms of each ontology class in the repository to associate them with the classes (lines 3 - 9). Ontology terms are the keys and ontology classes are the values. Some ontology terms can be derived from external sources (not included in the ontologies) – e.g., synonyms in WordNet[13]. Finally, the result is stored in the Dictionary of Ontology Terms (line 10).

The Semantic Annotator creates semantic annotations by linking the workflow metadata in the Metadata Repository with the ontology classes in the Dictionary of Ontology Terms. The Semantic annotator uses the SEMANTIC_ANNOTATION algorithm (see Algorithm 2), which is inspired by the Open Biomedical Annotator [26].

Our algorithm receives as input the metadata of workflows to be annotated, a dictionary of ontology terms, and a list of options for semantic expansion. It stores the resulting

Algorithm 1 Algorithm to build Dictionary of Ontology Terms

Require: O is a list of Ontologies

```
1: function BUILD_DICTIONARY($O$)
2:     $D \leftarrow \emptyset$
3:     for each $ontology \in O$ do
4:         for each $class \in ontology$ do
5:             for each $term \in class$ do
6:                 $D[term] \leftarrow class$
7:             end for
8:         end for
9:     end for
10:    $store(D)$
11: end function
```

semantic annotations in the Repository of Semantic Annotations. We create semantic annotations associating many ontology classes with a workflow through the $occurs - in$ relationship. C represents the context or provenance information such as creation date of the annotation. Recall that a semantic annotation, here, is the tuple $< s, p, o, c >$, where s is the subject of annotation, o is the object of annotation, p is the predicate (the relationship between s and o), and c is the context (provenance information) [36] (see line 7 of Algorithm 3).

The algorithm iterates over the list of workflow metadata and extracts the text that belongs to each metadata field (line 5). This text and the workflow URI (line 6) are used as input for the TEXT_ANNOTATION algorithm (see Algorithm 3), which identifies the ontology classes that are in the text and creates semantic annotations (lines 7 - 8). Finally, the semantic annotations are stored in the Repository of Semantic Annotations (line 11).

Algorithm 2 Semantic Annotation Algorithm

Require: W is the metadata of workflows to be annotated, D is a dictionary of ontology terms, and Opt is a list of options for semantic expansion

```
1: function SEMANTIC_ANNOTATION($W, D, Opt$)
2:     $SA \leftarrow \emptyset$
3:     for each $metadata \in W$ do
4:         for each $field \in getFields(metadata)$ do
5:             $text \leftarrow getValue(field)$
6:             $workflowURI \leftarrow getWorkflowURI(metadata)$
7:             $semanticAnnotations \leftarrow TEXT\_ANNOTATION(text, workflowURI, D, Opt)$
8:             $SA.add(semanticAnnotations)$
9:         end for
10:    end for
11:    $store(SA)$
12: end function
```

In more detail, the TEXT_ANNOTATION algorithm (see Algorithm 3) receives as

Algorithm 3 Semantic Annotations from Free Text Algorithm

Require: T is free text, which is the value of a metadata field of a workflow, $WorkflowURI$ is the URI of such workflow, D is a dictionary of ontology terms, and Opt is a list of options for semantic expansion

```
1: function TEXT_ANNOTATION($T, WorkflowURI, D, Opt$)
2:     $SA \leftarrow \emptyset$
3:     $D \leftarrow sortByLength(D)$
4:     for each $pair(term, class) \in D$ do
5:         if $term \subseteq T$ then
6:             $classURI \leftarrow getURI(class)$
7:             $semanticAnnotation \leftarrow (classURI, \text{“}occurs - in\text{”}, WorkflowURI, C)$
8:             $SA.add(semanticAnnotation)$
9:             $expandedAnnotations \leftarrow SEM\_EXPANSION(semanticAnnotation, Opt)$
10:            $SA.add(expandedAnnotations)$
11:            $termLength \leftarrow getLength(term)$
12:            $w \leftarrow createWildcard(WHITE\_SPACE, termLength)$
13:            $T \leftarrow replace(T, term, w)$
14:        end if
15:    end for
16:    return $SA$
17: end function
```

input a text, a workflow URI, a dictionary of ontology terms, and a list of options for semantic expansion. Its output is a list of semantic annotations with the ontology classes detected in the input text and their corresponding semantically expanded annotations according to the input options. The algorithm starts by sorting the dictionary of ontology terms by the length of the ontology terms in descending order (line 3). We give priority to larger strings because they probably have more semantic content than shorter strings. Moreover, we should avoid overlapping semantic annotations in the text as suggested in [1]. For example, consider the string “a nucleic acid sequence” that belongs to some metadata value of a workflow. If our system annotates that string with the ontology term “nucleic acid sequence”[1], then it should not create an annotation with the ontology term “sequence”[2] on the same string because this causes an overlap. Overlap can be avoided by replacing reoccurring terms by, e.g., blanks. A given substring cannot be annotated by more than one ontology term, thus avoiding overlapping (lines 4 - 15). Finally, the algorithm returns semantic annotations with the corresponding semantic expansion (line 16).

The SEM_EXPANSION algorithm (see Algorithm 4) expands an initial semantic annotation using the super/sub class structure of the ontologies. It receives as input a semantic annotation and a list of options for semantic expansion and returns a list of semantic annotations expanded from the original one. Options are generalization (lines 3-6), specialization (lines 7-10), and semantic distance expansion (lines 11-14). General-

[1]`http://edamontology.org/data_2977`

[2]`http://edamontology.org/data_2044`

Algorithm 4 Semantic Expansion Algorithm

Require: *sa* is a semantic annotation and *Opt* is a list of options for semantic expansion

```
1: function SEM_EXPANSION($sa, Opt$)
2:     $SE \leftarrow \emptyset$
3:     if $Opt.hasSemanticGeneralization()$ then
4:         $generalizedAnnotations \leftarrow generalization(sa, Opt.gSize, C)$
5:         $SE.add(generalizedAnnotations)$
6:     end if
7:     if $Opt.hasSemanticSpecialization$ then
8:         $specializedAnnotations \leftarrow specialization(sa, Opt.sSize, C)$
9:         $SE.add(specializedAnnotations)$
10:    end if
11:    if $Opt.hasSemanticDistanceExpansion$ then
12:        $semDistanceAnnotations \leftarrow semanticDistanceExpansion(sa, Opt.sdSize, C)$
13:        $SE.add(semDistanceAnnotations)$
14:    end if
15:    return $SE$
16: end function
```

ization (resp. specialization) creates new semantic annotations with the same structure of the initial semantic annotation but replacing the ontology class associated with its superclasses (resp. subclasses) in the ontology. The parameter *gSize* represents the number of superclasses that will be included in the semantic expansion (resp. *sSize* for number of subclasses). Semantic distance expansion creates new semantic annotations with the same structure of the initial semantic annotation but replacing the ontology class associated with its neighbors in the ontology. The parameter *sdSize* represents the number of edges that separate the original ontology class from the ontology classes that will be included in the expansion.

For example, in the case of generalization, consider the string "...gene ids for that chromosome" that belongs to the description metadata of a workflow with title "chicken_ensembl_gene_id"[3]. Our system detects the ontology term "chromosome" that belongs to the ontology class *DNA*[4] and creates a semantic annotation
$(http : //edamontology.org/topic\_0654$, "$occurs - in$",
$http : //www.myexperiment.org/workflows/902, \{date : \text{"2017} - 05 - 04\text{"}\})$. Next, the system finds that *Nucleic acids*[5] is the superclass of *DNA*[6] and creates another semantic annotation $(http : //edamontology.org/topic\_0077$, "$occurs - in$",
$http : //www.myexperiment.org/workflows/902, \{date : \text{"2017} - 05 - 04\text{"}\})$, which is result of replacing the ontology class in the initial semantic annotation by one of its superclasses in the ontology (in this case, the immediate superclass).

[3]http://www.myexperiment.org/workflows/902
[4]http://edamontology.org/topic_0654
[5]http://edamontology.org/topic_0077
[6]http://edamontology.org/topic_0654

3.3.2 Indices

Indices are used as data structures to provide distinct kinds of access to workflows. We provide two indices – keyword and hybrid.

The general algorithm is the following:

1. Traverse metadata text
2. Extract tokens from text
   (a) Extract keywords to build keyword index (normalized K-token)
   (b) Extract hybrid expressions to build hybrid index (normalized H-token)
3. Create and store indices, eliminating duplicate tokens
   (a) Construct K-index structure
   (b) Construct H-index structure

As will be seen, the keyword index contains keywords found in textual metadata. A K-index entry is expressed as <**normalized K-token, list of workflow ids**>, where the list indicates the ids of workflows associated with that keyword, which has been normalized. Normalization is the process of standardizing tokens so that tokens with superficial differences in the character sequences are grouped together to represent the same token [33]. In this project, normalization is done by the Index Generator, which converts the workflow metadata to lowercase, removes the stop words (common words that have little value to match relevant results), and applies a stemming algorithm on the metadata. Stemming is the process of reducing the many (inflectional and derivationally related) forms of a word to a common base form by chopping off the ends of words [33].

A H-index entry can be of two types:

1. <**normalized K-token, list of workflow ids**> or
2. <**normalized H-token, list of workflow ids**>

The normalized H-token contains a text representation of an ontology class using with the format <**ontology**>**:**<**class identifier**>, which is obtained using ontology terms (i.e., a semantic reference). When no such semantic reference exists, the hybrid index will contain a normalized K-token. This is why this is called a hybrid index.

Algorithm 5 shows the algorithm used for indexing (BUILD_INDICES Algorithm). Lines 4 through 16 extract tokens (K-tokens and H-tokens) from metadata fields, normalize them into lists, which are used to build or update a keyword index (line 18) and a hybrid index (line 20). Moreover, tokenization and normalization differ for keyword and hybrid indices.

In more detail, lines 8 and 9 extracts atomic keywords (K-tokens) to build the keyword index. Figure 3.2 shows how the keyword index is created. Two workflows are used in the figure: *Transcribe a DNA sequence into an RNA sequence*[7] and *Add "chr" to*

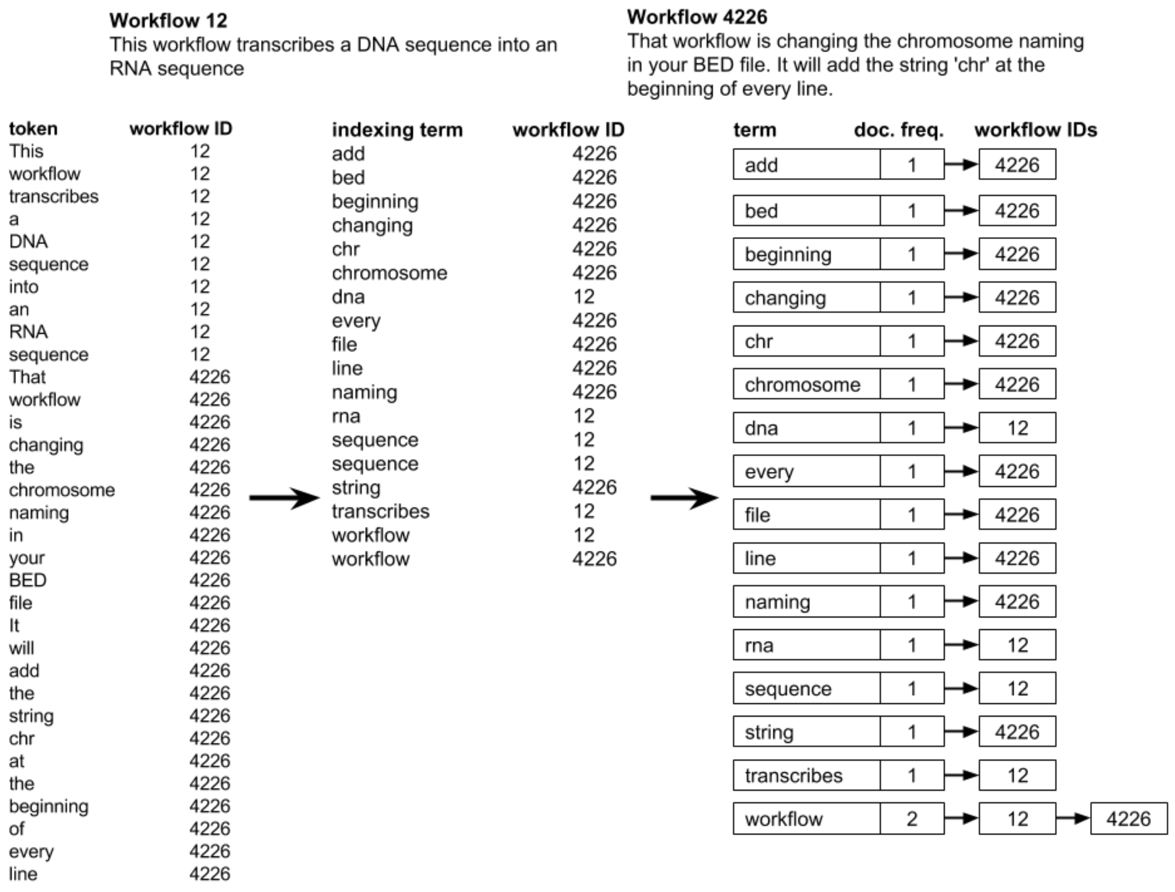
[7]`http://www.myexperiment.org/workflows/12.html`

Algorithm 5 Indexing Algorithm

Require: W is a list of metadata grouped by workflow in the Metadata Repository

```
1: function BUILD_INDICES($W$)
2:     $normalizedKtokens \leftarrow \emptyset$
3:     $normalizedHtokens \leftarrow \emptyset$
4:     for each $workflow \in W$ do
5:         for each $metadata\_field \in workflow$ do
6:             $text \leftarrow getValue(metadata\_field)$
7:                 ▷ get a set of keywords (K-tokens) from text and normalize the set
8:             $Ktokens \leftarrow tokenize(text)$
9:             $normalizedKtokens \leftarrow normalizedKtokens \cup normalize(Ktokens)$
10:                ▷ get a set of hybrid expressions (H-tokens) from text and normalize the set
11:            $sem\_annotations \leftarrow getSemanticAnnotations(workflow)$
12:            $hybridText \leftarrow replaceOntologyTermsByOntologyClasses(text, sem\_annotations)$
13:            $Htoken \leftarrow tokenize(hybridText)$
14:            $normalizedHtokens \leftarrow normalizedHtokens \cup normalize(Htokens)$
15:        end for
16:    end for
17:                ▷ Create or update K-Index using a list of normalized K-tokens
18:    $SAVE\_K\_INDEX(normalizedKtokens)$
19:                ▷ Create or update H-Index using a list of normalized H-tokens
20:    $SAVE\_H\_INDEX(normalizedHtokens)$
21: end function
```

the first column of a 6 column BED file[8]. This example takes into account just one metadata field (description). The tokens in each document are tagged by their workflow identifier (Figure 3.2 - left). Next, the tokens are normalized and converted to indexing terms (Figure 3.2 - middle). In this case, the stop words *a*, *an*, *at*, *in*, *into*, *is*, *it*, *of*, *that*, *the*, *this*, *will*, and *your* were removed. Tokens were converted to lowercase and sorted alphabetically. Instances of the same indexing term are grouped by term and then by workflow identifier. Consequently, each indexing term has a list of workflow identifiers (Figure 3.2 - right). Usually, inverted indexes store summary information like the document frequency of each indexing term [33]. This information is used to rank results and improve query time efficiency. Each list of workflow identifiers can store other information like the term frequency – the frequency of each term in each workflow metadata value – or the position(s) of the term in each workflow metadata value.



Workflow 12
This workflow transcribes a DNA sequence into an RNA sequence

Workflow 4226
That workflow is changing the chromosome naming in your BED file. It will add the string 'chr' at the beginning of every line.

token	workflow ID
This	12
workflow	12
transcribes	12
a	12
DNA	12
sequence	12
into	12
an	12
RNA	12
sequence	12
That	4226
workflow	4226
is	4226
changing	4226
the	4226
chromosome	4226
naming	4226
in	4226
your	4226
BED	4226
file	4226
It	4226
will	4226
add	4226
the	4226
string	4226
chr	4226
at	4226
the	4226
beginning	4226
of	4226
every	4226
line	4226

indexing term	workflow ID
add	4226
bed	4226
beginning	4226
changing	4226
chr	4226
chromosome	4226
dna	12
every	4226
file	4226
line	4226
naming	4226
rna	12
sequence	12
sequence	12
string	4226
transcribes	12
workflow	12
workflow	4226

term	doc. freq.	workflow IDs	
add	1	4226	
bed	1	4226	
beginning	1	4226	
changing	1	4226	
chr	1	4226	
chromosome	1	4226	
dna	1	12	
every	1	4226	
file	1	4226	
line	1	4226	
naming	1	4226	
rna	1	12	
sequence	1	12	
string	1	4226	
transcribes	1	12	
workflow	2	12	4226

Figure 3.2: Example of Keyword Index.

Similarly, lines 11-14 extract hybrid expressions (H-tokens) from hybrid text to build the hybrid index. This index merits a more detailed explanation. A hybrid expression can be an atomic keyword or a reference to an ontology class. Ontology classes are identified using ontology terms associated with them, and ontology terms can contain more than one keyword. Hence, to avoid mistakes in the tokenization they need to be mapped to a format with an atomic form to be considered as tokens.

[8]`http://www.myexperiment.org/workflows/4226.html`

Figure 3.3 shows how the text is converted to hybrid text. A hybrid text is a text that combines atomic keywords with ontology classes (in atomic form) identified in the text. Two workflows are used in the figure: *Transciption (DNA into RNA)*[9] and *Add "chr" to the first column of a 6 column BED file*[10]. This example considers just one metadata field: description (Figure 3.3 - top). Ontology terms, in green, are detected using the semantic annotations (Figure 3.3 - middle), which are linked to ontologies in the Ontology Repository. In this case, we are using the EDAM ontology. Such terms are converted to a string with the form <**ontology**>**:**<**class identifier**> (Figure 3.3 - bottom). Note that the ontology terms *chromosome* and *DNA* use the same ontology class identifier because both terms belong to the same ontology class (see Listing 2.1).

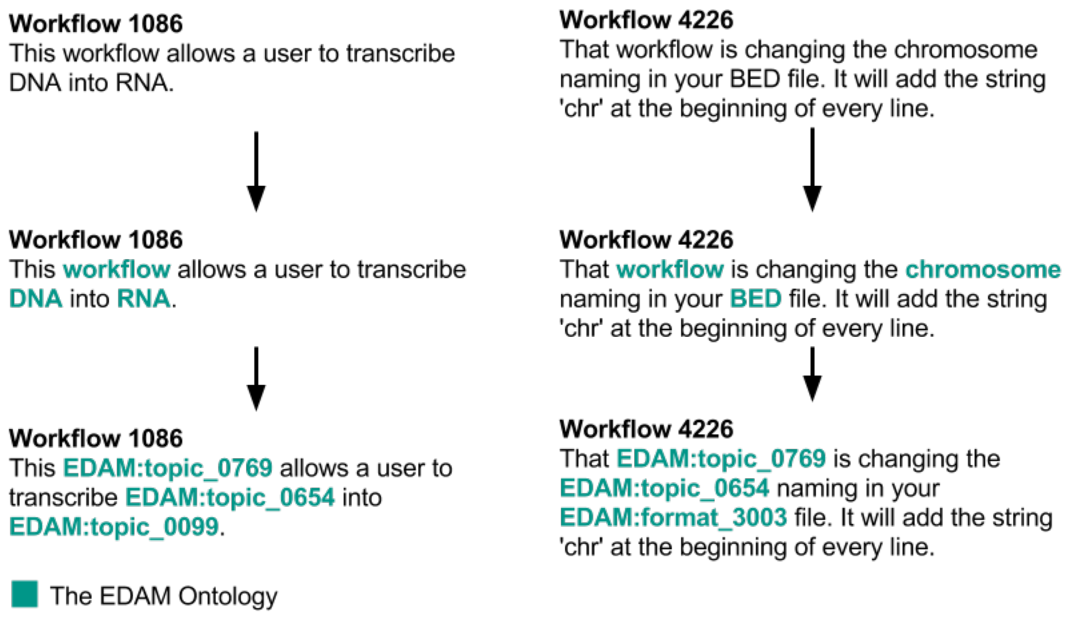
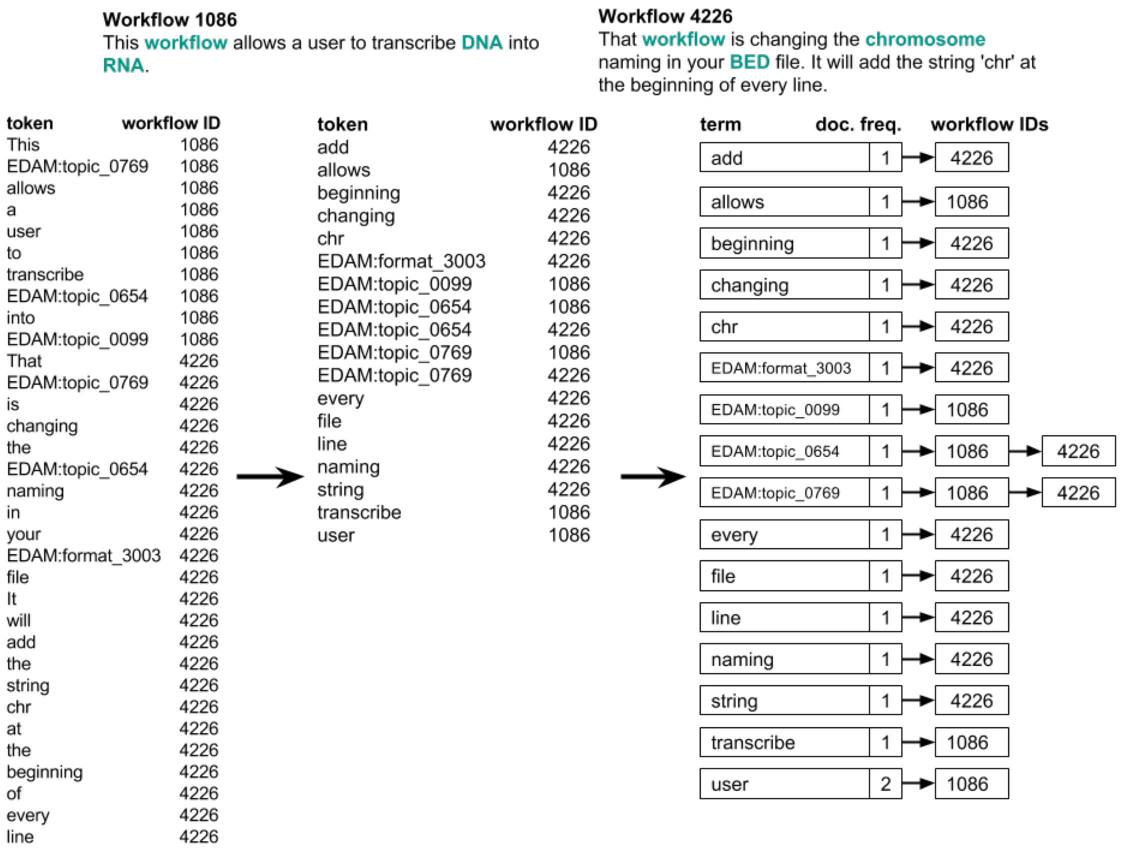

Workflow 1086
This workflow allows a user to transcribe DNA into RNA.
Workflow 4226
That workflow is changing the chromosome naming in your BED file. It will add the string 'chr' at the beginning of every line.
Workflow 1086
This workflow allows a user to transcribe DNA into RNA.
Workflow 4226
That workflow is changing the chromosome naming in your BED file. It will add the string 'chr' at the beginning of every line.
Workflow 1086
This EDAM:topic_0769 allows a user to transcribe EDAM:topic_0654 into EDAM:topic_0099.
Workflow 4226
That EDAM:topic_0769 is changing the EDAM:topic_0654 naming in your EDAM:format_3003 file. It will add the string 'chr' at the beginning of every line.
🟩 The EDAM Ontology

Figure 3.3: Example of Hybrid Text.

Figure 3.4 shows how the hybrid index is created. We use the same workflows that we used in Figure 3.3. This example takes into account just one metadata field (description). After the transformation of text into hybrid text (see Figure 3.3), the tokens in each document are tagged by their workflow identifier (Figure 3.4 - left). In this case, strings with the form <**ontology**>**:**<**class identifier**> and keywords are taken as a token. Next, the tokens are normalized and converted to indexing terms (Figure 3.4 - middle). In this case, the stop words *a*, *at*, *in*, *into*, *is*, *it*, *of*, *that*, *the*, *this*, *to*, *will*, and *your* were removed. Tokens were converted to lowercase (with exception of tokens with the form <**ontology**>**:**<**class identifier**>) and sorted alphabetically. Instances of the same indexing term are grouped by term and then by workflow identifier. Consequently, each indexing term has a list of workflow identifiers (Figure 3.4 - right).

[9]`http://www.myexperiment.org/workflows/1086.html`
[10]`http://www.myexperiment.org/workflows/4226.html`



Workflow 1086
This **workflow** allows a user to transcribe **DNA** into **RNA**.

Workflow 4226
That **workflow** is changing the **chromosome** naming in your **BED** file. It will add the string 'chr' at the beginning of every line.

token	workflow ID
This	1086
EDAM:topic_0769	1086
allows	1086
a	1086
user	1086
to	1086
transcribe	1086
EDAM:topic_0654	1086
into	1086
EDAM:topic_0099	1086
That	4226
EDAM:topic_0769	4226
is	4226
changing	4226
the	4226
EDAM:topic_0654	4226
naming	4226
in	4226
your	4226
EDAM:format_3003	4226
file	4226
It	4226
will	4226
add	4226
the	4226
string	4226
chr	4226
at	4226
the	4226
beginning	4226
of	4226
every	4226
line	4226

token	workflow ID
add	4226
allows	1086
beginning	4226
changing	4226
chr	4226
EDAM:format_3003	4226
EDAM:topic_0099	1086
EDAM:topic_0654	1086
EDAM:topic_0654	4226
EDAM:topic_0769	1086
EDAM:topic_0769	4226
every	4226
file	4226
line	4226
naming	4226
string	4226
transcribe	1086
user	1086

term	doc. freq.	workflow IDs	
add	1	4226	
allows	1	1086	
beginning	1	4226	
changing	1	4226	
chr	1	4226	
EDAM:format_3003	1	4226	
EDAM:topic_0099	1	1086	
EDAM:topic_0654	1	1086	4226
EDAM:topic_0769	1	1086	4226
every	1	4226	
file	1	4226	
line	1	4226	
naming	1	4226	
string	1	4226	
transcribe	1	1086	
user	2	1086	

Figure 3.4: Example of Hybrid Index.

3.4 Search Layer

Keyword search uses a four-step approach:

1. Tokenize the text in the query,
2. Get the list of workflows for each token in the *Keyword Index* that match with the tokens in the query,
3. Find the workflows that belong to the intersection of such lists,
4. Show results

Hybrid search uses the same approach. Nonetheless, in Step 1, ontology terms in the query are identified and replaced by strings with the form <**ontology**>**:**<**class identifier**>.

Although in Step 3 the system should retrieve all the workflows relevant to our query, it is useful to retrieve workflows that are in the interception of a percentage of the lists obtained in Step 2. In other words, when we write a query that has many words, it is useful to get workflows with metadata that match a high percentage of the tokens derived from the query, because they probably are relevant to our query.

In an ideal world, a full semantic search (using just semantic annotations) would be better than our approach using hybrid search (with a hybrid index). Nonetheless, ontologies evolve, losing and gaining ontology classes along time. Coverage of classes in most ontologies is incomplete because always there are missing elements in ontologies [23]. Moreover, there is a meaning loss when natural language is translated to semantic annotations because languages are semantically more expressive than ontologies [3]. Thus, we decided to combine keywords and semantic annotations to create a *Hybrid Index* because it expands the search options.

3.5 Final Remarks

In this chapter, we presented the WorkflowHunt architecture, including some details about semantic annotations, indexing, and search. In Chapter 4, we will present implementation aspects of WorkflowHunt. This architecture is generic and does not depend on specific technological solution.

Chapter 4

Implementation Aspects

In this chapter, we present details about the implementation of a WorkflowHunt prototype available at `http://www.workflowhunt.com`. This chapter explains the implementation of the Persistence Layer (Section 4.1), the Preprocessing Layer (Section 4.2), and the Search Layer (Section 4.3), the interface of the prototype (Section 4.4), and a case study (Section 4.5).

4.1 Implementing the Persistence Layer

Most repositories we implemented use MySQL as underlying DBMS. Figure 4.1 shows the repositories' schema. Tables *ontology* and *ontology_ class* belong to the Ontology Repository. Table *ontology_ term* stores the Dictionary of Ontology Terms, which is associated with ontology classes in the Ontology Repository. Tables *tag*, *tag_ wf*, and *workflow* belong to the Metadata Repository. The *semantic_ annotation* belong to the Repository of Semantic Annotations, which stores the workflow identifier, the metadata type of the field (e.g., title, description, or tag), the ontology class detected in the metadata, and the annotation type – e.g., direct annotation or semantic expansion (generalization, specialization, or distance expansion). Additionally, the system stores the distance between the original ontology class detected and the ontology class used in the annotation. Distance equals zero for direct annotations.

MyExperiment[1] is the External Scientific Workflow Repository. The prototype still does not use a Local Scientific Workflow Repository. Finally, we used ElasticSearch to create and store our keyword and hybrid indices. ElasticSearch [20] is a project based on Lucene and provides distributed and scalable search using inverted indexes, and full-text search or term-based search. The indices link to the information in the *workflow* table.

4.2 Implementing the Preprocessing Layer

Modules in the Preprocessing Layer were implemented using PHP and framework CodeIgniter. MyExperiment provides an API[2] to collect the workflow metadata (title, description, tags,

[1]`https://www.myexperiment.org/`
[2]http://wiki.myexperiment.org/index.php/Developer:API

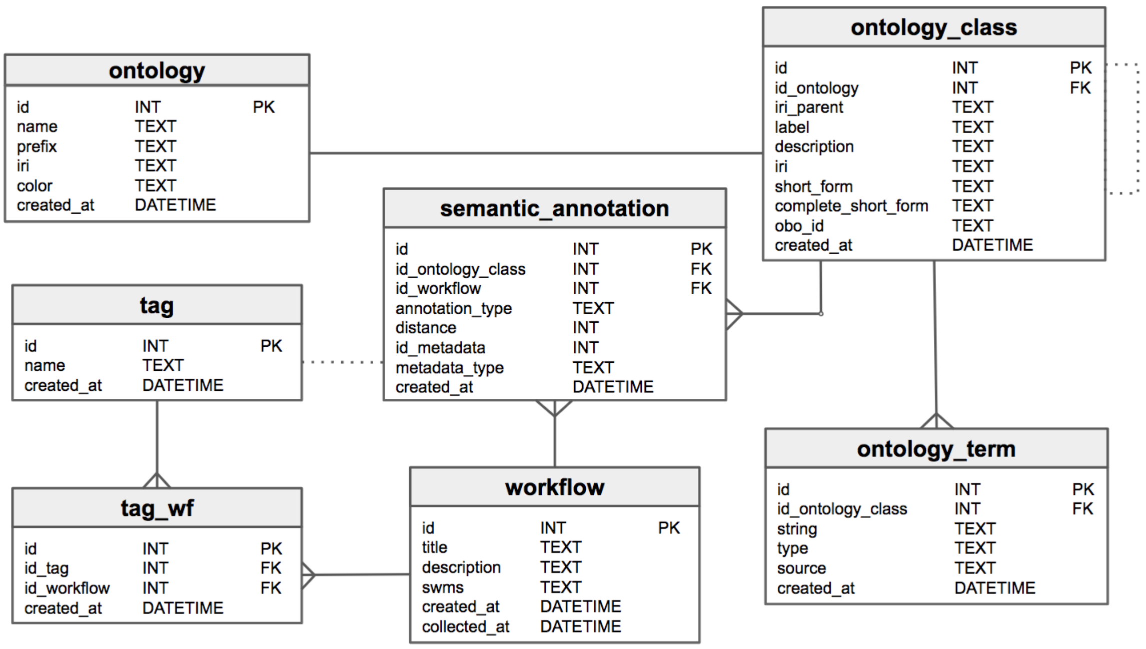
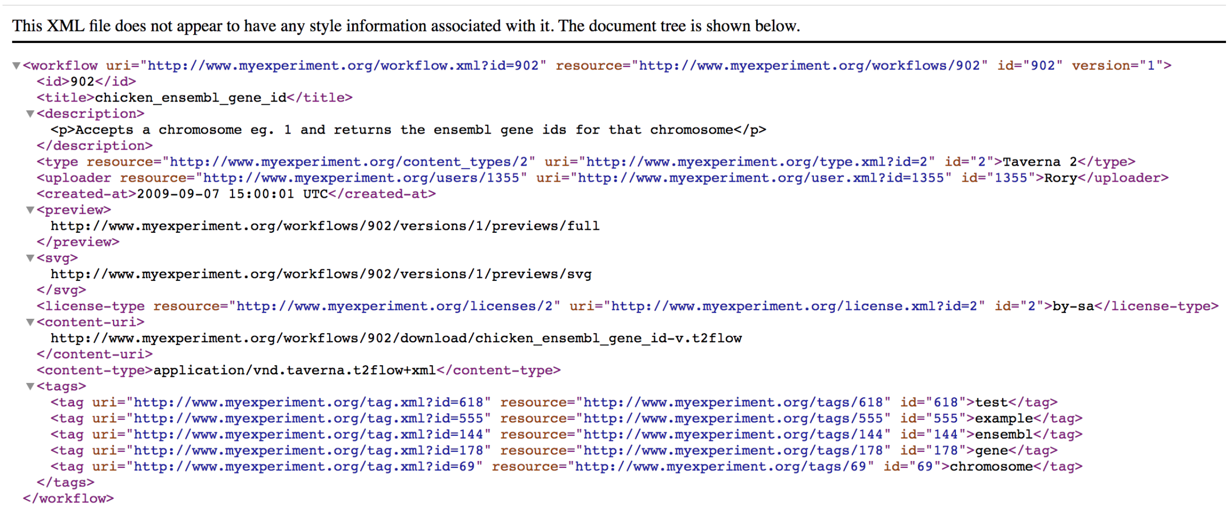


Figure 4.1: Database Schema for the Prototype.

etc.) from its repository in XML format. Listing 4.1 shows an example of metadata in XML for a workflow with title *chicken_ ensembl_ gene_ id*. The Web Crawler collects such information and the Metadata Collector transforms the raw metadata to store a subset of the available metadata (id, title, description, tags, and SWMS) in the table *workflow* (see Figure 4.1). Furthermore, tags are stored in the *tag* table and they are linked to the *workflow* table via the *tag_ wf* table. For instance, XML field *type* in Listing 4.1 will be stored in field *swms* of the table *workflow*. In total, we collected 15,054 metadata fields (title, description, SWMS and tags) that belong to 2,873 workflows.

Listing 4.1: Workflow Metadata in XML Format.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<workflow uri="http://www.myexperiment.org/workflow.xml?id=902" resource="http://www.myexperiment.org/workflows/902" id="902" version="1">
   <id>902</id>
   <title>chicken_ensembl_gene_id</title>
 ▼<description>
     <p>Accepts a chromosome eg. 1 and returns the ensembl gene ids for that chromosome</p>
   </description>
   <type resource="http://www.myexperiment.org/content_types/2" uri="http://www.myexperiment.org/type.xml?id=2" id="2">Taverna 2</type>
   <uploader resource="http://www.myexperiment.org/users/1355" uri="http://www.myexperiment.org/user.xml?id=1355" id="1355">Rory</uploader>
   <created-at>2009-09-07 15:00:01 UTC</created-at>
 ▼<preview>
     http://www.myexperiment.org/workflows/902/versions/1/previews/full
   </preview>
 ▼<svg>
     http://www.myexperiment.org/workflows/902/versions/1/previews/svg
   </svg>
   <license-type resource="http://www.myexperiment.org/licenses/2" uri="http://www.myexperiment.org/license.xml?id=2" id="2">by-sa</license-type>
 ▼<content-uri>
     http://www.myexperiment.org/workflows/902/download/chicken_ensembl_gene_id-v.t2flow
   </content-uri>
   <content-type>application/vnd.taverna.t2flow+xml</content-type>
 ▼<tags>
     <tag uri="http://www.myexperiment.org/tag.xml?id=618" resource="http://www.myexperiment.org/tags/618" id="618">test</tag>
     <tag uri="http://www.myexperiment.org/tag.xml?id=555" resource="http://www.myexperiment.org/tags/555" id="555">example</tag>
     <tag uri="http://www.myexperiment.org/tag.xml?id=144" resource="http://www.myexperiment.org/tags/144" id="144">ensembl</tag>
     <tag uri="http://www.myexperiment.org/tag.xml?id=178" resource="http://www.myexperiment.org/tags/178" id="178">gene</tag>
     <tag uri="http://www.myexperiment.org/tag.xml?id=69" resource="http://www.myexperiment.org/tags/69" id="69">chromosome</tag>
   </tags>
</workflow>
```

We used EDAM and CHEMINF ontologies to semantically annotate myExperiment workflow metadata. EDAM is an ontology for bioinformatics information, including operations, types of data, topics, and formats [25]. CHEMINF is an ontology for chemical information, including terms, and algorithms used in chemistry [22]. Those ontologies contain 3,045 ontology classes linked to a dictionary of 5,057 ontology terms. The ontology classes and terms (see Figure 4.1) were collected using the Ontology Lookup Service [8]. This service allows querying, browsing, and navigating over a database that integrates several biomedical ontologies and related vocabulary. We used WordNet [13] as an additional source of synonyms to complement the ontology terms collected via Ontology Lookup Service. Ontology classes are associated to one ontology and include the Internationalized Resource Identifier (IRI) and the IRI of its superclass for eventual generalization of semantic annotations (**our prototype only supports one level generalization among the semantic expansion options**). Ontology terms are stored in the table *ontology_term* and are associated with one ontology class. Such terms include a text, the term type (e.g., synonym, or labels), and the source (e.g., Ontology Lookup Service, or WordNet).

The Semantic Annotator uses the SEMANTIC_ANNOTATION algorithm (see Algorithm 2), which takes input data from *ontology_term*, *workflow*, *tag_wf*, and *tag* tables to populate the *semantic_annotation* table. The *semantic_annotation* table stores the ontology classes detected in workflow metadata. It also stores the metadata type of the field (e.g., title, description, or tag), the annotation type – e.g., direct annotation or semantic expansion (generalization, specialization, or distance expansion). For each annotation generated by semantic expansion, the system stores the distance between the original ontology class detected and the ontology class used in the annotation. Distance equals 1 for one level generalization (semantic annotations using the first superclass of the ontology class in a direct annotation). Distance equals zero for direct annotations. After the annotation process, we got 27,697 semantic annotations (51.5% by direct annotation and 48.5% by semantic expansion - one level generalization).

The Index Generator uses the BUILD_INDICES algorithm (see Algorithm 5), which takes input data from *semantic_annotation*, *workflow*, *tag_wf*, and *tag* tables to create the *Keyword* and *Hybrid* indices. Such inverted indices are created using ElasticSearch. The *Keyword Index* maps keywords to workflow identifiers, while the *Hybrid Index* maps keywords and ontology classes to workflow identifiers. The keywords belong to metadata of a workflow and the ontology classes are the ones stored in the *semantic_annotation* table, which are associated with a workflow. Finally, workflows identifiers link to the workflow metadata associated with them.

4.3 Implementing the Search Layer

Modules in this layer were implemented using PHP (with CodeIgniter Framework) and ElasticSearch. The user's query at the user interface (see Figure 4.2) is intercepted by the Log Manager. This information is used in the case study (see Section 4.5) to identify some cases where one search approach outperforms others. Users choose between *keyword* and *hybrid* search. The Search Engine takes the query and splits it in tokens – words when the user chooses keyword search and hybrid tokens when the user chooses hybrid search. Then, the Search Engine performs a search on the appropriate index according to user's choice (keyword or hybrid) and returns a list of metadata grouped by workflow.

Given a user's query string, we try to match it with at least 75% of the tokens extracted, considering they will return results relevant to the query. For instance, if the query string contains *"A B C D"* each of which is a token, then our results will be the set of workflows that satisfy at the same time at least 3 of these tokens.

4.4 User Interface

The user interface is inspired by modern search engines like Google, Bing, and DuckDuckGo because it provides a soft transition to hybrid search (semantic and keyword search). Figure 4.2 shows the home page of the WorkflowHunt prototype. The user poses the query (a string) at the input box; the search is triggered by clicking the search button or by pressing enter on the keyboard.

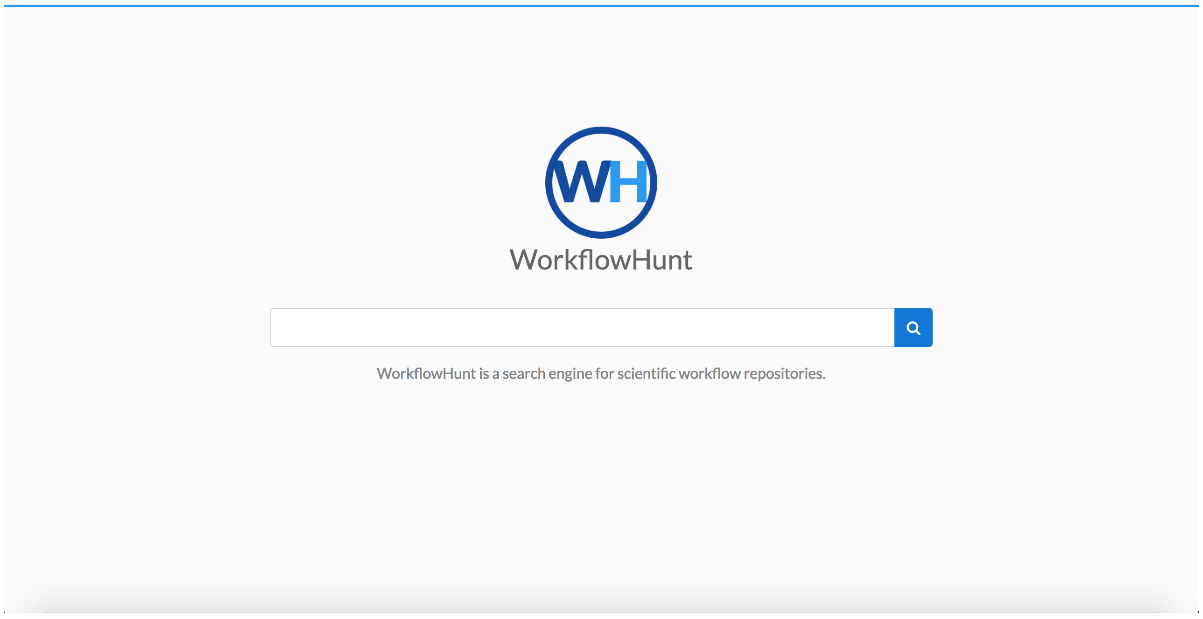
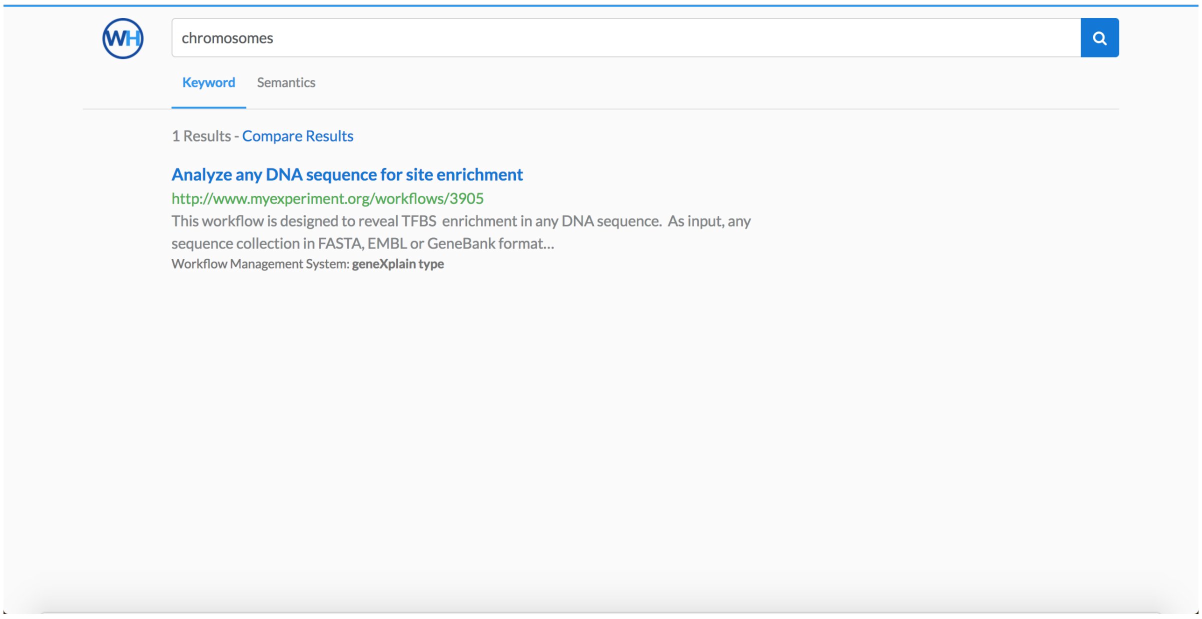

WH
WorkflowHunt
WorkflowHunt is a search engine for scientific workflow repositories.

Figure 4.2: Home Page of WorkflowHunt.

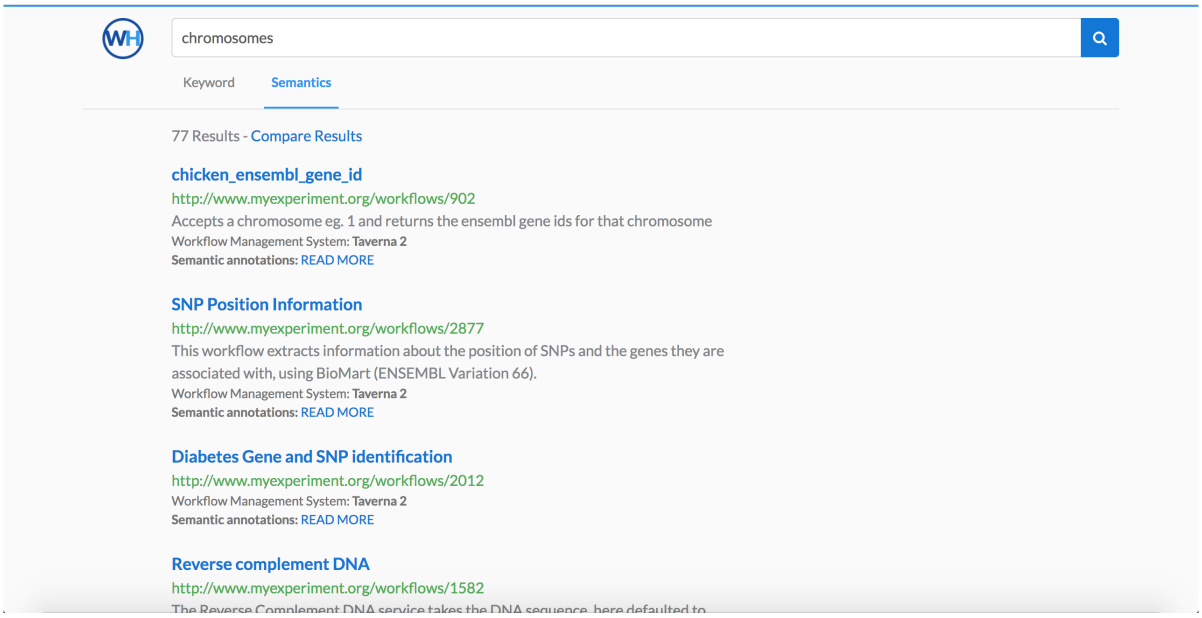
Figures 4.3 and 4.4 show results for the query *chromosomes* using keyword search (see Figure 4.3) and hybrid search (see Figure 4.4). The user chooses which kind of search s/he wants to perform by selecting the tab *Keyword* for keyword search or tab *Semantics* for hybrid search. The output shows the number of results retrieved and a link with label *Compare Results* – see Case Study (Section 4.5). Results are paginated and each page contains 10 results at most. Each result comprises four metadata fields: title, URL, description and Scientific Workflow Management System (SWMS). Link *READ MORE* displays the semantic annotations for the corresponding result (see Figure 4.6). Users can access a workflow using the URL in the results and download the workflow from myExperiment (see Figure 4.5).



1 Results - Compare Results

Analyze any DNA sequence for site enrichment
http://www.myexperiment.org/workflows/3905
This workflow is designed to reveal TFBS enrichment in any DNA sequence. As input, any sequence collection in FASTA, EMBL or GeneBank format...
Workflow Management System: **geneXplain type**

Figure 4.3: Keyword search results.



77 Results - Compare Results

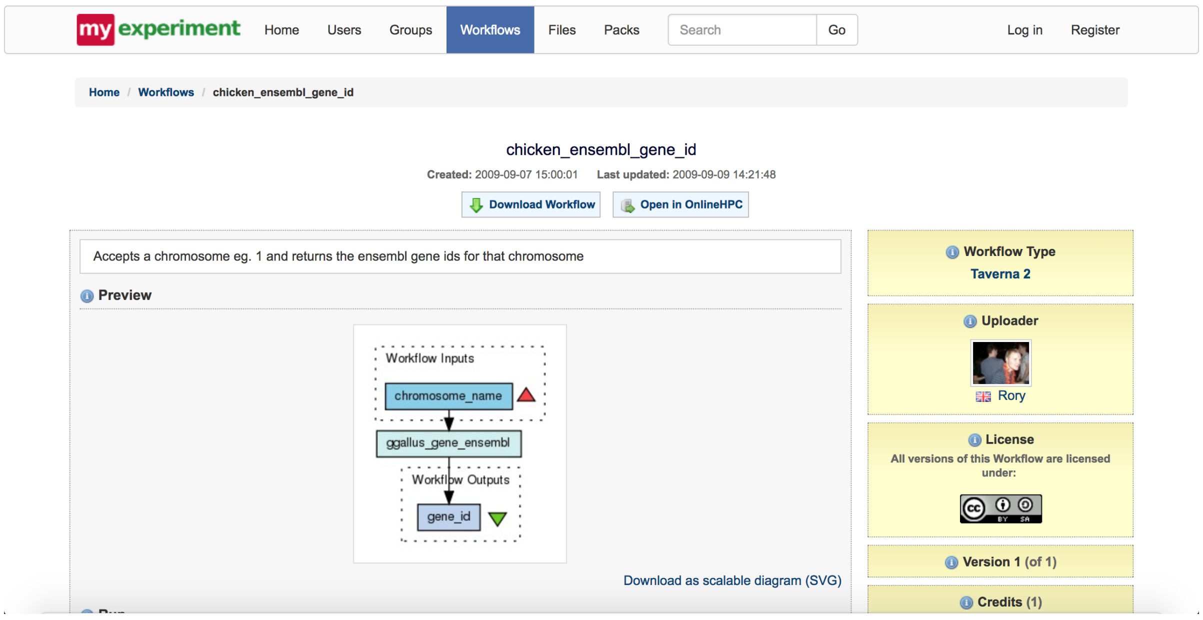
chicken_ensembl_gene_id
http://www.myexperiment.org/workflows/902
Accepts a chromosome eg. 1 and returns the ensembl gene ids for that chromosome
Workflow Management System: **Taverna 2**
Semantic annotations: READ MORE

SNP Position Information
http://www.myexperiment.org/workflows/2877
This workflow extracts information about the position of SNPs and the genes they are associated with, using BioMart (ENSEMBL Variation 66).
Workflow Management System: **Taverna 2**
Semantic annotations: READ MORE

Diabetes Gene and SNP identification
http://www.myexperiment.org/workflows/2012
Workflow Management System: **Taverna 2**
Semantic annotations: READ MORE

Reverse complement DNA
http://www.myexperiment.org/workflows/1582
The Reverse Complement DNA service takes the DNA sequence, here defaulted to

Figure 4.4: Hybrid search results (partial view).



chicken_ensembl_gene_id

Created: 2009-09-07 15:00:01 **Last updated:** 2009-09-09 14:21:48

Accepts a chromosome eg. 1 and returns the ensembl gene ids for that chromosome

Workflow Type Taverna 2

License All versions of this Workflow are licensed under:

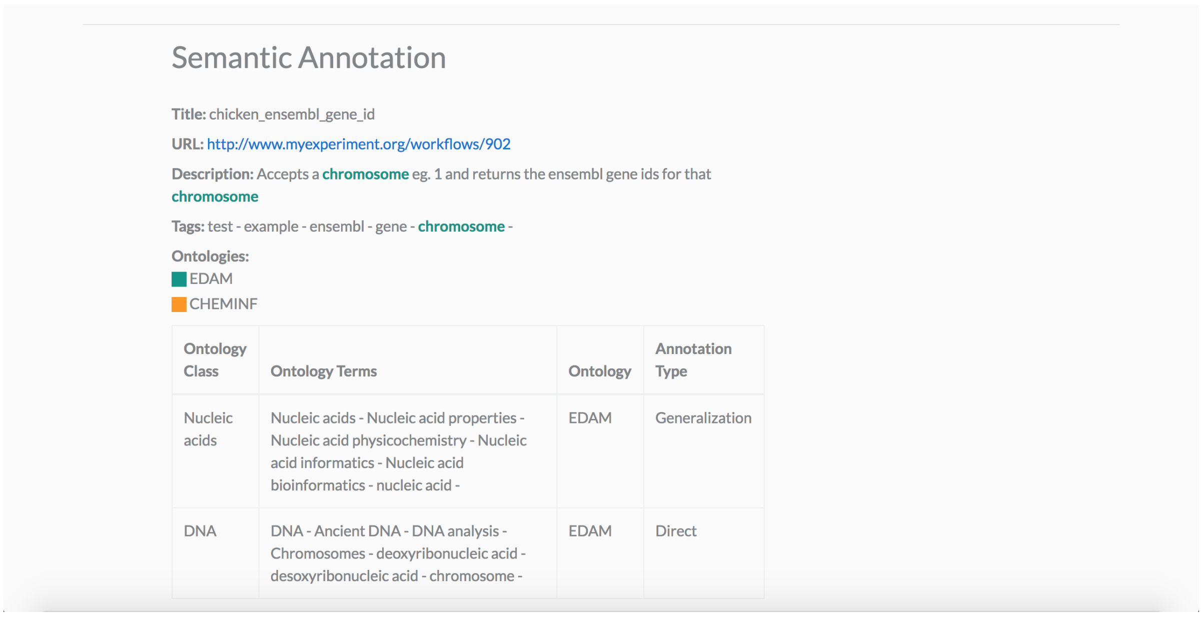
Figure 4.5: Workflow metadata in myExperiment.

Figure 4.6 shows the semantic annotation page for a workflow with title *chicken_ensembl_gene_id*, which is the first result for the query *chromosomes* when the user selects hybrid search (see Figure 4.4). This page presents four metadata fields: title, URL, description, and tags. The system performs semantic annotations on the title, description, and tags using the EDAM and CHEMINF ontologies. When the prototype detects an ontology term that belongs to one of the ontologies in the workflow metadata, the ontology term is highlighted and colored according to the ontology color. The table at the bottom of Figure 4.6 shows the ontology classes associated with the workflow metadata. The table has four headers: ontology class, ontology terms, ontology, and annotation type. Ontology terms are used to find exact string matches in the workflow metadata. Ontology classes are used to create the semantic annotation. Finally, the *Annotation Type* column shows if the semantic annotation is a direct annotation or a semantic expansion (generalization, specialization, or semantic distance expansion) of a semantic annotation.

4.5 Case Study

We compared WorkflowHunt (keyword search and hybrid search) with the retrieval system provided by myExperiment to analyze the cases for which it is convenient to use each system. We show three Venn diagrams (see Figures 4.8, 4.9, and 4.10) comparing different search approaches for different queries. Although three examples may seem insufficient, the diagrams illustrate the gist of our findings. A comparison table for more queries is available in the GitHub of the project [3].

[3]https://github.com/jbeleno/workflowhunt



Semantic Annotation

Title: chicken_ensembl_gene_id

URL: http://www.myexperiment.org/workflows/902

Description: Accepts a **chromosome** eg. 1 and returns the ensembl gene ids for that **chromosome**

Tags: test - example - ensembl - gene - **chromosome** -

Ontologies:
🟩 EDAM
🟧 CHEMINF

Ontology Class	Ontology Terms	Ontology	Annotation Type
Nucleic acids	Nucleic acids - Nucleic acid properties - Nucleic acid physicochemistry - Nucleic acid informatics - Nucleic acid bioinformatics - nucleic acid -	EDAM	Generalization
DNA	DNA - Ancient DNA - DNA analysis - Chromosomes - deoxyribonucleic acid - desoxyribonucleic acid - chromosome -	EDAM	Direct

Figure 4.6: Semantic annotations for a workflow in WorkflowHunt.

Figures 4.8, 4.9, and 4.10 show the Venn diagrams for the queries “chromosomes”, “ecology”, and “pathway simulation” respectively. We chose Venn diagrams instead of tables because they are better to analyze different cases where a system outperforms others.

Let q be a scientist’s query, and A, B, and C sets of workflows defined as follows:

A = set of workflows retrieved using the keyword search provided by myExperiment given the query q. This set is represented by a red circle in the Venn diagrams.

B = set of workflows retrieved using the keyword search provided by WorkflowHunt given the query q. This set is represented by a green circle in the Venn diagrams.

C = set of workflows retrieved using the semantic search provided by WorkflowHunt given the query q. This set is represented by a purple circle in the Venn diagrams.

Figure 4.7 presents a draft of a Venn diagram using the sets and query defined earlier. The numbers inside the Venn diagrams represent the cardinality (number of elements) of each subset. t is the number of elements that belong to A and do not belong to B or C. u is the number of elements in the intersection between A and B that do not belong to C. v is the number of elements that belong to B and do not belong A or C. w is the number of elements in the intersection between A and C that do not belong to B. x is the number of elements in the intersection among A, B, and C. y is the number of elements in the intersection of B and C that do not belong to A. z is the number of elements that belong to C and do not belong to A or B.

From the cardinality of these subsets, we can calculate the cardinality of each set. The cardinality of A is the sum of t, u, w, and x ($/A/ = t + u + w + x$). The cardinality of B is the sum of u, v, x, and y ($/B/ = u + v + x + y$). The cardinality of C is the sum of w, x, y, and z ($/C/ = w + x + y + z$).

More importantly for our case study, we can calculate the number of elements that belong to one set and do not belong to a specific set. The number of elements that belong to A and do not belong to B is the sum of t and w ($/A$ - $B/ = t + w$). The number of

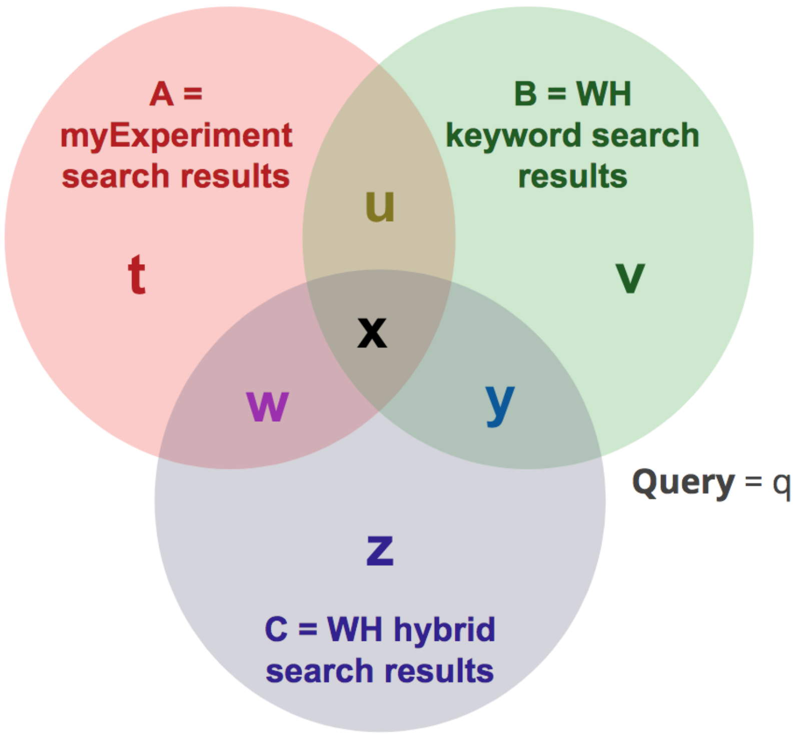

A = myExperiment search results
B = WH keyword search results
C = WH hybrid search results
t
u
v
x
w
y
z
Query = q

Figure 4.7: Draft of a Venn diagram comparing different search approaches for the query "q".

elements that belong to A and do not belong to C is the sum of t and u ($/A$ - $C/ = t + u$). The number of elements that belong to B and do not belong to A is the sum of v and y ($/B$ - $A/ = v + y$). The number of elements that belong to B and do not belong to C is the sum of u and v ($/B$ - $C/ = u + v$). The number of elements that belong to C and do not belong to A is the sum of y and z ($/C$ - $A/ = y + z$). The number of elements that belong to C and do not belong to B is the sum of w and z ($/C$ - $B/ = w + z$).

The search comparisons allow us to analyze six cases:

Case 1. $(A - B) \neq \emptyset$: This case occurs when, given a query, myExperiment returns workflows that keyword search in WorkflowHunt does not (see Figures 4.9 and 4.10). The main reason is that myExperiment indexes more metadata than WorkflowHunt[4]. For example, consider the query in Figure 4.9: "ecology", which produces 5 workflows that belong to $(A - B)$. After an analysis, we found that in most of them like the "Age specific analysis"[5] workflow contained the string "... Journal of Ecology" in the descriptions of some workflow inputs. At the moment, the prototype of WorkflowHunt just indexes title, description, and tags.

Case 2. $(B - C) \neq \emptyset$: This case occurs when a keyword search in WorkflowHunt returns workflows that the hybrid search in WorkflowHunt does not (see Figures 4.9 and 4.10). The main reason is that the hybrid search in WorkflowHunt replaces substrings in the query that are detected as ontology classes. Such substrings can have more than one word, which means that a hybrid search query can have fewer tokens than a keyword search in WorkflowHunt. Moreover, keyword search in WorkflowHunt does not consider

[4]https://goo.gl/STIQWR
[5]http://www.myexperiment.org/workflows/3286.html

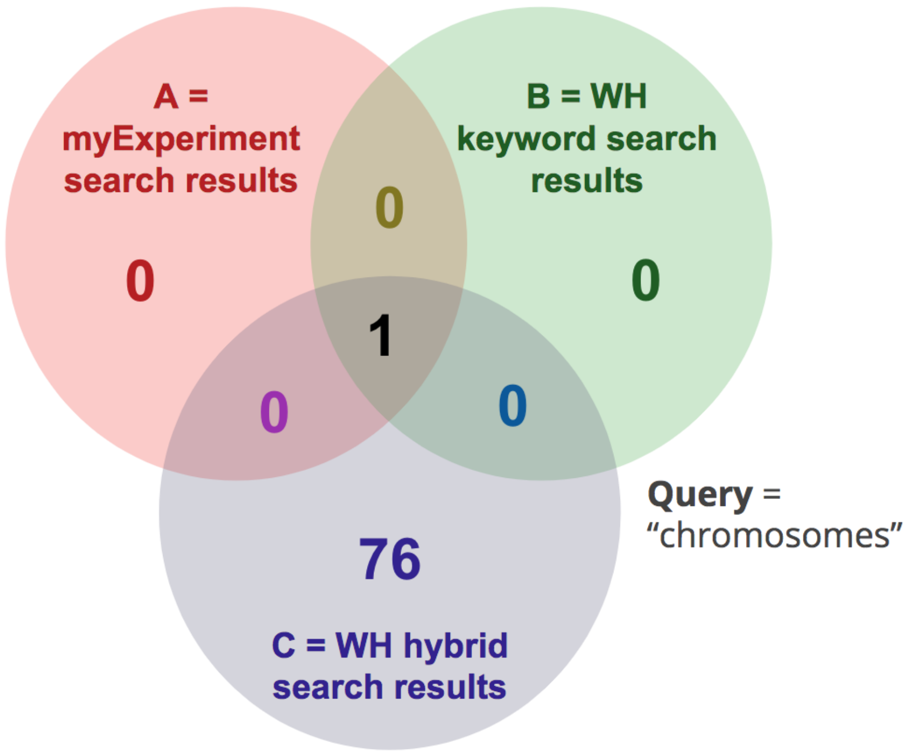


Figure 4.8: Venn diagram comparing different search approaches for the query "chromosomes".

the position of the words in the query, and semantic search in WorkflowHunt does. For example, consider the query "pathway simulation", which produces 140 workflows that belong to $(B - C)$. We found that there are no semantic annotations in WorkflowHunt associated to "pathway simulation" (ontology class EDAM:operation_3562). Nevertheless, keyword search in WorkflowHunt returned 140 workflows because the metadata of those workflows contains the words "pathway" and "simulation", but such words could appear in different order, in different metadata fields of the same workflow or could be separated by several words.

For the query "ecology", the result that belongs to $(B - C)$ exists because the Semantic Annotator has problems to identify ontology terms in the workflow metadata when they are near punctuation marks. The workflow[6] that belong to $(B - C)$ contains the text "...under the subject of Ecology." in the description field. That point after the word "Ecology" hindered the semantic annotation process of the ontology class "Ecology" (EDAM:topic_0610), which hides that workflow from results when results come from the hybrid search.

Case 3. $(A - C) \neq \emptyset$: This case occurs when a keyword search in myExperiment returns workflows that the hybrid search in WorkflowHunt does not. This case summarizes the reasons given in Case 1 and 2: myExperiment indexes more metadata fields, hybrid search in WorkflowHunt uses fewer tokens than keyword search, errors in semantic annotations affects hybrid search and keyword search in myExperiment has lax rules about the position of the words given in the query to match against the workflow metadata. This

[6]`http://www.myexperiment.org/workflows/2464.html`

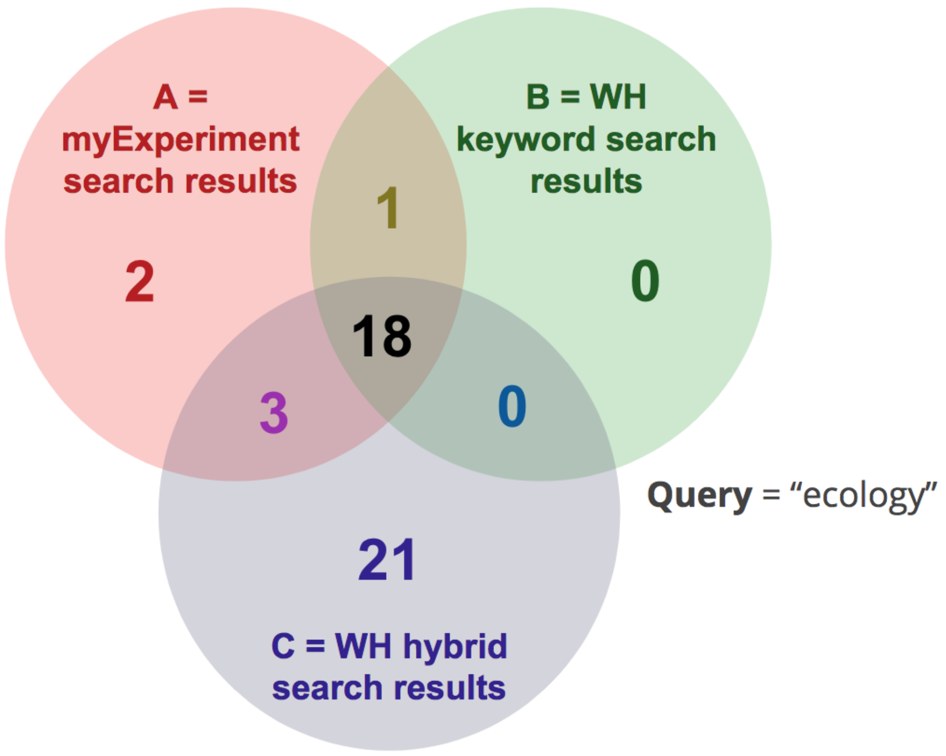


Figure 4.9: Venn diagram comparing different search approaches for the query "ecology".

case occurs with the queries "pathway simulation" (Figure 4.10) and "ecology" (Figure 4.9).

Case 4. $(B - A) \neq \emptyset$: This case occurs when a keyword search in WorkflowHunt returns workflows that the keyword search in myExperiment does not. The main reason is that WorkflowHunt has a lower threshold in the number of words that should be matched against the workflow metadata than the myExperiment keyword search. This case occurs with the query "pathway simulation" (Figure 4.10), where 123 workflows belong to $(B - A)$. For example, the workflow "Rank Phenotype Terms"[7] is in the set because it has the word "pathway" in the workflow metadata. However, it does not have the word "simulation" in the metadata.

Case 5. $(C - A) \neq \emptyset$: This case occurs when a hybrid search in WorkflowHunt returns workflows that the keyword search in myExperiment does not. The main reason is that hybrid search in WorkflowHunt allows finding concepts with similar meaning and not just keywords. Therefore, the results are semantically similar to what is searched. For example, consider the query "chromosomes" (Figure 4.8), where keyword search in myExperiment returns 1 workflow. The string "chromosomes" represents the ontology class EDAM:topic_0654 that is linked to many ontology terms: "DNA", "Ancient DNA", "DNA analysis", "Chromosomes", "deoxyribonucleic acid", "desoxyribonucleic acid", and "chromosome". Hence, hybrid search in WorkflowHunt finds workflows that match with those ontology terms in the workflow metadata, returning 77 workflows.

Case 6. $(C - B) \neq \emptyset$: This case occurs when a hybrid search in WorkflowHunt returns workflows that the keyword search in WorkflowHunt does not. The reason and example

[7]`http://www.myexperiment.org/workflows/854.html`

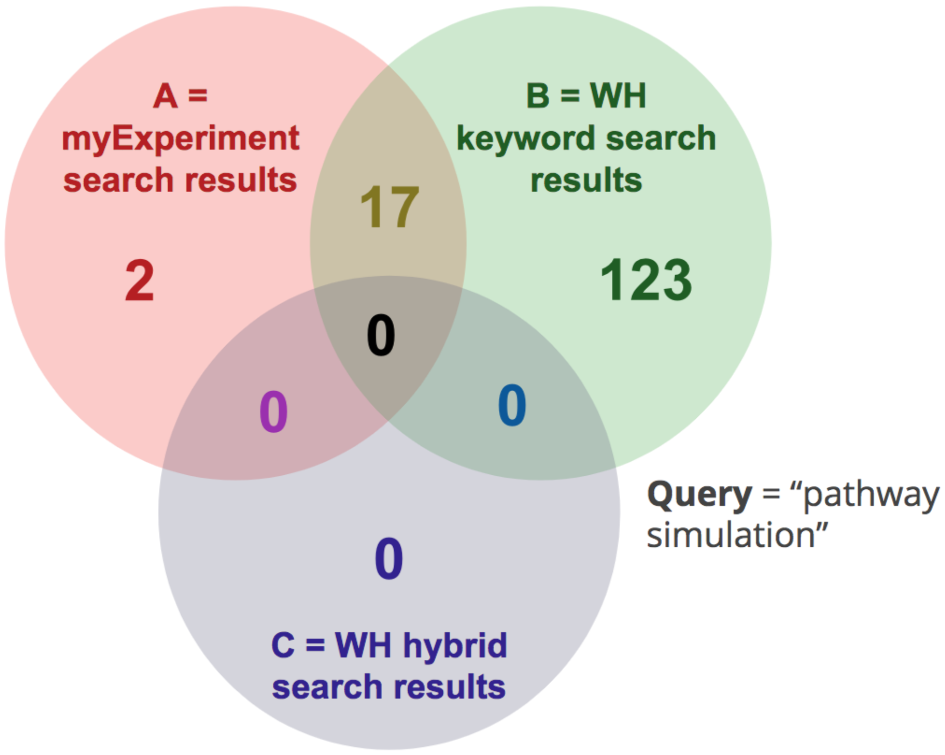

A = myExperiment search results
B = WH keyword search results
C = WH hybrid search results
2
17
123
0
0
0
0
Query = "pathway simulation"

Figure 4.10: Venn diagram comparing different search approaches for the query "pathway simulation".

are the same than the presented in Case 5.

The cases where the difference between sets is empty ($(A - B) = \emptyset$, $(A - C) = \emptyset$, $(B - C) = \emptyset$, $(C - B) = \emptyset$, $(C - A) = \emptyset$, and $(B - A) = \emptyset$) are not interesting because the systems return the same set of workflows for a query.

The conclusions given about set A are based on reverse engineering applied in myExperiment and a study of the source code of that website[8].

4.6 Final Remarks

In this chapter, we detailed implementation aspects of WorkflowHunt. Moreover, we presented a case study that shows where one search approach outperforms others, comparing myExperiment search engine and WorkflowHunt prototype. However, there is no data about precision and recall of the prototype. For more details about the WorkflowHunt prototype please visit `https://github.com/jbeleno/workflowhunt`, which contains a GitHub repository with the code of the prototype and some instructions for setup.

Summing up, the main differences found between our solution and the keyword search provided by myExperiment are:

- WorkflowHunt indexes fewer metadata fields than myExperiment;
- WorkflowHunt has a static minimum threshold for the number of tokens matched

[8]https://github.com/myExperiment/myExperiment

with workflow metadata to classify a workflow as relevant given a query, while myExperiment seems to have a flexible threshold;

- WorkflowHunt expands the search using ontology terms associated with the ontology classes detected in the query, while myExperiment uses just the keywords in the query;
- Missing semantic annotations in workflow metadata hidden relevant results in WorkflowHunt, while myExperiment does not have such problem because it uses keyword search;
- myExperiment considers a workflow relevant if the keywords in the query appear in the workflow metadata in any order when some keywords in the query are detected as ontology class in WorkflowHunt, they should appear in the same order as they appear in the query.

Chapter 5 presents contributions and future work.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Workflow reuse helps accelerate the design of scientific experiments. However, workflow repositories do not provide enough search tools and help scientists find relevant workflows. This work is a step towards this direction. It presented the design architecture and implementation of WorkflowHunt, a retrieval system that combines semantic and keyword search in scientific workflow repositories. It aims to contribute helping scientists in workflow reuse, re-purpose, and experiment replication by facilitating the discovery of workflows relevant to scientists' experiments. Our work is a step towards helping scientists find and select the most appropriate workflows for reuse by a given experiment.

We created a prototype[1] to validate our proposal using data from myExperiment, [11], one of the largest scientific workflow repositories at the moment. We also used EDAM and CHEMINF ontologies to create semantic annotations, which serve as inputs for the hybrid search. EDAM is an ontology of bioinformatics information, including operations, types of data, topics, and formats [25]. CHEMINF is an ontology for chemical information, including terms, and algorithms used in chemistry [22]. WorkflowHunt combines semantic and keyword search. It is extensible, and can accept more ontologies from the Ontology Lookup Service. This service allows querying, browsing, and navigating over a database that integrates several biomedical ontologies and related vocabulary [8]. Moreover, including new workflow metadata from other repositories would require few modifications in the source code.

The main contributions of our work are:

- A study comparing different scientific workflow repositories and their retrieval systems;
- The design of a generic architecture for workflow retrieval combining keyword and semantic search;
- The implementation of a prototype for WorkflowHunt, which can run on any repository that export workflow metadata, for arbitrary workflows and ontologies;

[1]`http://www.workflowhunt.com`

- A case study that shows cases where our system outperforms myExperiment and vice versa

Our proposal and its implementation are generic and not restricted to a specific domain or ontology. Moreover, the construction of our indices and the annotations are automatics. This is an advantage over related work, which is either restricted to specific workflow platform or application domains and/or depends on manual annotations.

5.2 Future Work

There are many possible extension to our work:

- **Semantic Annotation Enhancement:** Our algorithm to extract semantic annotations from text uses an exact string comparison with labels and synonyms in the ontologies of the Ontology Repository. This is a simple approach that can be improved using more advanced NLP techniques. One problem that needs to be solved is to identify which ontologies should be used to annotate a workflow given that there are many workflows from many scientific domains.

- **Structure-based Search:** This is the only search approach that was not considered in our retrieval system. Usually, this approach takes much time to compute structure similarities, so it is necessary to balance precision and usability.

- **Suitability Study:** Our case study presents cases to justify why one system outperforms others in terms of number of workflows. Nonetheless, we did not perform a study about suitability in terms of precision and recall. This is because the task of measuring such metrics is complex when the number of workflows is high. Moreover, this requires scientists that use workflow in their research from different scientific domains and with time to evaluate different retrieval systems.

- **Using instances of ontology classes:** A instance, in the context of ontologies, represents an element in the domain attached to a specific concept (ontology class) [19]. Usually, ontologies do not contain instances, but they can be derived from specialized sources in each domain. Such instances can be used to have more elements to extract semantic annotation from text, resulting in a better performance of the hybrid search.

- **Implementation with n levels of semantic expansion:** Our prototype uses just one level generalization and a relational database. This implementation works well, but it can grow just a few degrees of semantic expansion (specialization, generalization and distance expansion) before becoming unusable because of large response times for queries. A solution to this problem would be the implementation of WorkflowHunt architecture using graph databases or databases specialized in ontologies and semantic annotations.

- **Ranking results for hybrid search:** We do not consider ranking at the moment of retrieving results in the hybrid search. This task should consider many aspects like the ratio between the tokens in the query and the workflow metadata, the correlation among different metadata fields, the importance of each metadata field, etc.

- **Ontology evolution:** as ontologies evolve, the annotations need to be refreshed. This is an open problem and depends on users asking the system to periodically re-annotate using the new ontology version. In our present implementation, this requires previously deleting annotations.

Bibliography

[1] Hannah Bast, Björn Buchhold, Elmar Haussmann, et al. Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10(2-3):119–271, 2016.

[2] Ralph Bergmann and Yolanda Gil. Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, 40:115–127, 2014.

[3] Kent D Bimson, Richard D Hull, and Daniel Nieten. The lexical bridge: A methodology for bridging the semantic gaps between a natural language and an ontology. In *Semantic Web*, pages 137–151. Springer, 2016.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227, 2009.

[5] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[6] Lucas AMC Carvalho, Khalid Belhajjame, and Claudia Bauzer Medeiros. Converting scripts into reproducible workflow research objects. In *e-Science (e-Science), 2016 IEEE 12th International Conference on*, pages 71–80. IEEE, 2016.

[7] Sarah Cohen-Boulakia and Ulf Leser. Search, adapt, and reuse: the future of scientific workflows. *ACM SIGMOD Record*, 40(2):6–16, 2011.

[8] Richard G Côté, Philip Jones, Rolf Apweiler, and Henning Hermjakob. The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC bioinformatics*, 7(1):97, 2006.

[9] Víctor Cuevas-Vicenttín, Parisa Kianmajd, Bertram Ludäscher, Paolo Missier, Fernando Chirigati, Yaxing Wei, David Koop, and Saumen Dey. The pbase scientific workflow provenance repository. *International Journal of Digital Curation*, 9(2):28–38, 2014.

[10] Jaudete Daltio. Aonde: um serviço web de ontologias para interoperabilidade em sistemas de biodiversidade. Master's thesis, Instituto de Computação - Unicamp, 2007. Supervisor Claudia Bauzer Medeiros.

[11] David De Roure, Carole Goble, and Robert Stevens. The design and realisation of the virtual research environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5):561–567, 2009.

[12] J. S. B. Diaz and C. B. Medeiros. Workflowhunt: Combining keyword and semantic search in scientific workflow repositories. In *2017 IEEE 13th International Conference on e-Science (e-Science)*, pages 138–147, Oct 2017.

[13] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[14] Beatriz García-Jiménez and Mark D Wilkinson. Automatic annotation of bioinformatics workflows with biomedical ontologies. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 464–478. Springer, 2014.

[15] Daniel Garijo and Yolanda Gil. Towards open publication of reusable scientific workflows: Abstractions, standards and linked data. *Internal Project Report*, 2012.

[16] Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro Gonzalez-Calero, Paul Groth, Joshua Moody, and Ewa Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72, 2011.

[17] Carole A Goble, Jiten Bhagat, Sergejs Aleksejevs, Don Cruickshank, Danius Michaelides, David Newman, Mark Borkum, Sean Bechhofer, Marco Roos, Peter Li, et al. myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic acids research*, 38(suppl 2):W677–W682, 2010.

[18] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, 2010.

[19] Asunción Gómez-Pérez and Oscar Corcho. Ontology languages for the semantic web. *IEEE Intelligent systems*, 17(1):54–60, 2002.

[20] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.", 2015.

[21] Thomas R Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[22] Janna Hastings, Leonid Chepelev, Egon Willighagen, Nico Adams, Christoph Steinbeck, and Michel Dumontier. The chemical information ontology: provenance and disambiguation for chemical data on the biological semantic web. *PloS one*, 6(10):e25513, 2011.

[23] Martin Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing*, 11(1), 2007.

[24] Tony Hey and Mike C Payne. Open science decoded. *Nature Physics*, 11(5):367–369, 2015.

[25] Jon Ison, Matúš Kalaš, Inge Jonassen, Dan Bolser, Mahmut Uludag, Hamish McWilliam, James Malone, Rodrigo Lopez, Steve Pettifer, and Peter Rice. Edam: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, 29(10):1325–1332, 2013.

[26] Clement Jonquet, Nigam Shah, and Mark Musen. The open biomedical annotator. In *AMIA summit on translational bioinformatics*, pages 56–60, 2009.

[27] Soner Kara, Özgür Alan, Orkunt Sabuncu, Samet Akpınar, Nihan K Cicekli, and Ferda N Alpaslan. An ontology-based retrieval system using semantic indexing. *Information Systems*, 37(4):294–305, 2012.

[28] Janez Kranjc, Vid Podpečan, and Nada Lavrač. Clowdflows: a cloud based scientific workflow platform. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 816–819. Springer, 2012.

[29] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[30] Bertram Ludäscher, Shawn Bowers, and Timothy McPhillips. Scientific workflows. *Encyclopedia of Database Systems*, pages 2507–2511, 2009.

[31] Carla Geovana N Macário, Sidney Roberto de Sousa, and Claudia Bauzer Medeiros. Annotating geospatial data based on its semantics. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 81–90. ACM, 2009.

[32] Christoph Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34, 2007.

[33] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[34] Phillip Mates, Emanuele Santos, Juliana Freire, and Cláudio T Silva. Crowdlabs: Social analysis and visualization for the sciences. In *International Conference on Scientific and Statistical Database Management*, pages 555–564. Springer, 2011.

[35] Luc Moreau, Juliana Freire, Joe Futrelle, Robert E McGrath, Jim Myers, and Patrick Paulson. The open provenance model: An overview. In *International Provenance and Annotation Workshop*, pages 323–326. Springer, 2008.

[36] Eyal Oren, Knud Möller, Simon Scerri, Siegfried Handschuh, and Michael Sintek. What are semantic annotations. *Relatório técnico. DERI Galway*, 9:62, 2006.

[37] Roger Peng. The reproducibility crisis in science: A statistical counterattack. *Significance*, 12(3):30–32, 2015.

[38] Qihong Shao, Peng Sun, and Yi Chen. Wise: A workflow information search engine. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1491–1494. Ieee, 2009.

[39] Johannes Starlinger, Bryan Brancotte, Sarah Cohen-Boulakia, and Ulf Leser. Similarity search for scientific workflows. *Proceedings of the VLDB Endowment*, 7(12):1143–1154, 2014.

[40] Johannes Starlinger, Sarah Cohen-Boulakia, Sanjeev Khanna, Susan B Davidson, and Ulf Leser. Effective and efficient similarity search in scientific workflow repositories. *Future Generation Computer Systems*, 56:584–594, 2016.

[41] J. Wainer, M. Weske, G. Vossen, and C. B. Medeiros. Scientific workflow systems. In *Proc. NSF Workshop on Workflow and Process Automation: State-of-the-art and Future Directions*, Athens, GA, 1996.

[42] 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(W1):W557–W561, 2013.

[43] Liyang Yu. *A developer's guide to the semantic Web*. Springer Science & Business Media, 2011.