Rafael de Oliveira Werneck

"A unified framework for design, deployment, execution, and recommendation of machine learning experiments"

"Uma ferramenta unificada para projeto, desenvolvimento, execução e recomendação de experimentos de aprendizado de máquina"

CAMPINAS
2014

UNICAMP

University of Campinas
Institute of Computing

Universidade Estadual de Campinas
Instituto de Computação

Rafael de Oliveira Werneck

"A unified framework for design, deployment, execution, and recommendation of machine learning experiments"

Supervisor: *Orientador(a):* Prof. Dr. Ricardo da Silva Torres

Co-Supervisor: *Co-orientador(a):* Prof. Dr. Anderson de Rezende Rocha

"Uma ferramenta unificada para projeto, desenvolvimento, execução e recomendação de experimentos de aprendizado de máquina"

MSc Dissertation presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a Mestre degree in Computer Science.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

This volume corresponds to the final version of the Dissertation defended by Rafael de Oliveira Werneck, under the supervision of Prof. Dr. Ricardo da Silva Torres.

Este exemplar corresponde à versão final da Dissertação defendida por Rafael de Oliveira Werneck, sob orientação de Prof. Dr. Ricardo da Silva Torres.

Ricardo Torres

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

W495u

Werneck, Rafael de Oliveira, 1989-
A unified framework for design, deployment, execution, and recommendation of machine learning experiments / Rafael de Oliveira Werneck. – Campinas, SP : [s.n.], 2014.

Orientador: Ricardo da Silva Torres.
Coorientador: Anderson de Rezende Rocha.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Aprendizado de máquina. 2. Ciência - Experiências. 3. Fluxo de trabalho. 4. Sistemas de recomendação. I. Torres, Ricardo da Silva,1977-. II. Rocha, Anderson de Rezende,1980-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Uma ferramenta unificada para projeto, desenvolvimento, execução e recomendação de experimentos de aprendizado de máquina
Palavras-chave em inglês:
Machine learning
Science - Experiments
Workflow
Recommendation systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ricardo da Silva Torres [Orientador]
André Santanchè
José Gustavo de Souza Paiva
Data de defesa: 22-08-2014
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Rafael de Oliveira Werneck**, aprovado(a) em **22 de agosto de 2014**, pela Banca examinadora composta pelos Professores Doutores:

[illegible]

Prof(a). Dr(a). José Gustavo de Souza Paiva
Titular

André Santanchè

Prof(a). Dr(a). André Santanchè
Titular

Ricardo Torres

Prof(a). Dr(a). Ricardo da Silva Torres
Presidente

Institute of Computing / *Instituto de Computação*
University of Campinas / *Universidade Estadual de Campinas*

A unified framework for design, deployment, execution, and recommendation of machine learning experiments

Rafael de Oliveira Werneck[1]

August 22, 2014

[1]Financial support: CAPES scholarship (2012) and Samsung Eletrônica da Amazônia Ltda., through the project "Pattern recognition and classification by feature engineering, *-fusion, open-set recognition, and meta-recognition" in the framework of law No. 8248/91.

Abstract

Due to the large growth of the use of technologies for data acquisition, we have to handle large and complex data sets in order to extract knowledge that can support the decision-making process in several domains. A typical solution for addressing this issue relies on the use of machine learning methods, which are computational methods that extract useful knowledge from experience to improve performance of target applications. There are several libraries and frameworks in the literature that support the execution of machine learning experiments. However, some of them are not flexible enough for being extended with novel methods and they do not support reusing of successful solutions devised in previous experiments made in the framework. In this work, we propose a framework for automating machine learning experiments that provides a workflow-based standardized environment and makes it easy to evaluate different feature descriptors, classifiers, and fusion approaches in a wide range of tasks. We also propose the use of similarity measures and learning-to-rank methods in a recommendation scenario, in which users may have access to alternative machine learning experiments. We performed experiments with four similarity measures (Jaccard, Sørensen, Jaro-Winkler, and a TF-IDF-based measure) and one learning-to-rank method (LRAR) in the task of recommending workflows modeled as a sequence of activities. Experimental results show that Jaro-Winkler yields the highest effectiveness performance with comparable results to those observed for LRAR. In both cases, the recommendations performed are very promising and might help real-world users in different daily machine learning tasks.

Resumo

Devido ao grande crescimento do uso de tecnologias para a aquisição de dados, temos que lidar com grandes e complexos conjuntos de dados a fim de extrair conhecimento que possa auxiliar o processo de tomada de decisão em diversos domínios de aplicação. Uma solução típica para abordar esta questão se baseia na utilização de métodos de aprendizado de máquina, que são métodos computacionais que extraem conhecimento útil a partir de experiências para melhorar o desempenho de aplicações-alvo. Existem diversas bibliotecas e arcabouços na literatura que oferecem apoio à execução de experimentos de aprendizado de máquina, no entanto, alguns não são flexíveis o suficiente para poderem ser estendidos com novos métodos, além de não oferecerem mecanismos que permitam o reuso de soluções de sucesso concebidos em experimentos anteriores na ferramenta. Neste trabalho, propomos um arcabouço para automatizar experimentos de aprendizado de máquina, oferecendo um ambiente padronizado baseado em *workflow*, tornando mais fácil a tarefa de avaliar diferentes descritores de características, classificadores e abordagens de fusão em uma ampla gama de tarefas. Também propomos o uso de medidas de similaridade e métodos de *learning-to-rank* em um cenário de recomendação, para que usuários possam ter acesso a soluções alternativas envolvendo experimentos de aprendizado de máquina. Nós realizamos experimentos com quatro medidas de similaridade (Jaccard, Sørensen, Jaro-Winkler e baseada em TF-IDF) e um método de *learning-to-rank* (LRAR) na tarefa de recomendar *workflows* modelados como uma sequência de atividades. Os resultados dos experimentos mostram que a medida Jaro-Winkler obteve o melhor desempenho, com resultados comparáveis aos observados para o método LRAR. Em ambos os casos, as recomendações realizadas são promissoras, e podem ajudar usuários reais em diferentes tarefas de aprendizado de máquina.

Acknowledgements

First and foremost, I would like to thank God, whose many blessings have made me who I am today. I would like to thank my parents and siblings, who supported me in every step of this journey. My family, who prayed for me this whole time, despite my distance these last days.

I would like to thank my advisors, Ricardo da Silva Torres and Anderson de Rezende Rocha, who inspired me to always seek something more. My friends from the Samsung project (Waldir, Pedro, Otávio, Bernardo, and Daniel), that always helped me.

I also would like to thank my friends, all of them, that even with the distance, always gave me strength to keep moving forward.

Finally, I would like to thank CAPES and Samsung Eletrônica da Amazônia Ltda. for the financial support.

"kitto ano sora wa miteta ne
nando mo tsumazuita koto
soredemo saigo made aruketa koto
(I'm sure, the sky above us saw it all
All the times we tripped
But also all the times we walked on till the end)"

Tenshi ni Fureta yo!
(Touched by an Angel!)
Inaba Emi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, due to several technologies to acquire and store data, we have to handle large and complex data sets that are difficult to process using traditional data analysis tools, as such, for some scenarios, it is too big, or it exceeds the current processing capacity. For example, the Sloan Digital Sky Survey (SDSS)[1] collects 200GB of astronomical data per night, storing a total of 140TB of information [37]; in the imaging hosting website Flickr[2], about 1.42 and 1.6 million of public photos were uploaded per day in 2012 and 2013, respectively [80]; and the NASA Center for Climate Simulations (NCCS)[3] stores 32PB of climate observations and simulations.[4]

With this vast amount of data, it is necessary to devise appropriate computational tools for extracting information that can lead to knowledge acquisition. This knowledge extraction process is usually performed by means of data mining and machine learning methods. The ultimate goal is to improve the decision-making process in a target application. Decision making is the practice of basing decision on the analysis of data [95]. For example, scientists use data mining techniques to find unusual patterns from text and visual content, while doctors can make decisions using similar cases from the past.

A typical machine learning solution comprises several steps, including, for example, feature extraction and normalization methods, and the definition of appropriate classifiers. Since there is no *silver bullet* that solves all machine learning problems, each technique has its own pros and cons when designed for specific applications. In this sense, one common strategy adopted for developers of machine learning systems consists in performing several experiments with the objective of identifying which techniques are more appropriate for a given application.

[1]`http://www.sdss.org/` — as of July 2014.

[2]`https://www.flickr.com/` — as of July 2014.

[3]`http://www.nccs.nasa.gov/` — as of July 2014.

[4]`http://www.csc.com/cscworld/publications/81769/81773-supercomputing_the_climate_nasa_s_big_data_mission` — as of July 2014

Several libraries and machine learning frameworks have been proposed in the literature to support users in the process of defining the most appropriate methods for their applications. However, these frameworks have some flaws, as they are not flexible enough for being extended with novel proposed descriptors or machine learning methods. Also, another important issue concerns the identification and the reuse of successful solutions devised in the past.

In this work, we address these issues by modeling scientific machine learning experiments as workflows. Workflow is the automation of a process, in which information is passed from one resource to another for action, according to a set of rules. The advantages of using workflows are that they are easily understandable, flexible, and reproducible, in which it is possible to redesign them and reproduce their results. The objective of this work is to specify and implement a workflow-based framework that can be used for designing, deploying, executing, and recommending machine learning experiments. This framework is able to provide a standardized environment, making it easy to evaluate different feature descriptors, normalizers, classifiers, fusion approaches in a wide range of tasks involving machine learning.

We also included in the framework a service for recommending machine learning workflows. This service is very important, even for experienced users, but specially for beginners in machine learning, as it may guide the user during the configuration of an experiment when facing new and challenging classification problems. We also performed experiments in the recommendation system aiming at evaluating four similarity measures (Jaccard, Sørensen, Jaro-Winkler, and a TF-IDF-based measure) in order to define which one is more appropriate for ranking workflows. We also performed experiments with the Learning to Rank using Association Rules (LRAR) method with the objective of comparing its accuracy performance with the methods that do not use any learning mechanism. We concluded that the LRAR method and the Jaro-Winkler similarity measure have the best performance in the recommendation, being not significantly different from each other.

In summary, the main contributions of this work are:

1. Specification and implementation of a workflow-based standardized framework for devising and running machine learning experiments;

2. Proposal of a workflow recommendation service that relies on the use of similarity measures that rank machine learning experiments modeled as a sequence of activities; and

3. Discussion upon the use of learning-to-rank methods for ranking workflows.

This dissertation is organized as follow: Section 2 presents some concepts used and related work in workflow management, machine learning framework, and recommendation systems. Section 3 describes the proposed machine learning framework and how it was implemented. A case study is presented in Section 4, in which the use of the framework is shown. This section also presents the extensibility of the implemented framework with another workflow management system, and an overview of the recommendation system framework execution, with the experiments performed to evaluate different similarity measures used in our recommendation system. Finally, Section 5 presents our conclusions and proposes important research directions for future work.

Chapter 2

Concepts and Related Work

This chapter presents the main concepts used in this dissertation, as well as some related work. Section 2.1 defines concepts related to machine learning, presenting a typical machine learning experiment, and related work on machine learning frameworks. Section 2.2 describes the concepts related to workflow management. Section 2.3, in turn, describes existing state-of-the-art recommendation systems. Finally, Section 2.4 presents concepts related to learning to rank. Special attention is given to the learning-to-rank method LRAR, used in our experiments.

2.1 Machine Learning

In this work, we developed a framework for the design, execution, and recommendation of machine learning experiments. Machine learning is the study of computational methods that extract useful knowledge from experience to improve performance of a target application [70].

We can separate the types of machine learning methods based on the available input on the training step:

- **Supervised learning**: The training data has known target values and the machine learning method is required to make predictions on data with unknown target value. An example of supervised learning is the classification of a person as male or female according to his/her height and weight, taking into account the heights and weights of a heterogeneous group.

- **Unsupervised learning**: In this case, there is no training data. The method deduces the structure present in the testing data to make predictions for a new data. An example of unsupervised learning is clustering people according to their height.

- **Semi-supervised learning**: The semi-supervised learning makes use of data with unknown values for training. This method uses data without target values to deduce the structure present in the training data, and uses the data with target values for making predictions. An example of semi-supervised learning is the classification of Web images, as some images have tags describing their context/content, while others do not. In this case, the features of the images without tags provide information regarding the distribution of dataset images. This information can be later combined with image tags in the learning process.

In this work, our focus on machine learning experiments on classification, that is the problem on identifying to which categories observed in the training data a new observation belongs.

2.1.1 Typical machine learning experiment

A typical machine learning classification experiment is composed of six main steps, namely:

1. selection of a collection;
2. selection of an approach to splitting the collection into train and test sets;
3. selection of a feature descriptor;
4. optional selection of a normalization method;
5. selection of a classification method;
6. and selection of an evaluation measure to assess the effectiveness performance of the defined machine learning solution.

Figure 2.1 presents the typical six-step machine learning classification experiment. In the following, we present a high-level view of how each step works when an experiment is running. Once the input collection is defined, the method selected for splitting it into train and test sets is executed and a feature descriptor is then employed to extract a feature vector from each object within the collection. If a normalization method is selected, the feature vectors of all data in the train and test sets are normalized accordingly. After that, a classification method is applied and the results are analyzed considering the chosen evaluation measure.

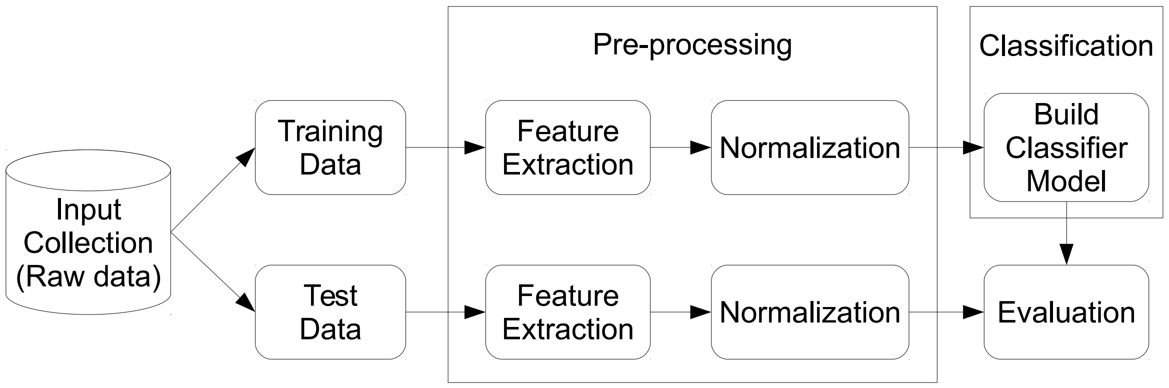


Figure 2.1: Organization of a typical machine learning experiment.

2.1.2 Machine Learning Frameworks

This section presents related work related to the proposal of frameworks that execute machine learning experiments.

PyML

PyML[1] is an interactive object oriented framework for machine learning written in Python. The framework has implemented the most used classifiers, as Support Vector Machines [131] and Nearest Neighbor [29], multi-class methods, and model selection. This framework allows combining classifiers and testing classifiers using typical evaluation process (cross-validation, ROC curves). However, it has some limitations, such as the lack of an interface (only command lines), the impossibility to add new algorithms to the tool, and the lack of a recommender system to support the experiment design process.

WEKA

The Waikato Environment for Knowledge Analysis (WEKA) [46] was created to supply the need of a unified workbench that allows easy access to state-of-the-art techniques in machine learning. It permits users to compare different machine learning methods on new data sets. The workbench includes algorithms for classification, regression, extraction of association rules, clustering, and attribute selection, as well as methods for preprocessing data and for statistical evaluation of the learning scheme. The WEKA framework is easily extensible thanks to its simple application programming interface (API), plugin-based scheme that automates the incorporation of new learning algorithm in the graphical

[1]`http://pyml.sourceforge.net/` — as of April 2014

interface. However, WEKA does not provide recommendation services.

Apache Mahout

The Apache Mahout framework[2] is a set of machine learning Java libraries systems used for classification, clustering, evaluation, pattern-mining, and building recommender. The advantage of the Mahout is that it was designed to be used for very large data sets, using the power of a Hadoop [118, 121] environment for distributed computing. Although it can implement a recommender, the Apache Mahout does not make recommendations for experiments in the framework, and it lacks an interface to design and visualize experiment specifications.

GraphLab

GraphLab [74, 75] is a parallel framework for machine learning that exploits the sparse structure and patterns of machine learning algorithms. It has a collection of applications for some tasks in large-scale graph computation, such as graph analytics, graphical models, computer vision, clustering, and collaborative filtering. Is it implemented in C++, being extensible for methods implemented in this language. The disadvantages of this framework are the absence of an interface and the lack of recommendation facilities.

Machine learning framework for Mathematica

The machine learning framework for Mathematica[3] (mlf) [85] is a collection of machine learning algorithms for intelligent data analysis, combining an optimized kernel with the manipulation, descriptive programming and graphical capabilities of Mathematica. As this framework is focused on intelligent data analysis, it has a limited number of machine learning algorithms. However, it is possible to combine them to solve problems. The disadvantages of the *mlf* is the lack of a way of extending the framework with other algorithms, the small number of machine learning algorithms, and the lack of a recommendation system.

Jubatus

Jubatus [52] is a distributed processing framework and streaming machine learning library. It has a client-server architecture, in which the client side has two commands: UPDATE that corresponds to the training phase of a machine learning algorithm, and ANALYZE,

[2]`http://mahout.apache.org` — as of April 2014.

[3]Mathematica is a registered trademark of Wolfram Research Inc. (`www.wolfram.com`) — as of July 2014.

that corresponds to the prediction phase of a machine learning algorithm; the server side consists of a feature vector preprocessing module and an algorithm module, which supports classification, regression, recommendation of data, simple statistics, and graph analysis. However, Jubatus framework lacks an interface for experiment design, although it supports the inclusion of new algorithms.

Encog

Encog [49] is a machine learning framework that supports a variety of algorithms, as well as data preprocessing. It has a GUI to help modeling and training the machine learning algorithms. Encog also supports multi-threads, scales well to multicore hardware, and can make use of GPUs. The disadvantage of this framework is the lack of a recommendation system to support the design of a more suitable machine learning experiment.

Accord.NET Framework

Accord.NET[4] is a framework that provides several scientific computing related methods, such as machine learning, statistics, and computer vision, to the .NET environment. The disadvantages of this framework is the lack of a recommendation system to improve experiments and the lack of mechanisms to include new algorithms to the framework.

KNIME

Konstanz Information Miner (KNIME) [10] is a platform based on the Eclipse platform, which enables easy visual assembly and interactive execution of data pipelines. It has a powerful intuitive interface, allows the integration of new modules, and supports the parallelization of the workload. However, this platform does not have a recommendation system to support the design of a more suitable machine learning experiment for specific applications.

Rattle

The R Analytical Tool To Learn Easily (Rattle) [123] is a R[5] package that provides a graphical user interface for data mining using R. Although it implement an interface for machine learning experiments, it is not intuitive. Also, the Rattle package does not implement parallelization, neither a mechanism to include new algorithms in the tool. The package also does not implement a recommendation system to support users in their experiment design process.

[4]`http://accord.googlecode.com` — as of April 2014.
[5]`http://www.r-project.org/` — as of July 2014.

Rapidminer

Rapidminer, formely known as YALE (Yet Another Learning Environment) [81] is an environment for machine learning and data mining process. It provides a rich variety of methods which allows rapid prototyping for new applications. Rapidminer uses XML documents to represent knowledge discovery process as operator trees. It contains more than 100 learning schemes for regression, classification, and clustering, and the environment is also easily to extend. However, Rapidminer does not implement a recommendation system to support users.

An overview of the related work is presented in Table 2.1. This overview describes each related framework according to different criteria: the number of machine learning algorithms, if the framework uses parallelization, if it is extensible, if it has an interface, and if it makes recommendations of experiments.

Framework	Machine Learning Algorithms	Parallel	Extensible	Interface	Recommendation
PyML[1]	A few	No	No	No	No
WEKA [46]	Many	Some Algorithms	Yes	Yes	No
Apache Mahout[2]	A few	Yes	No	No	No
GraphLab [74, 75]	Many	Yes	Yes	No	No
mfl [85]	A few	No	No	Yes	No
Jubatus [52]	Many	Yes	Yes	No	No
Encog [49]	Many	Yes	Yes	Yes	No
Accord.NET [4]	Many	No	No	No	No
KNIME [10]	Many	Yes	Yes	Yes	No
Rattle [123]	A few	No	No	Yes	No
Rapidminer [81]	Many	Yes	Yes	Yes	No

Table 2.1: Overview of existing machine learning frameworks.

2.2 Workflows

This section presents some concepts of workflow and some related work regarding workflow management systems.

2.2.1 Concepts

The proposed framework relies on the use of workflows to model machine learning experiments. This section describes the main terms related to workflows, such as workflow, activity, and workflow management system. These terms were defined by the Workflow Management Coalition (WfMC) [125] to standardize the terminology of workflows.

Activity

An activity is the description of a part of the work to be performed within the process. An activity can be atomic, representing an undivided action for the workflow, or may be composed by other activities. Activities are the basic elements of the construction of a workflow [125].

Process

A process is a set of one or more linked procedures or activities that fulfill a particular objective. We can split a process into business and scientific process, distinguishing them by its context: involving trades or conducting scientific experiments [125].

Workflow

A workflow is the automation of a process, in which documents, information or tasks are passed from one resource to another for action, according to a set of procedural rules [125]. Figure 2.1 shows the representations of a typical machine learning experiment workflow.

Workflow Management System

A Workflow Management System (WfMS) is an automatic system to design, manage, and monitoring the execution of a sequence of tasks, arranged as a workflow [125].

Scientific Workflow

A scientific workflow is the specification of a process that describes a scientific experiment [120]. There are some differences between scientific and business workflows, in which scientific workflows need to provide an easy-to-use environment for individual applications, simplify the process of sharing and reuse of process between scientists, and enable to track the provenance of the workflow execution and the workflow creation steps.

Ludäscher et al. [76] summarize some requirements of scientific workflows, such as:

- Seamless access to resources and services;
- Activity composition and workflow design, in which complex tasks are performed through the composition of simple activities;
- Scalability, as some workflows involve large volume of data or computational resources;
- Detached execution, as workflows with long running requires that it can be executed in the background or in a remote server, without being connected to a user's application;
- Reliability and fault tolerance, especially with Web services, as it can fail or become unacceptably slow;
- User-interaction, as many workflows require interaction of the user at some step of the execution;
- 'Smart' re-runs. Modifying an already executed workflow, the system does not need to execute the whole workflow again, only the parts that were modified;
- 'Smart' (semantic) links, as the system assisting the workflow design in which activity can be linked with other activities;
- Data provenance, in which scientific workflows should be reproducible and indicate specific data and tools used to perform the experiment.

2.2.2 Related Work on Workflow Management

This section presents related work on workflow management systems. Our objective is to illustrate typical solutions proposed in the literature for the management of workflows.

VisTrails

VisTrails [9] is an open-source tool that provides data and process management support for computational tasks. By combining features of workflows and visualization systems, it allows the combination of resources and libraries, and provides mechanisms to compare different results. The major difference of the VisTrails to other workflows systems is a comprehensive provenance infrastructure that maintains history and steps in the evolution of workflows and data in the course of exploratory tasks. Workflows are heavily used by scientists and engineers to generate and evaluate hypotheses, being constantly changed. VisTrails was designed to manage these workflows [20].

Here, we list the main features of the VisTrails tool:

- Flexible Provenance Architecture: VisTrails can track changes in the workflows and run-time informations about the execution;
- Querying and Re-using History: The tool provides an intuitive query interface to the exploration and reuse of the provenance information;
- Extensibility: Packages and libraries can be added dynamically in the system, without any change in the user-interface or recompilation of the system;
- Scalable Derivation of Data Products and Parameter Exploration: VisTrails supports the specification of a set of values for different parameters of the workflows [9].

Taverna

Taverna [86] is an open source workflow tool that enables scientists to manipulate existing bioinformatics applications in workflows. This project represents workflows in the Scufl language. Sculf is an XML-based, conceptual language in which each processing step of the workflow represents an atomic task.

The Taverna tool also contains a workbench that allows users to write workflows even without having to learn the Scufl language that acts as a container for a number of interface components which provide views and controllers involved in the composition of workflows. Workflows can be executed in the workbench using a local instance of the Freefluo enactment engine, a Java workflow orchestration tool for Web services.

Kepler

Kepler [5, 76] is a software system for designing, executing, reusing, and sharing scientific workflows. It is build upon Ptomely II [38], a mature, dataflow-oriented system, that has as main advantage a modeling and design paradigm called actor-oriented modeling. This paradigm calls workflow activities as *actors*, and the execution model of the workflow is specified by an object called *director*. The director defines how actors are executed and how it communicates with each other, improving the reusability of actor designs.

The Kepler system permits users to prototype the scientific workflow before implementing its actual code; it allows the use of computational resources on the net in a distributed workflow; it supports other language interfaces via Java Native Interface, permitting the reuse of existing analysis components and to target appropriate computational tools. This system also permits to run scientific workflows in batch mode using Ptomely's background execution feature.

This system has been used to design and execute scientific workflows in various fields, such as biology, ecology, astrophysics, and chemistry.

WOODSS

WOrkflOw-based spatial Decision Support System (WOODSS) [78, 105] is a computational tool implemented to be used in conjunction with Geographical Information System (GIS). This system is centered in monitoring the user activities in GIS and documenting them by means of scientific workflows.

WOODSS has three main objectives: document the user interactions with GIS in a scientific workflow representation, that: allows comparison between different methods, thus helping to understand standard GIS programming facilities; support the decision-making process, allowing reproducibility of steps, interactive updating, reuse as partial solutions, and validation against predefined criteria; and support progressive construction of the spatial analysis model and decision methods of decision support systems [78, 105].

WOODSS system started from environment studies and was upgraded to an agro-environmental planning, challenging a scientific workflow specification still combining preservation and exploitation issues.

SciPhylomics

SciPhylomics [34] is a data-intensive workflow proposed to produce phylogenomic trees based on a set of protein sequences to infer evolutionary relationships. This workflow was designed to benefit from parallel processing techniques, such as SciCumulus [35] cloud engine and Hadoop [118].

SciPhylomics provides for the users the management of the experiments during their execution, an analysis of local and distributed computation provenance, advantages of elasticity and adaptivity in managing virtual machines, and the benefit of cloud processing without assembling expensive infrastructure.

WASA

Workflow-based Architecture to support Scientific Applications (WASA) [79] is a system that integrates database knowledge with tools in a scientific environment, and that supports the management of scientific experiments. This system provides an environment that helps scientists to plan, organize, conduct, evaluate, document, and disseminate results of scientific experiments.

The WASA architecture supports not only the scientific data management, but also the development of scientific experiments, allowing users to specify new experiments and to reproduce and reuse past experiments. This system has the following key features: it is constructed according to distinct layers of abstraction and functionality, but it includes the core components on which a scientific environment relies. This architecture uses emerging standards for interoperability on datasets, enabling it to build top arbitrary types of databases, and its operation is centered on workflow and workflow management paradigms.

WASA has been applicable in a lot of scientific environment, like molecular biology, deciphering the genetic DNA information of organisms, and in an experiment of the impact of deforestation in hummingbird populations.

Workflow for the Alignment, Taxonomy, and Ecology of Ribosomal Sequences (WATERS) [48] is a workflow system that integrates software tools for sequence alignment, chimera removal, OTU determination, taxonomy assignment, phylogenetic tree construction, ecological analysis, and visualization tools. WATERS uses the Kepler [5, 76] system as a platform for its collection-oriented approach.

The motivation for building WATERS was to minimize technical challenges when performing DNA sequence clustering, pylogenetic tree, and statistical analysis by automating the workflow, minimize technical analysis, making it more available and allowing users with little skills to still use the software, and standardize the analysis methods of ribosomal sequences, facilitating comparability and reproducibility of results.

Armadillo

Armadillo [73] is a workflow platform built to design and conduct phylogenetic studies. It includes a number of phylogenetic and general bioinformatics tools, and also permits the extension of new bioinformatics tools. A typical bioinformatics task is described as a workflow with the following steps: data acquisition, data analysis, and report and result generation.

This platform represents datasets and bioinformatics applications as components that can be linked together to create a dataflow. The configuration of each application is made in a dialog box, facilitating the use of the most used features. It also integrates a sequence viewer and provides access to phylogenetic tree interference and manipulation applications.

Towards Adapting Scientific Workflows System to Healthcare Planning

Vilar et al. [119] proposed a context-driven approach to produce workflows in healthcare activities. Healthcare is a dynamic scenario, in which professionals constantly interact with a tool, registering patient information, intervention plans, and desired outcomes, creating the need for a workflow management system that supports different kinds of activities, as well as allows annotation of tasks, provenance analysis, and changes according to context variables. Their work identified three aspects in healthcare workflow: each tasks may have subworkflows that are conducted according to a context; tasks can be interrupted or changed; and tasks must provide a record of performed changes.

2.3 Recommendation Systems

The difference between the available set of data and user's interest subset of data is enormous, and it grows daily. However, the identification of the user's subset is difficult, and therefore, tools that help this identification, like search methods, metadata and recommendations, are very important. Recommendation is defined by Gonçalves [45] as: *given a collection and an actor, and a set of ratings for objects in that collection produced by others or the same actor, recommends (produces a subset of that collection) for that particular actor.* This kind of service is of great value when the actor has a little knowledge about the subject, or either for experts actors, especially for a rapidly growing database.

Recommendation systems became an important research area since middle 1990 [53, 96, 106] and continues to be studied because it is a problem-rich research area with several practical applications, such as recommendation of books [56], musics [27], CDs [71], movies [82], friends [72], and news [11]. The recommendation problem is usually formulated as a problem of estimating ratings for objects that have not been rated by an actor. There are two possible ways of rating unknowns objects: using *traditional heuristics* based on information retrieval methods or *model-based* that uses a model learned from the underlying data using statistical learning and machine learning techniques.

The literature in recommender systems classifies them into different categories based on how the recommendations are made [8, 23, 56, 115]: Content-based filtering [12, 36, 126, 128, 132], Collaborative-Filtering [16, 47, 51, 89, 130], and hybrid approaches [19, 101, 107].

2.3.1 Content-based filtering

The Content-based filtering is a recommendation technique based on the similarity between the content of the objects [25]. Its idea is that an actor may have interest in similar objects to the item already rated. Therefore, the system identifies and recommends the objects similar to the previously rated by the actor.

The advantages of this type of recommendation system are: it does not need information of the user, only his/her history behavior in the system; it is adaptive, making better recommendations over time; and it does not need previously knowledge about the domain to work [94]. However, recommendation systems face the new-user problem, where the lack of information about the actor prevents any recommendation based on user preferences.

2.3.2 Collaborative-filtering

The collaborative-filtering is a recommendation technique based on the assumption that actors who had similar object interest in the past, will probably have similar interest in other objects [96]. Therefore, for an user, the system identifies other users with similar interest and recommends their well-rated objects.

The collaborative-filtering has the same advantages of the content-based filtering (it does not need data from the user, it is adaptive, and does not need previously knowledge of the domain) and an exclusive advantage, which refers to the possibility of recommending objects that do not have similarities with the objects rated by the actor, thereby identifying cross-genres niches [94].

The disadvantages of the collaborative-filtering are: the new-user problem; new-item problem, according to which, when a new object is included, the system does not have any information about the object, which hampers novel recommendations until other actors rate it; the black-sheep problem, according to which when users present unusual preferences, the system will have difficult to find other users with similar preferences, leading to bad recommendations; and sparsity, which refers to the fact when the number of items already rated are very small compared to the number of ratings that need to be predicted, e.g., in a movie recommendation system there may be some movies rated by only few people, and these movies would be rarely recommended, even if the movies had a good rating [1, 23].

Table 2.2 summarizes the advantages and disadvantages of content-based and collaborative-filtering.

2.3.3 Hybrid approaches

Hybrid approaches consist in the combination of content-based and collaborative-filtering techniques to combine its advantages and minimize its disadvantages. Burke [19] presents a taxonomy to those hybrid methods:

- *Weighted*: Several recommendation techniques produce scores to be combined into a single recommendation;

Technique	Advantages	Disadvantages
Content-based filtering	• Does not need data from user • Adaptive • Domain knowledge not necessary	• New-user problem
Collaborative-filtering	• Does not need data from user • Adaptive • Domain knowledge not necessary • Cross-genre niches	• New-user problem • New-item problem • Black-sheep problem • Sparsity

Table 2.2: Comparison of recommendation techniques

- *Switching*: The system switches between recommendation techniques according to the current situation;
- *Mixed*: The result recommendation of several techniques are mixed in the final recommendation;
- *Feature combination*: Features from different recommendation data sources are used in a single recommendation technique;
- *Cascade*: One recommendation technique refines the recommendation output of another;
- *Feature augmentation*: The output of a technique is used as input of another;
- *Meta-level*: the model learned by a recommendation technique is used as input of another.

2.3.4 Related work on Recommendation Systems

In this subsection, we introduce research related to the construction of recommendation systems.

Huang et al. [56] described a two-layer graph model on book recommendation using a hybrid approach. In the first stage of the computation, the users and books in the digital library are represented as feature vector. The user feature vector contains the user demographic data and the book feature vector contains the book attributes and content information. The recommendation model computes the similarities between the users and the similarities between the books. At the second stage, a two-layer graph is constructed with a book layer and an user layer. Inter-layer links are based on the purchase histories of all users. In this system, the recommendation is made using graph searches, traversing the graph to find books with strong association with the user.

Miller et al. [83] presented the PocketLens algorithm that solves two key problems in recommendation system: portability and trust. This algorithm performs in a peer-to-peer environment, wherein, when the user is online, it creates a model that can be used to make recommendations offline. The trust issue is resolved by the personal recommender, which shares only the information explicitly identified by the user.

Lacerda et al. [69] proposed a framework for associating advertisements (ads) with web pages. It implements a content-based filtering, using the structural parts of an ad: title, textual description, and hyperlink, a genetic programing to learn a model to recommend the most appropriate ads, given the context of the Web page.

Lo and Lin [72] proposed a new graph-based algorithm named weighted minimum-message ratio (WMR), which generates a personalized friend list by message interaction among web members. The link in the graph was defined as the minimum number of interactions between the users, and the WMR algorithm considers that, the longer the path, the lower the recommendation. The recommendation rating is calculated as:

$$R_{o,j} = \sum_k [P_k(j).CSum \times \prod_i C(S_{i-1}, S_i)]$$

where $R_{o,j}$ is the rating for the node j from the node o; $P_k(j).CSum$ is the sum of the path from the node o to j; $C(S_{i-1}, S_i)$ is the proportion of the message from the path and the total messages of the level i.

Bollen et al. [14] presented a system for recommendation on videos based on the content-based filtering that comprises three stages. First, the system harvests download logs and metadata (video title, abstract, description, type, usually generated by users) from all videos. Next, a video relationship matrix derived from video downloads is generated. Finally, a Spreading Activation search in the relationship matrix that encodes paths connecting related videos is performed to generate the recommendations.

Koenigstein et al. [65] presented a work of recommendation on the Yahoo! Music dataset using collaborative filtering. This dataset is characterized by three features: multi-taped rated items, four level taxonomy (dealing with the sparsity), and timestamps associated with the ratings. The system employs a model based on matrix factorization to map items and users into comparable latent factors, being the predicted rating by an user u to item i the following equation:

$$\widehat{r}_{ui} = \mu + b_i + b_u + p_u^T q_i$$

where μ is the average rating, b_u and b_i are the user and item biases respectively, and $p_u^T q_i$ the affinity of user u to item i.

Tayebi et al. [114] presented a novel approach (CrimeWalker) to crime suspect recommendation based on partial knowledge of offenders involved in a crime incident and a known co-offending (offender who have committed crimes together) network. The proposed model extends a existing random walk based model, TrustWalker [59], to address link prediction combined with the ability to perform recommendations based on a set of offenders given as input. CrimeWalker performs several walks on the co-offending network starting with an already charged offender, to recommend offender suspicious.

Kaster et al. [63] presented the use of Case-Based Reasoning (CBR) [97] as a retrieval mechanism in the WOODSS [78, 105] (Section 2.2.2) to help users choose the most adequate models from those available in the database. CBR is a reasoning model which consists in solving new problems by adapting solutions that were already used to solve previous problems [97]. The similarity retrieval applied with the CBR approach uses the metadata associated with each WOODSS workflow, which contains the problem focused by the workflow and its meaning. The process of similarity analysis employed by the CBR system is described as the following steps: (i) Find correspondences, aligning the input problem with the stored workflows; (ii) Compute the degree of similarity of corresponding features; and (iii) Assign importance values to features. WOODSS' CBR mechanism uses city-blocks metrics to calculate the similarity evaluation between the input and the stored workflow.

For a further deep investigation on existing solutions for recommendation services, please refer to the following surveys: Almazro et al. [4] and Bobadilla et al. [13].

Different from the above approaches, we proposed in this dissertation the use of a recommendation based in the search of similar workflows that were previously executed in the framework.

2.4 Learning to Rank: LRAR

For making recommendations in the framework, we propose the use of a learning-to-rank method based in association rules. We describe the concept of learning to rank and the learning-to-rank approach used in this work in the next sections.

2.4.1 Learning to Rank

Ranking models and functions is an important research topic in many fields. In the literature several empirical ranking methods are proposed, such as boolean, vector space, and probabilistic models [7]. However, it is difficult to empirically tune the parameters of ranking functions of the above methods, therefore, recently, learning-to-rank approaches have been proposed. These methods exploit machine learning methods to automatically learn effective ranking functions.

The task of learning to rank is defined as follow. A training data D consisting of a set $< q, d, r >$, where q is a query, d is a document, represented as a list of features $(f_1, f_2, ...f_n)$, and r is the relevance of d to q, with discrete values, is used to create a model to relate the features of the document to the corresponding relevance. The test set T consists of a set $< q, d, ? >$, where the relevance of the document d for the query q is unknown. The model learned is used to produce a likelihood of relevance of such documents to the corresponding queries, which are used to generate the final ranking.

2.4.2 Learning to Rank using Association Rules

The learning-to-rank methods in the literature rely on techniques such as support vector machines [50,131], neural networks [18], and genetic programming [39]. Veloso et al. [117] proposed an alternative method using associative rules [2], that generates a model R, composed of rules of the form $f_i \cap ... \cap f_j \rightarrow r$, describing the training data by feature-relevance associations. Once the model is built, the rules are used to estimate the relevance of documents in the test set. There are two measures used to quantify the quality of a rule: the confidence θ (conditional probability of relevance r given $f_i \cap ... \cap f_j$) and the support σ (fraction of training examples containing features $f_i \cap ... \cap f_j$ and relevance r).

	Query	Documents				Relevance
		id	Feature 1	Feature 2	Feature 3	
Training Data	Query 1	1	[0.85-0.92]	[0.36-0.55]	[0.23-0.27]	1
		2	[0.74-0.84]	[0.36-0.55]	[0.46-0.61]	1
		3	[0.51-0.64]	[0.56-0.70]	[0.23-0.27]	0
	Query 2	4	[0.74-0.84]	[0.36-0.55]	[0.28-0.45]	0
		5	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	1
		6	[0.93-1.00]	[0.36-0.55]	[0.62-0.76]	0
	Query 3	7	[0.74-0.84]	[0.22-0.35]	[0.12-0.22]	0
		8	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	0
		9	[0.85-0.92]	[0.71-0.80]	[0.46-0.61]	1
Test Data	Query Test	10	[0.85-0.92]	[0.56-0.70]	[0.46-0.61]	1
		11	[0.51-0.64]	[0.36-0.55]	[0.28-0.45]	0
		12	[0.34-0.50]	[0.22-0.35]	[0.46-0.61]	1

Table 2.3: Queries, Documents and Relevance, extracted from [117]

Consider the training and test sets shown in Table 2.3, with three queries for the training data and one query for test data, and each query having three documents associated, represented by three features. We can generate the following rules:

- Feature 1 = [0.85-0.92] $\rightarrow r = 1$ $(\theta = 1.00, \sigma = 0.22)$
- Feature 1 = [0.74-0.84] $\rightarrow r = 0$ $(\theta = 0.67, \sigma = 0.22)$
- Feature 2 = [0.36-0.55] $\rightarrow r = 1$ $(\theta = 0.50, \sigma = 0.22)$
- Feature 3 = [0.28-0.45] $\rightarrow r = 0$ $(\theta = 1.00, \sigma = 0.11)$
- Feature 1 = [0.74-0.84] $\cap$ Feature 2 [0.36-0.55] $\rightarrow r = 1$ $(\theta = 0.50, \sigma = 0.11)$

However, generating every rule for the training data is costly. The method of Veloso et al. [117] generates the rules on demand-driven basis, making it fast. Using the id 10 as example, only a few documents will be used to generate the rules, as seen in Table 2.4.

Documents				Relevance
id	Feature 1	Feature 2	Feature 3	
1	[0.85-0.92]	—	—	1
2	—	—	[0.46-0.61]	1
3	—	[0.56-0.70]	—	0
4	—	—	—	0
5	—	[0.56-0.70]	[0.46-0.61]	1
6	—	—	—	0
7	—	—	—	0
8	—	[0.56-0.70]	[0.46-0.61]	0
9	[0.85-0.92]	—	[0.46-0.61]	1
10	[0.85-0.92]	[0.56-0.70]	[0.46-0.61]	1

Table 2.4: Learning to Rank using Association Rules for the document id 10.

The generated rules, ordered by its confidence, are:

- Feature 1 [0.85-0.92] $\rightarrow r = 1$ $(\theta = 1.00, \sigma = 0.22)$
- Feature 1 [0.85-0.92] $\cap$ Feature 3 [0.46-0.61] $\rightarrow r = 1$ $(\theta = 1.00, \sigma = 0.11)$
- Feature 3 [0.46-0.61] $\rightarrow r = 1$ $(\theta = 0.75, \sigma = 0.33)$
- Feature 2 [0.56-0.70] $\rightarrow r = 0$ $(\theta = 0.67, \sigma = 0.22)$

- Feature 2 [0.56-0.70] $\cap$ Feature 3 [0.46-0.61] $\rightarrow r = 1$ $(\theta = 0.50, \sigma = 0.11)$
- Feature 2 [0.56-0.70] $\cap$ Feature 3 [0.46-0.61] $\rightarrow r = 0$ $(\theta = 0.50, \sigma = 0.11)$
- Feature 2 [0.56-0.70] $\rightarrow r = 1$ $(\theta = 0.33, \sigma = 0.11)$
- Feature 3 [0.46-0.61] $\rightarrow r = 0$ $(\theta = 0.25, \sigma = 0.11)$

These rules are combined to estimate the relevance of document id 10, and the score of each rule is weighted according to its confidence by the Equation 2.1, where R_d is the total number of rules generated and $\theta(r_i)$ are the rules confidence with relevance r_i.

$$s(r_i) = \frac{\sum_{r_i \in R_d} \theta(r_i)}{|R_d|} \tag{2.1}$$

Therefore, the rank of a document is estimated by the linear combination of the normalized score for each relevance, as shown in Equation 2.2.

$$rank = \sum_{i=0}^{k} r_i \frac{s(r_i)}{\sum_{j=0}^{k} s(r_j)} \tag{2.2}$$

With this, we have that the rank of document id 10 is: $s(0) = 0.1775$ and $s(1) = 0.4475$, and $rank = 0.716$.

2.5 Final Considerations

In this work, we approach three different topics to elaborate our framework: Machine Learning, Workflow Management, and Recommendation System. In machine learning, we presented 11 related work and its features. We concluded that some are not extensible to novel features (e.g., [85, 123]), some does not have an interface (e.g., [52, 74, 75]), and none of them has a functionality for recommending previous experiments made in the system.

We also present eight related work on Workflow Management, being applied in a number of scenarios, such as ecology (e.g., [5,76,78,105]) and biology (e.g., [34,73,79,86]). Each one of these systems has its own advantages, however, a few of them are general enough to be extended to a machine learning application, like VisTrails [9], Kepler [5,76], and WASA [79], and just one of them (WOODSS [78,105]) has a system for recommending previous experiments.

As none of the studied machine learning frameworks and workflow management systems have a functionality for recommending experiments made in the system, we also studied recommendation systems in the literature to guide the implementation of the functionality in our machine learning framework. Some of the recommendation systems are based on graph models between its itens, e.g., Huang et al. [56] and Lo and Lin [72], and others are based on content, e.g., Lacerda et al. [69] and Bollen et al. [14]. Different from the approaches, we proposed the use of similarities measures to calculate the similarities between similar workflows that were previously executed in the framework.

Chapter 3

Machine Learning Framework

In this chapter we introduce the proposed framework for designing, deploying, executing, and recommending machine learning experiments. The framework provides a workflow-based standardized environment, making it easy to evaluate different feature descriptors, classifiers, and fusion approaches. In the next sections, we describe how the proposed machine learning framework is structured, and present how the framework was implemented, describing its architecture and its modules, as well as the employed plugin scheme.

3.1 Modeling a Machine Learning Experiment as a Workflow

In our framework, we model each machine learning experiment step (e.g., feature extraction, normalization, and classification) as a workflow activity, allowing the user to construct a machine learning experiment as a workflow.

The proposed workflow representation in the framework is modeled as a directed acyclic graph (DAG), where its activities are depth-first traversed, usually beginning in a collection and ending at an evaluation measure. Figure 3.1 shows a representation workflow of a typical machine learning experiment, with all the typical six steps.

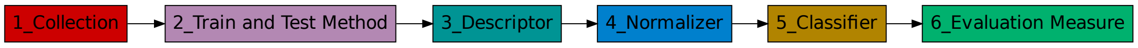


Figure 3.1: Workflow representation of a typical machine learning experiment.

3.2 Machine Learning Framework

The objective of the proposed machine learning framework is to facilitate the automation of classification experiments. It is responsible for managing the steps of machine learning experiments. Each step consists of a module in the framework (e.g., collection, train and test set definition, feature extraction, normalization, classification, fusion, and evaluation), that runs independently of others. Efficiency aspects on the workflows execution are addressed by exploiting multiple cores in the extraction and normalization modules.

The architecture of the proposed framework is shown in Figure 3.2. It consists of three layers: an interface, the core of the framework, and a set of repositories. The interface is responsible for the communication with the user. Using the interface modules, users can design, call the execution of a workflow, and receive the results of the execution. In the design of a workflow, the core of the framework is in charging of connecting to the methods of each module in the repositories to build the machine learning experiment, and call them again for the execution of the experiment, being responsible for the communication between the executing modules. Once the execution is done, the workflow and its results are stored in a repository, which can be used later for making recommendations. The recommendation service can help users in their task of building a workflow experiment, by avoiding common errors and providing best practices from the past. The service will look for past experiments that are similar to the built one, intending to provide the user with alternative solutions that can lead to better results. The use of a recommendation service in this context also allows users to reuse experiments and activities successfully used previously. For this recommendation, we used similarity functions to estimate the similarity of two graph-based representation of machine learning experiments.

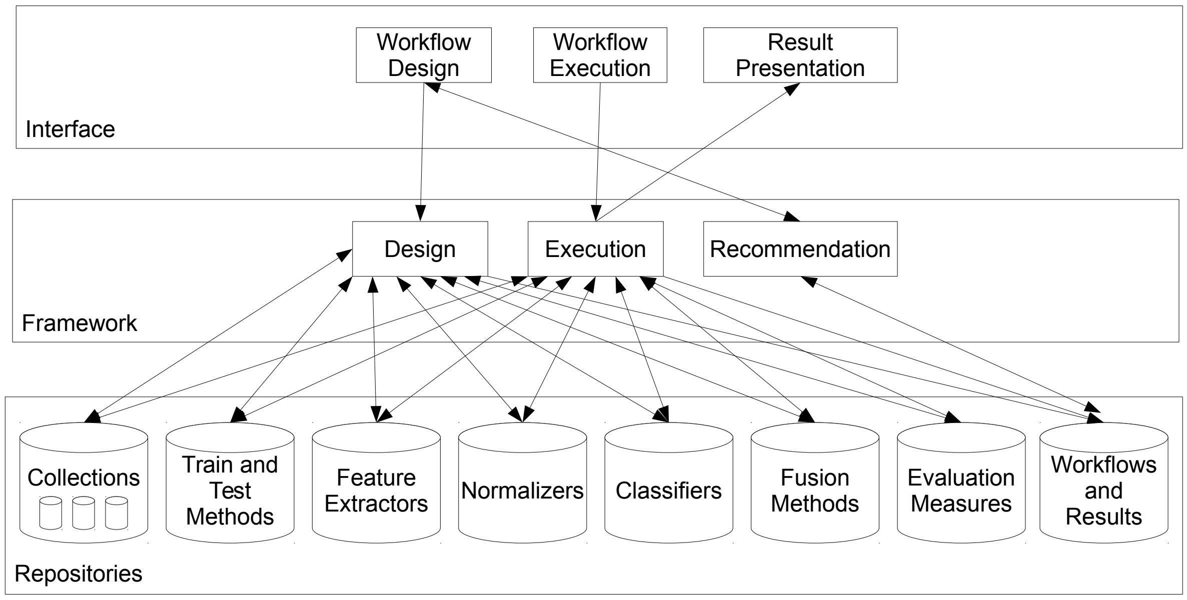


Figure 3.2: Architecture of the framework to automate machine learning experiments.

Each one of these modules is responsible for a step in a machine learning experiment. The *Collection Module* is responsible for gathering the objects of the collection for the framework. The *Train-and-Test Module* splits the objects of the collection into two sets, a training set, which is used to train a model in the classifier and a testing set, used to test the built model. The *Feature Extraction Module* extracts a feature vector from the collection objects according to a selected method. The normalization of the feature vectors of all data is responsibility of the *Normalization Module*, and the *Classification Module* performs the classification of the test set using the model learned with the train set. The results of the classification are evaluated in the *Evaluation Module.*

The implementation of a new framework for experiments in machine learning, instead of using a generic workflow, has some benefits and drawbacks. The disadvantages of the implementation of a new framework are:

- Workflow community: The implementation of a new framework is disadvantageous, because it lacks interaction with the workflow community, which could help improving the framework;
- New modules: The insertion of new modules in the machine learning framework may lead to problems concerning the connections among old and novel modules.

However, implementing a new framework has benefits, such as:

- Predefined matching: With a specific and predefined matching between modules, users are not able to introduce errors in the workflow during the design of an experiment;

- Productivity: Having defined modules, a specialist is more productive on designing and deploying experiments; and

- Recommendation system: The insertion and implementation of recommendation tools is easier in the new framework.

3.3 Implementation Aspects

In this section, we present the main decisions for the implementation of the machine learning framework, such as the language that the framework was implemented, the use of XML-based documents for storing configuration parameters of module plugins and for representing a workflow experiment, and the definitions of the connections between the modules.

3.3.1 Plugin scheme

In order to provide a extensibility functionality in the build framework, we decided to use a plugin scheme. Plugins consists of components that implement or encapsulate new features to an existing software, developed according to defined standards and interfaces [94].

Each module of the machine learning framework is composed of plugins of different implemented methods. This plugin scheme makes the framework more flexible and easily extensible, i.e. it is possible to define methods in any programming language, and add those methods to the framework using only a Python wrapper. The plugins are organized according to the step of a machine learning experiment (module).

This framework has already implemented a wide variety of plugins. Table 3.3 summarizes the plugins already available in the framework.

Modules	Number of plugins	List of plugins
Train and Test	5	K-fold [66], Leave Video Out [66], Number of Images, Percentage of Images, Read Files
Extraction	18	ACC [55], Bag of Visual Words, BIC [110], CCOM [67], CCV [92], CEDD [26], GCH [112], Gist [88], HOG [31], HTD [77, 127], JAC [122], LAS [113], LBP [87], M-SPyd [84], QCCH [54], SASI [21, 22], Stats[a], Unser [116]
Normalization	4	Min-Max [3], Term Frequency [100], TF-IDF [100], Z-Score [3, 58]
Classification	21	DecisionTree [17], kNN [6], LDA [42, 44], libSVM [24], LogisticRegression [15], MCOCSVM[b], MCOSOPF1[b], MCOSOPF2[b], MCSVM1VS[b], MCSVMDBC[b], MCSVMexternal[b], MCSVMSH[b], OCSVM[b], OPF [91], OSOPF1[b], OSOPF2[b], OSOPF_mc[b], SVM [24], SVM1VS[b], SVMDBC [32, 33], SVMSH[b]
Fusion	3	Concatenation [99], Majority Voting [68], Probability Fusion[c]
Evaluation Measures	10	Confusion Matrix [111], False Negative, False Positive, F-measure, Global Accuracy Score, Cohen's Kappa [28], Normalized Accuracy Score, ROC curve [41, 109], True Negative, True Positive

Figure 3.3: List of the plugins implemented in the framework.

[a]This is a descriptor recently proposed by the research group. Its description has not been published yet.

[b]This is a classifier recently proposed by the research group. Its description has not been published yet.

[c]This is a fusion method recently proposed by the research group. Its description has not been published yet.

3.3.2 XML Documents

For organizing experiments, the use of files in the eXtensible Markup Language (XML)[1] was implemented. The XML documents store the values of parameters of the methods implemented in each plugin and the setup of a machine learning experiment. The XML file

[1]`http://www.w3.org/XML/` — as of May 2014.

format was chosen due to its flexibility, portability, and ease management by computers and, in some cases, by humans.

For controlling either the plugin methods or the workflow itself, the framework uses the eXtensible Markup Language for defining specific tags for each module. For the Collection Module, each collection is stored and linked to the framework as a XML file, containing three tags: a “collection” root tag with the identification of the collection, the number of classes and the number of objects in the collection; a “class” tag with the identification of the class and the number of objects in the class; and a “object” tag with the path to the object of the class being considered. Figure 3.4 shows the XML Schema that describes the structure of the XML of a collection.

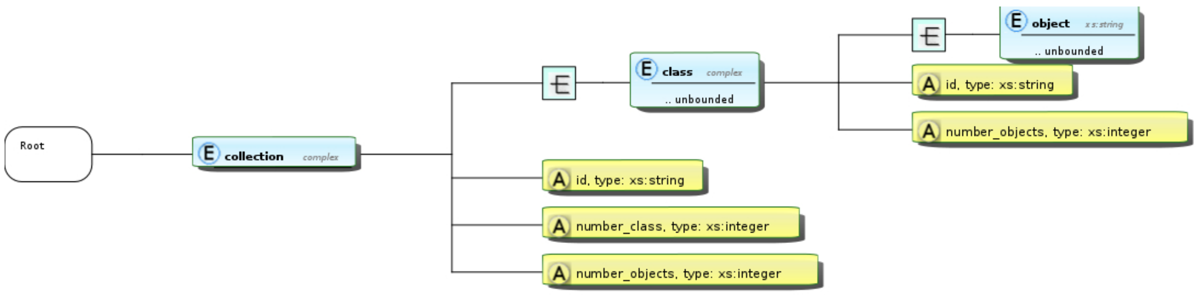


Figure 3.4: XML Schema of a collection XML document.

For the other modules of the framework, XML documents are used to describe the plugins of the module. Those XML documents have a “software” root tag with the name of the plugin and a list of “abletolink” tags with the module from which the plugin can receive a link. In the case that the software in the plugin needs parameters, they are described in “parameter” tags. The XML Schema of this plugin XML is presented in Figure 3.5.

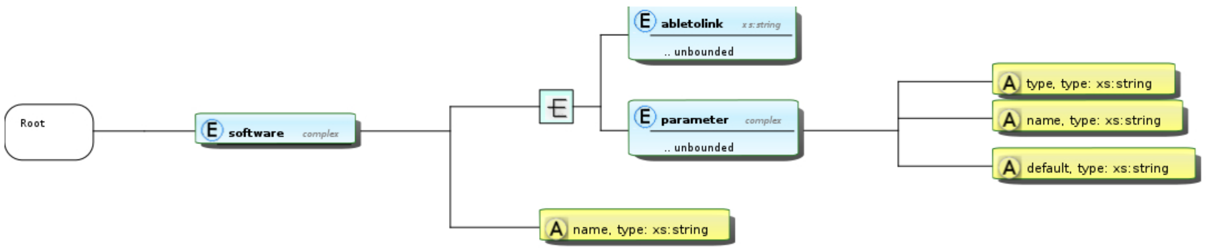


Figure 3.5: XML Schema of a plugin XML.

The last use of an XML document in the framework is in the representation of a workflow experiment, which describes each module and each method in the built workflow.

The workflow is represented as a graph, with a list of modules and a list of links between the modules (nodes and edges, respectively). A root tag "experiment" contains a name for the experiment, the name of the author, the number of iterations in the experiment and a date and hour control. Child tags of the root are: the modules present in the workflow, with an identification for the module, the plugin selected and its parameters; and a "link" tag representing the input and output links for each module id. Figure 3.6 shows an example of an experiment XML file.

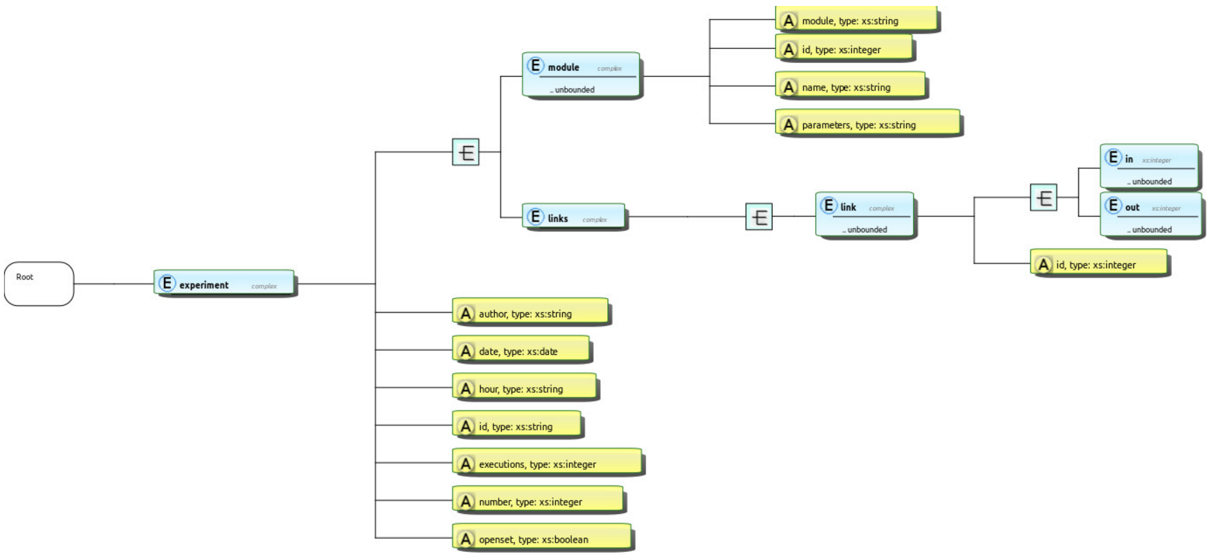


Figure 3.6: XML Schema of an experiment XML document.

With the XML document of the machine learning experiment, the framework traverses the workflow as in a deep-first traversing, beginning in the Collection Module and following the output links present in the XML file until there is no more modules to be visited. This method favors the execution of a whole branch in the machine learning experiment, saving the result of the branch, and avoid possible conflicts between the results of two different branches on a parallel execution of modules.

Each module of the framework has a default construction for the plugin wrapper, based on which functions of the plugin and their parameters are defined. This construction is explained in the next sections.

3.3.3 Module Implementation

This section describes the implementation of each module in the framework, providing pseudo-algorithms for the plugins of each module. These algorithms are presented in Appendix B.

The framework was implemented in Python Programming Language[2], due to its easy usability, efficient high-level data structures, its wide range of supported libraries and its easily interface with other languages, and its simple but effective approach to object-oriented programming. The basic machine learning methods in the framework were implemented using the Scikit-learn [93] tool for machine learning in Python. This tool has all the well-known algorithms implemented, and it is fast and has a better memory footprint [30] than other machine learning tools (e.g., WEKA [46]).

Collection Module

The Collection Module is responsible for reading the XML document that contains the path to the objects of the collection and associate each objects with each class of the collection. A new collection can be added to the framework simply including its XML document to the collection module in the framework folder.

Train-and-Test Module

This module is responsible for splitting the collection selected in the workflow into two sets: a training set, used to train the classification method and create a classifier model; and a testing set, used to test the built classifier model to evaluate the proposed workflow. This module is selected before the feature extraction because some extraction features methods need a train and test split to perform the calculation, e.g., the Bag of Visual Words [110]. The Train and Test Module has two functions: *split_train_test*, that calls the selected plugin for splitting the collection into train and test (Line 2 of Algorithm 1); and *write_tex* for writing the parameters of the module in the result TeX file (Line 2 of Algorithm 2).

Plugins in this module have at least two functions: *train_test* (Algorithm 3), to create the train and test sets, and *write_tex* (Algorithm 4), to write the parameters of the train and test plugin in the file with the results of the experiment.

Feature Extraction Module

The Feature Extraction Module extracts features from the objects in the collection that describe them according to the descriptor method selected. The module has only one function (*extract_features* presented in Algorithm 5), which is responsible for checking if the collection has its features already extracted (Line 1) and calling the plugin for feature extraction for each object without the feature extracted in the collection (Line 5).

[2]`https://www.python.org/` — as of May 2014.

Each plugin of a descriptor method has two main functions: *extract* and *fv_transform*. Algorithm 6 performs the extraction of the feature vector for an object, according to the method selected. Algorithm 7 transforms the feature vector returned by the method in the defined standard of the framework. To avoid unnecessary reruns, the Feature Extraction Module always saves the result of the extraction, to reuse it in subsequent executions.

Normalization Module

The Normalization Module is responsible for encoding every feature vector into the same feature space, to avoid problems with distances, and facilitating the combination between the feature vectors in the classification step. This module has one function *normalize_features*, presented in Algorithm 8, that calls the selected plugin to normalize the feature vector of the collection (Lines 3 and 5).

Plugins of this module has one main function *normalize*, shown in Algorithm 9, which uses training set parameters of the normalization in Line 2, and normalize the feature of the object according to the parameters in Line 4.

Classification Module

A classification consists in allocating new objects in previously defined classes, so that objects belonging to the same class share the same properties. This module performs the classification of the feature vectors in the testing set, using a model trained with the objects in the training set. The *classification* function of the Classification Module, responsible for performing the classification of the testing set according to a model learned with the training set, is shown in Algorithm 10.

The main function of plugins for the Classification Module, *classify*, is shown in Algorithm 11. A classification model is learned with the training set in Line 1, and used to predict the objects in the testing set (Line 3).

Fusion Module

For different problems, the ability to combine the results of multiples algorithms provides a significant improvement in overall performance [103]. There are two levels of fusion, early fusion (sensors and features) and late fusion (rank and classifiers). This Fusion Module is a complex module that performs fusion both for feature vectors and for classification results. The XML document of the plugin defines which modules are possible to be merged by the plugin. As fusion, this module is the only one that accepts more than one input link. *fusion* (Algorithm 12) is the function of the Fusion Module responsible for performing the fusion.

The main function of a plugin for this module is described in Algorithm 13, which in Line 1 performes the fusion of the results of linked modules.

Evaluation Module

This module shows how the built workflow performs in the classification experiment, according to the methods selected (e.g., accuracy score, confusion matrix, etc.). The Evaluation Module is responsible for creating a PDF file at the end of the execution to show the result of the workflow experiment, helping the interpretation of the results. For creating the PDF with the results of the experiment, it was implemented the use of the $\TeX$ [64] typesetting system. The $\TeX$ was selected because it produces a high-level result using a minimum effort, provides the same result in any machine, and is robust with the use of complex mathematical formulae. The wrapper of the plugins of this module consists of three function: *evaluation* (Algorithm 15) that performs the evaluation of the classification result according to the method implemented; *string_file* (Algorithm 16) to write its results at the end of each execution; and *write_tex* (Algorithm 17) to write in a TeX file the results to be shown to the user.

Recommendation Module

The Recommendation Module works apart from the other modules. This module receives as input the built experiment workflow in the framework and a recommendation plugin. It uses the selected plugin to rank previous experiment workflows executed in the framework and presents the top-ranked ones recommended by the plugin. The wrapper of the method contains the main function *distance*, as shown in Algorithm 19, that calculates the distance between a sequence in the previous experiments and the sequence currently built in the framework. With the distance to every previous experiments, the module ranks this values and shows to the user the top-ranked ones.

Chapter 4

Validation

In this chapter, we present a case study (Section 4.1), in which we create a workflow experiment. Our objective is to show the use of the framework in a real-world scenario concerning the evaluation of machine learning algorithms. Section 4.2 describes how the framework can be extended to be used with other workflow management systems, e.g., the VisTrails system [9]. Finally, Section 4.3 presents an overview about the recommendation module and the conducted experiments related to its validation.

4.1 Case study

In this section, we present a case study concerning the use of the framework in the design and the execution of a machine learning experiment.

4.1.1 Application

Suppose that we want to compare the results of a machine learning experiment using two different methods of feature extraction. For exemplifying this experiment, we selected a fruit and vegetable identification problem, a recurrent task in supermarkets. This problem is defined as: given a product, identify its species (e.g., apple, potatoes, oranges) and its variety (e.g., Gala and Fuji apples) to define its price [40, 98].

For this experiment, we selected a representative collection of fruits [98], with 15 classes and 2,633 images. The K-Fold plugin was selected to split this collection into train and test sets. The plugin splits the collection into 3 folds, in which each fold is used as test, and the other two as the training set. For the Feature Extraction Module, we select two descriptor plugins, Border/Interior Pixel Classification (BIC) [110] and Local Activity Spectrum (LAS) [113]. These descriptors extract the feature vectors based on color and texture visual properties, respectively. With those two methods, we can compare the results with the objective of determining which descriptor performs better in this collection. For fair comparison, the steps following the Feature Extraction Module has to be identical. The selected plugin for the Normalization Module was the Min-Max [3] method, with parameters min = 0.0 and max = 1.0, For the Classification Module, the libSVM plugin was selected, based on the libSVM 3.17 [24] implementation, with linear kernel with degree 3, and a grid-search for the C and γ parameters. To present the results of the classification, the Global Accuracy Score and Confusion Matrix evaluation measures were selected. The Global Accuracy Score shows the mean accuracy of the classification for each fold, and the Confusion Matrix plugin plot the mean confusion matrix of the folds, showing the percentage of images classified as each class. The PDF with the result of this experiment is shown in Appendix A.

4.1.2 Use of the framework

Upon the framework start-up, the user has access to an interface with a large area for designing the machine learning experiment, as shown in Figure 4.1.

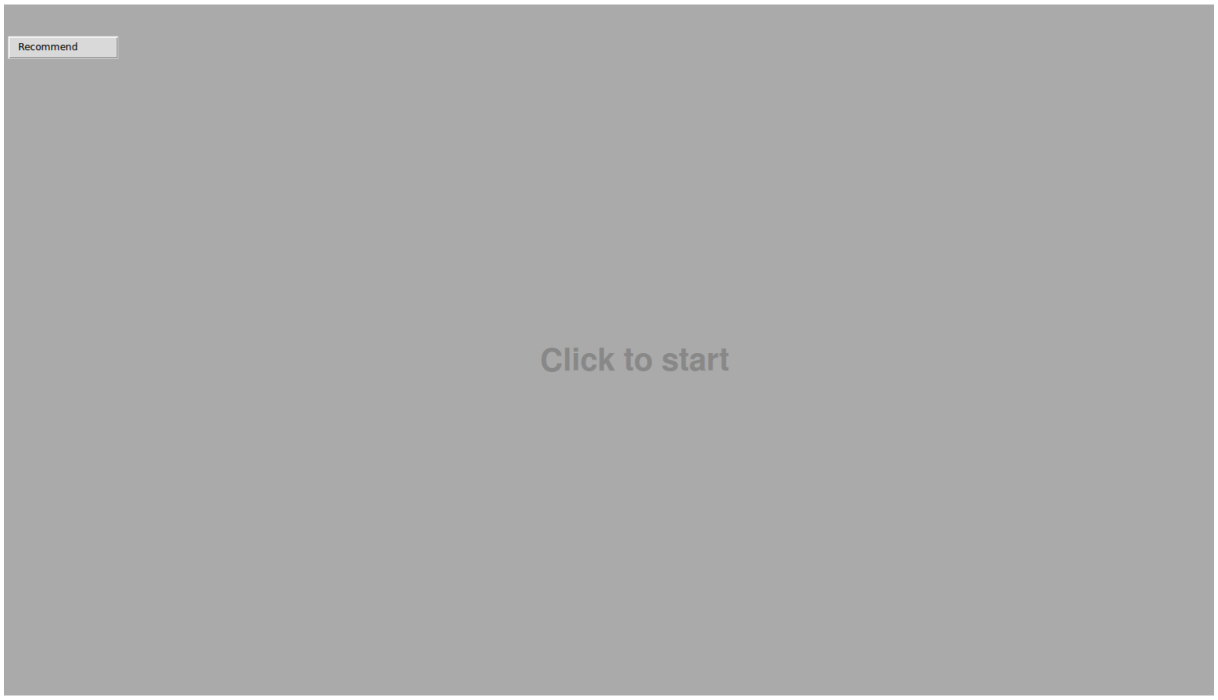


Figure 4.1: Initial interface of the framework.

To start building the workflow, users have to left click in the interface. That action will open a circle-shaped selection menu with all of the modules of the framework, as shown in Figure 4.2. By selecting one of the module, it will be added to the workflow design area.

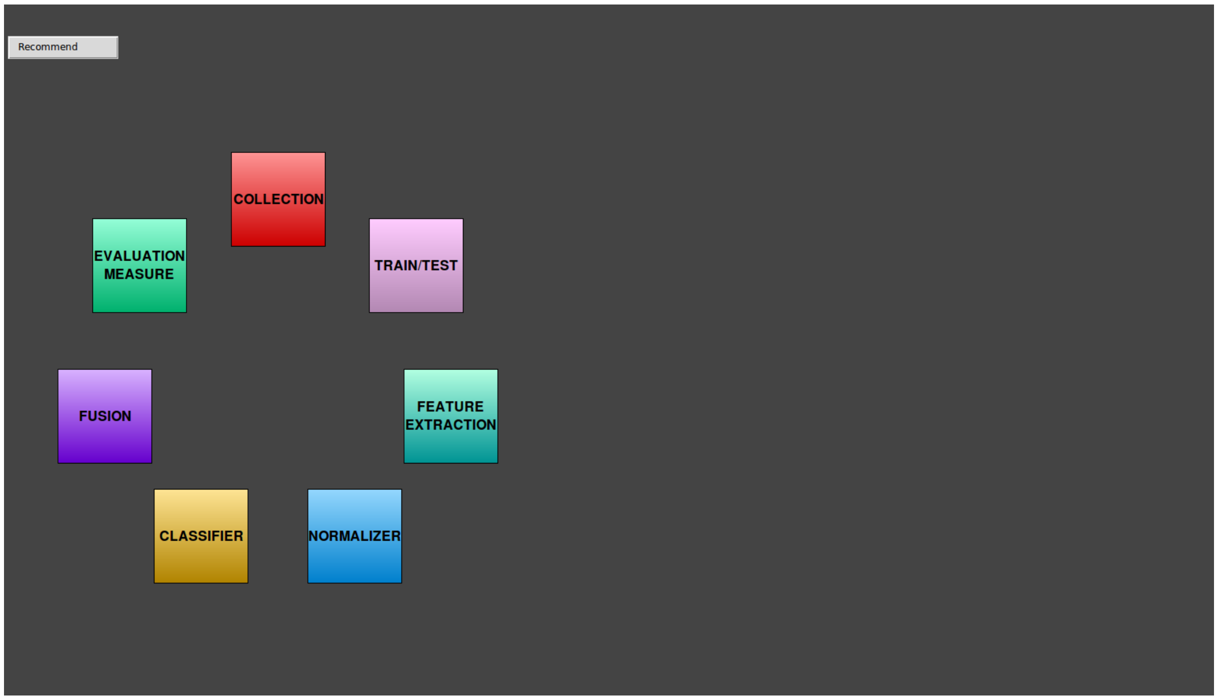


Figure 4.2: Modules of the framework.

For example, we selected the Collection Module. Right clicking on this module box opens a window with a list of available collections in the framework (Figure 4.3). For the other modules, users will be able to select one of the available plugins.

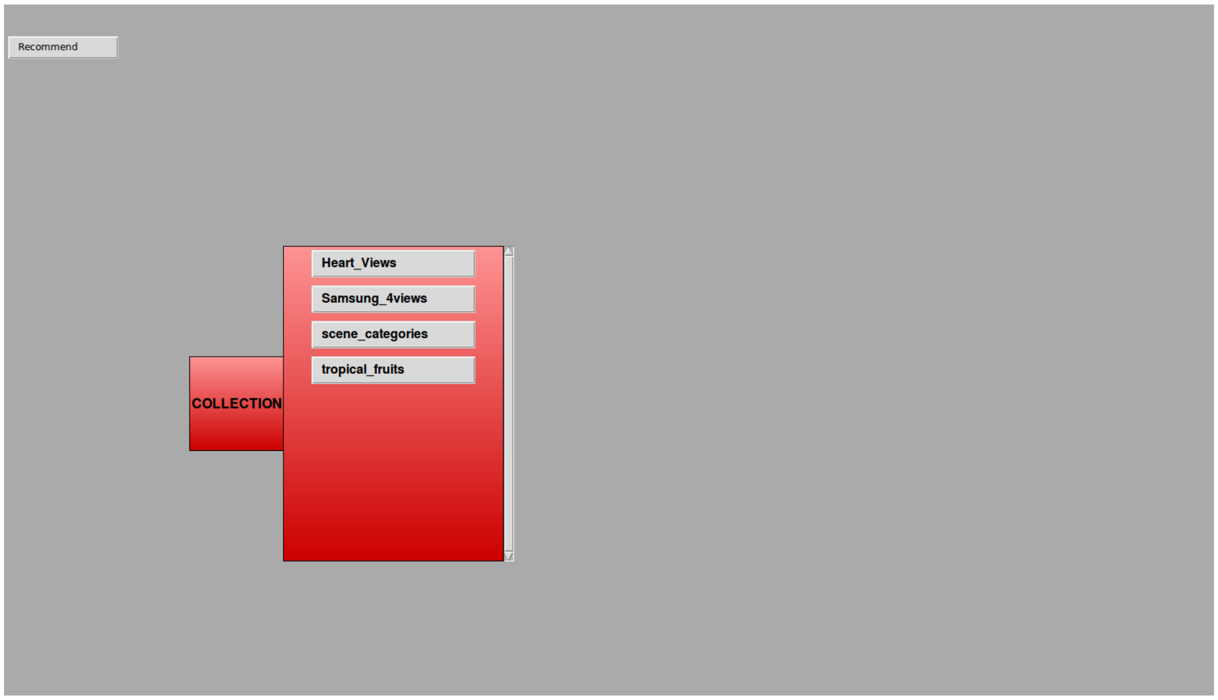
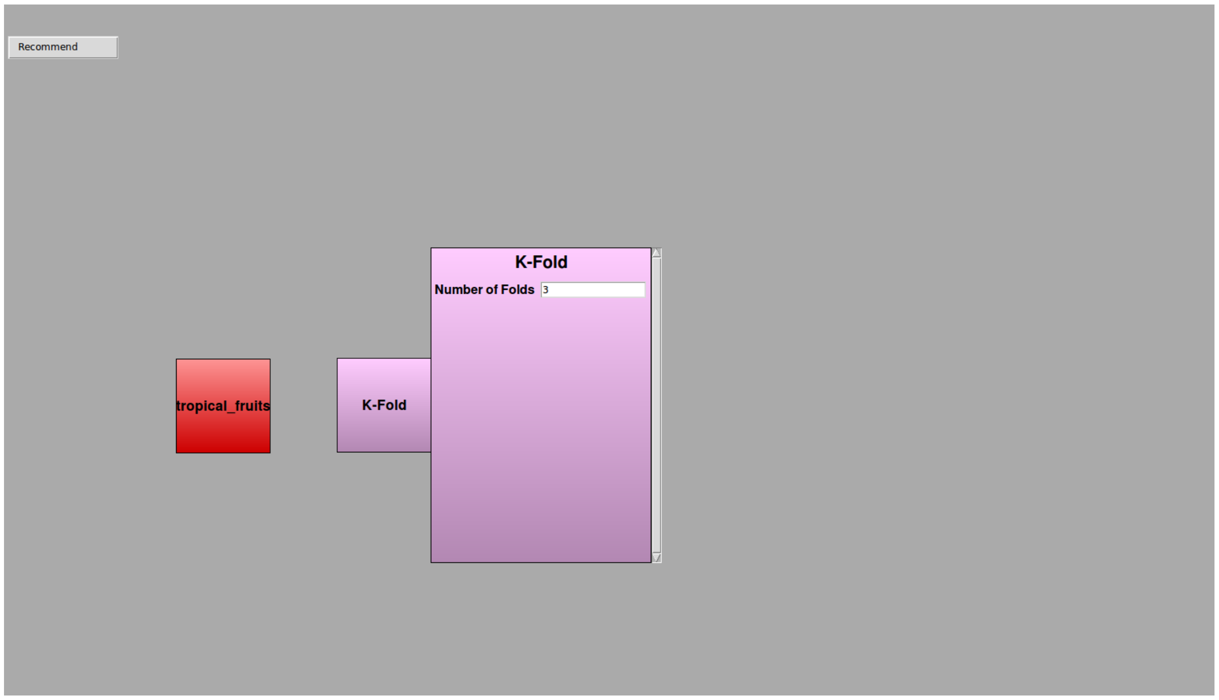


Figure 4.3: List of databases in the framework.

When selecting the collection (or plugin) in the module, it opens a window with the parameters of the collection or the plugin. Figure 4.4 shows the parameters of the K-Fold plugin under the Train-and-Test Module.



Number of Folds 3

Figure 4.4: Parameters of the K-Fold plugin.

With two modules built in the framework, it is possible to connect them. For that, left clicking in the origin module creates a directed edge attached to the mouse cursor, that can be linked to other module by left clicking in this target module. Figure 4.5 shows a link between the Database Module and the Train-and-Test Module, with the Tropical Fruits database and K-Fold method, respectively.

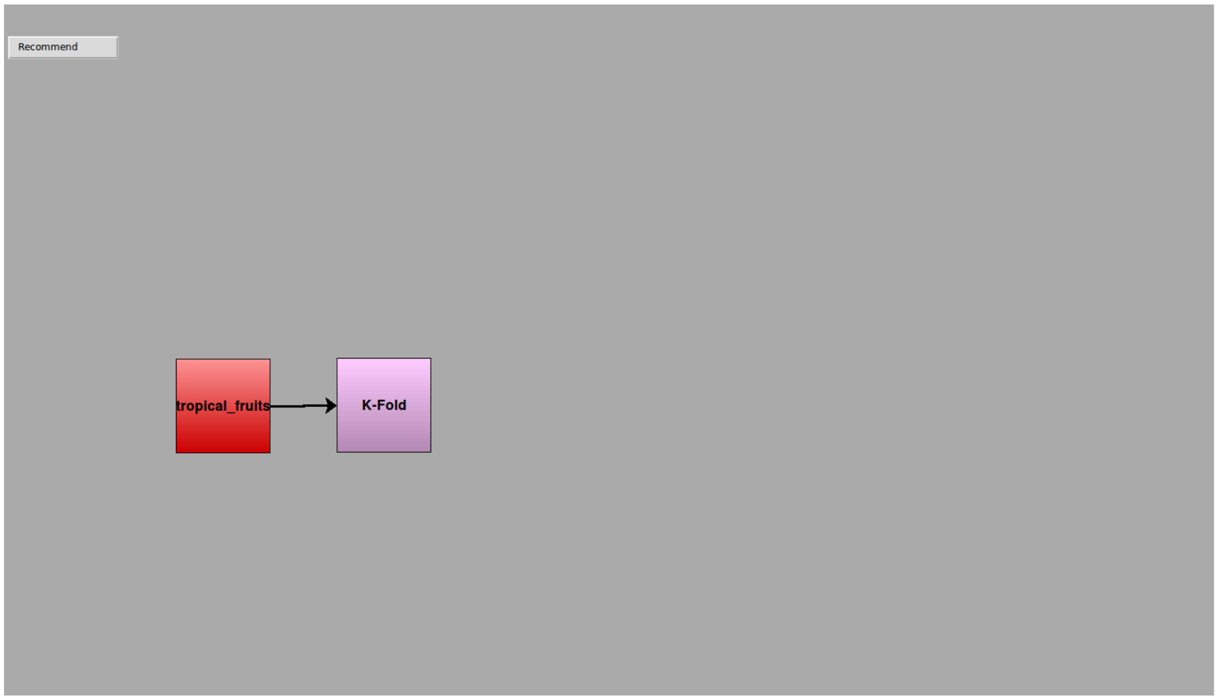


Figure 4.5: Two modules of the framework linked. The Collection Module (Tropical Fruits) is used as input for the Train-and-Test Module (K-Fold) in the execution.

Repeating these steps, it is possible to create a workflow representing a machine learning experiment. Figure 4.6 shows a simple workflow experiment, using the two modules added previously, with a BIC descriptor, Min-Max normalizer, libSVM classifier, and two evaluation measures, Global Accuracy Score and Confusion Matrix, with the parameters described in Section 4.1.1.

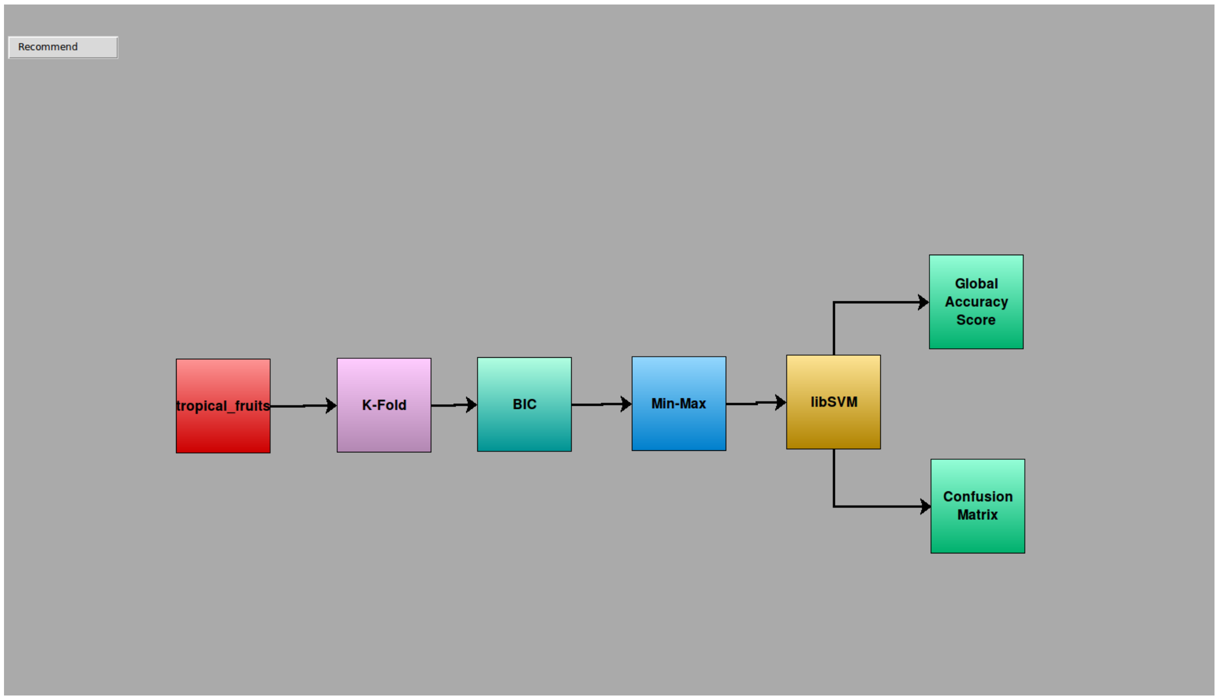


Figure 4.6: A complete machine learning experiment built in the framework.

In order to compare the classification results of the color descriptor and the texture descriptor in this collection, we insert another Feature Extraction Module in the workflow, with the LAS plugin, and create a link from the K-Fold to this new module, as shown in Figure 4.7.

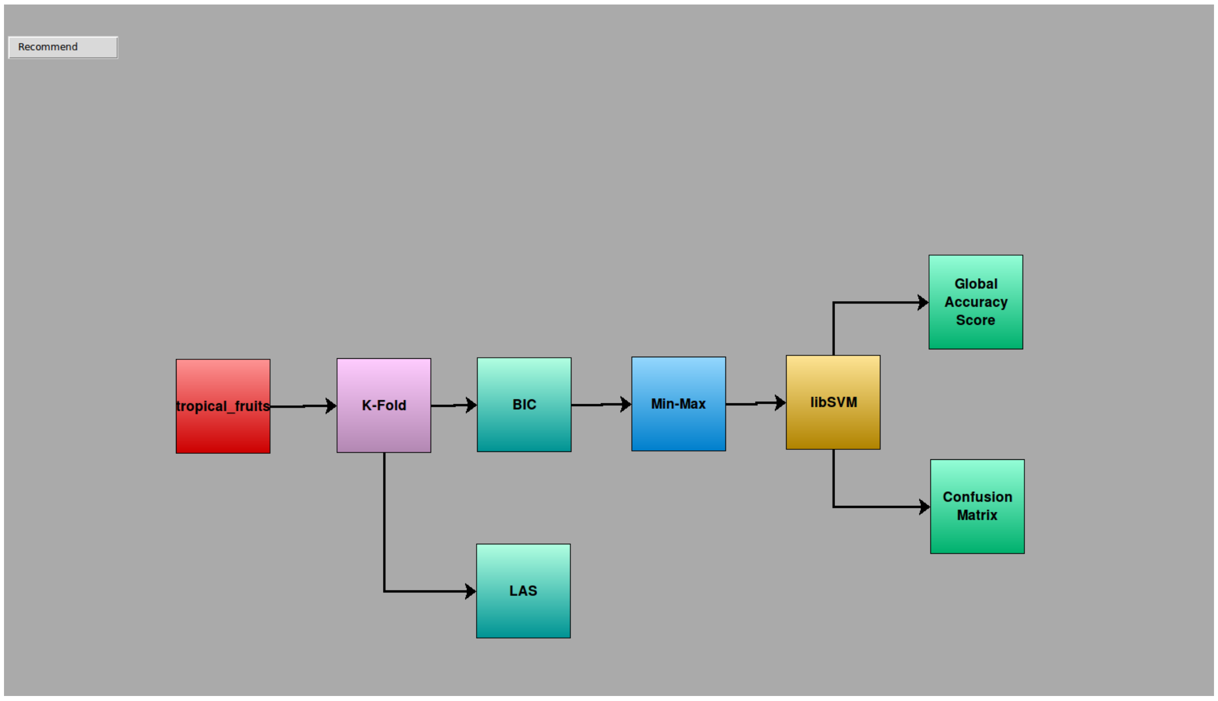


Figure 4.7: It is possible to build another machine learning experiment just by creating output edges from a module and connecting with another module.

Figure 4.8 presents a more complex workflow with two branches, one with a color feature extraction and other with texture feature extraction. This model, structured in branches, allows that the result of each branch be in the final PDF, making it easier to compare. It is important to notice that, even if the two branches use the same plugins for normalization and classification, two new activities were created to avoid any possible problems with the modification of its parameters.

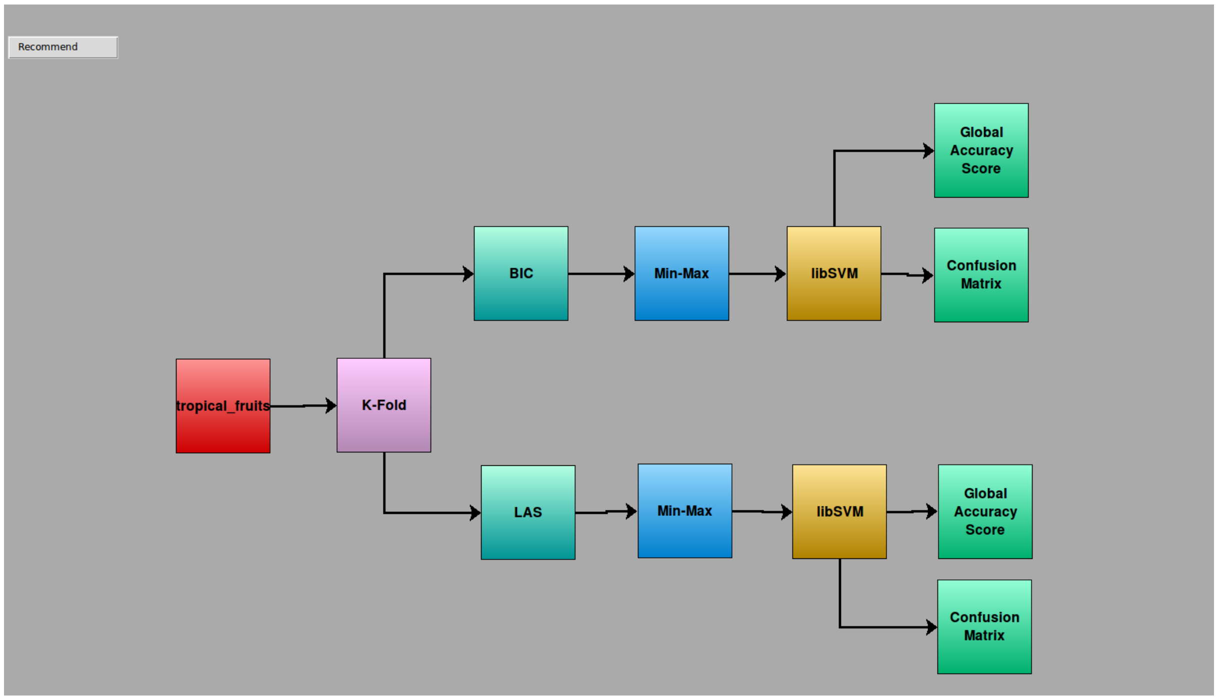


Figure 4.8: Complex machine learning experiment, to compare the results of two different feature extraction.

Listings C.1 in Appendix C presents the XML of the experiment built in Figure 4.8.

With the workflow completely built, users can execute the experiment. Figure 4.9 illustrates that. The user takes the mouse cursor to the top of the framework, where a bar appears. This bar contains two buttons: “Begin Experiment” and “Load Experiment”, a check box “Open-Set Experiment”, and four entries: “Number of Iterations”, “Experiment Name”, “Author”, and “Server IP”. Here we list the function of each of those entries.

Begin Experiment: Begin the execution of the experiment built in the framework;

Load Experiment: Load an experiment into the interface, selecting the XML file that describes the experiment;

Open-Set Experiment: Modify the experiment to a open-set experiment[1], making possible to modify the Database Module to select which classes in the database will be used as "known" by the framework, labeling the rest as "unknown";

Number of Iterations: Number of times that the workflow will be executed;

Experiment Name: Name for the experiment, given by the user;

Author: Name of the author of the experiment;

Server IP: IP for the interface, used in the communication of the framework with the interface.

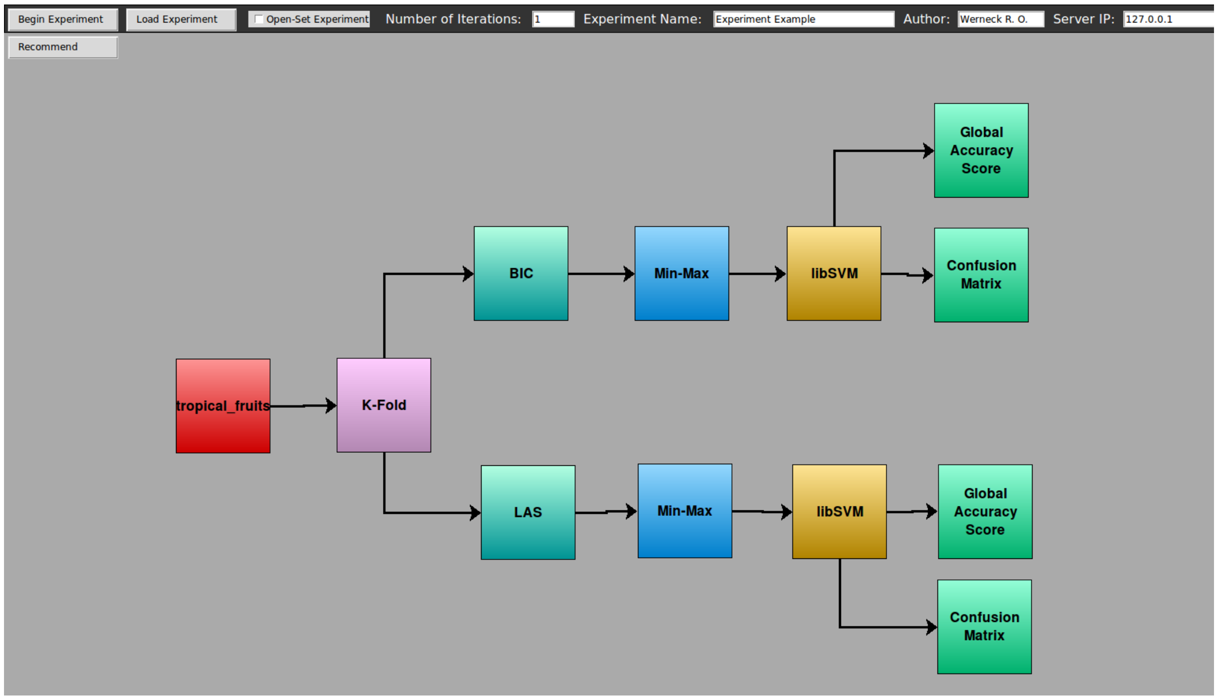

Number of Iterations: 1
Experiment Name: Experiment Example
Author: Werneck R. O.
Server IP: 127.0.0.1

Figure 4.9: Top bar of the machine learning framework, with parameters of the execution and a button to begin the experiment execution.

[1]An open-set recognition scenario is the one in which there are no a priori training samples for some classes that might appear during testing [104].

When the workflow is running (Figure 4.10), as soon as a module is executing, it receives a green borderline and the percentage of the execution below the plugins name.

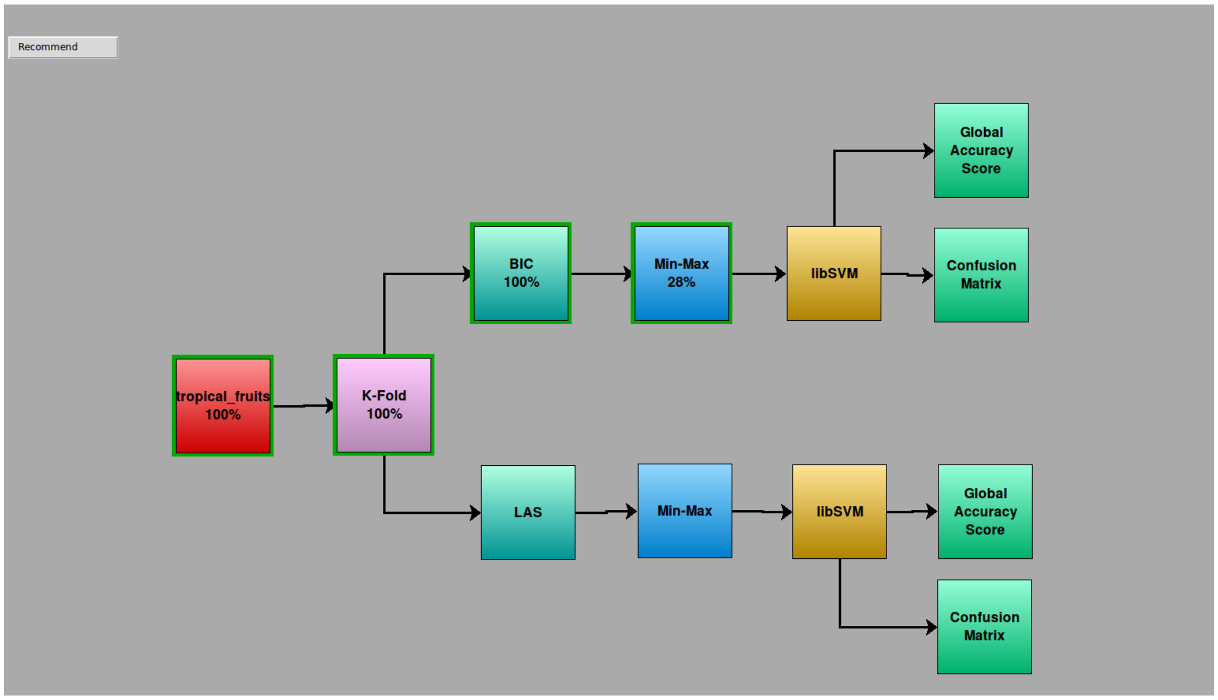


Figure 4.10: As the execution of the workflow goes, the modules being executed and already executed get a green borderline, with the percentage of the execution already done.

At the end of the execution of the workflow, the framework opens the PDF file generated with the results of the running, as shown in Appendix A. The appendix shows that the branch with the BIC method had a 98.48% of global accuracy against 74.86% of the branch with the LAS descriptor.

4.2 Integration with another workflow management system

With every module working independently of each other, the framework can be extended to be used with other workflow management systems, like VisTrails [9]. VisTrails [9] is an open-source tool that provides data and process management support for computational tasks. By combining features of workflows and visualization systems, it allows the combination of resources and libraries, and provides mechanisms to compare different results. The major difference of the VisTrails to other workflow systems is a comprehensive provenance infrastructure that maintains history and steps in the evolution of workflows and data in the course of exploratory tasks.

To extend the machine learning framework in another workflow management system, each module of the framework is integrated in the management system. With this, the execution of a machine learning experiment will be made in the framework, however, it is possible to take advantage of the features of the management system, e.g., in VisTrails, the provenance of the evolution of the workflow is the most important feature.

Figure 4.11 shows an experiment workflow considering both the framework interface and the VisTrails interface, implemented using the modules of the framework added in the VisTrails management system.

However, the VisTrails workflow management has some limitations, as it lacks a functionality for recommending previous experiments designed in the system.

4.3 Workflow recommendations

In this section, we present an overview of the recommendation system execution, with the methods used to perform the recommendation and the experiments made to evaluate the system.

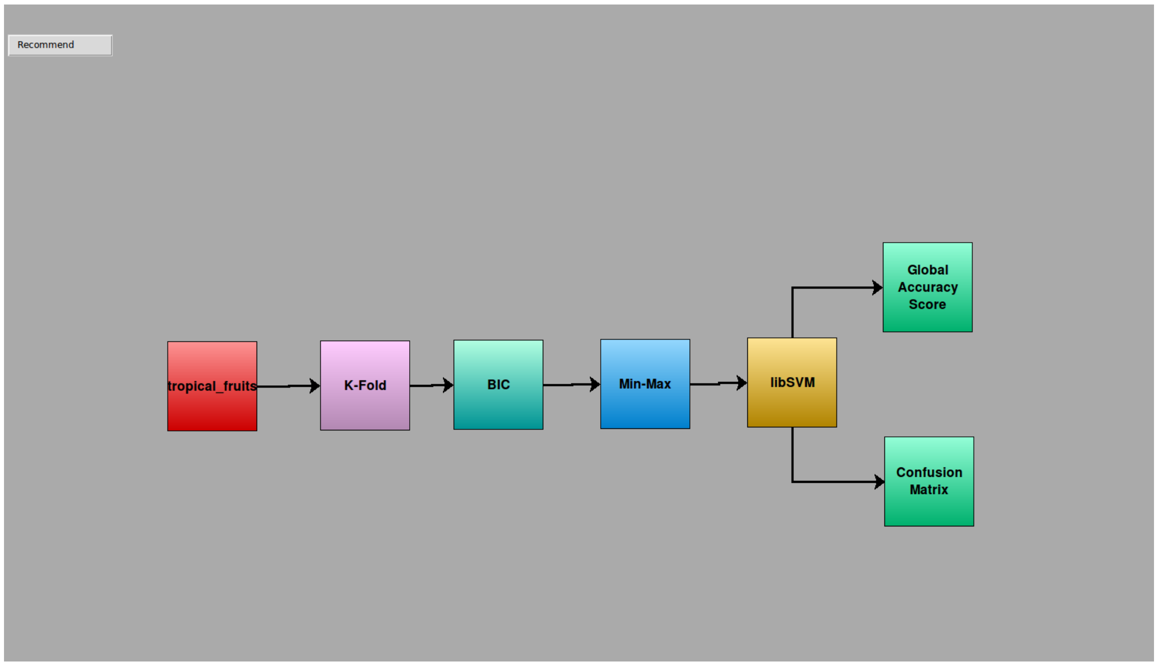
4.3.1 Overview

In this work, we describe the process of sequences of activities, more specifically, a workflow recommendation. The objective is to support the reuse of experiments and activities successfully used in the past, avoiding common mistakes in workflow design. To make the recommendations, we implemented four similarities measures to estimate the proximity of two workflow experiments and a *learning-to-rank* method, that “learns” how to rank workflows according to the user interests.

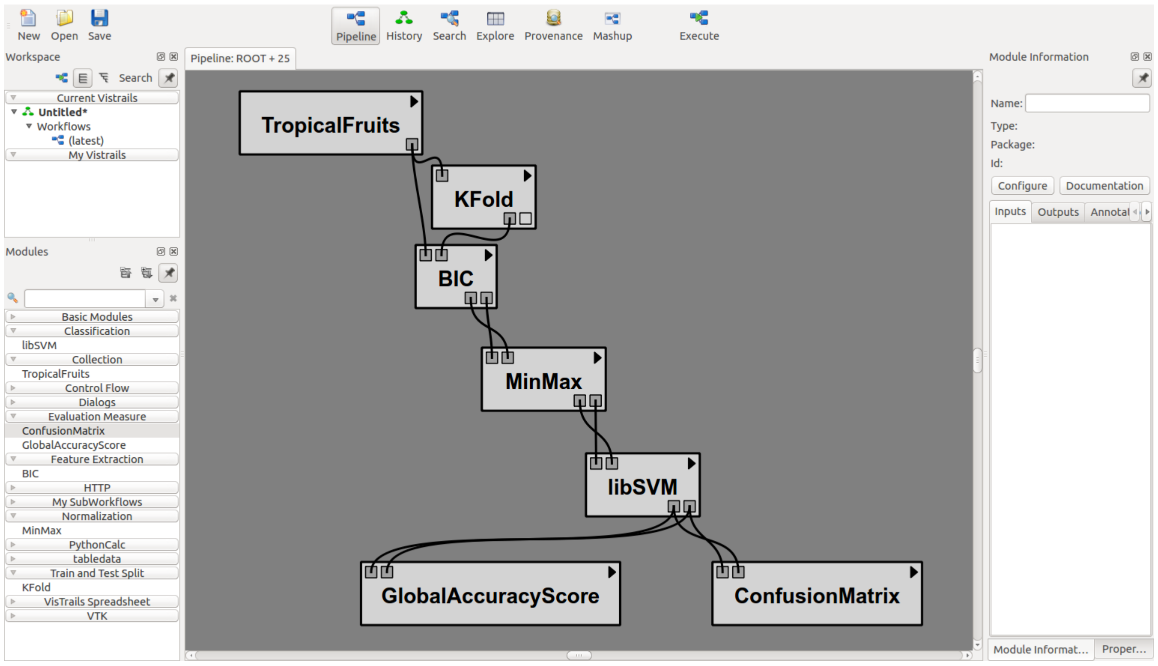
The similarities measures implemented in the framework consider that an experiment workflow is a textual sentence, in which each module is represented by a word in the sentence. To calculate the distance between two sentences, we implemented the Jaccard [57], Sørensen [108], and Jaro-Winkler [60,61,124] distances and a measure based on Term Frequency-Inverse Document Frequency (TF-IDF) [100] to represent an experiment workflow.

Jaccard Distance

Let A and B be two sequences, the Jaccard index calculates the similarity between these two sequences using Equation 4.1, and Equation 4.2 measures the distance between A and B.



(a) Experiment workflow in the machine learning framework.



(b) Experiment workflow in the VisTrails interface.

Figure 4.11: Example of an experiment workflow in the machine learning framework and in the VisTrails system.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

$$d_J(A,B) = 1 - J(A,B) \tag{4.2}$$

Sørensen Distance

This measure, represented by Equation 4.3, is used to compare the similarity of two samples (A and B) and was originally used to be applied to presence/absence data. Equation 4.4 represents the distance between two samples.

$$QS = \frac{2|A \cap B|}{|A| + |B|} \tag{4.3}$$

$$d_{QS} = 1 - QS \tag{4.4}$$

Jaro-Winkler Distance

The Jaro-Winkler distance measures the similarity between two strings. This similarity measure is composed of two algorithms. Equation 4.5 refers to the Jaro distance,

$$d_j(A,B) = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|A|} + \frac{m}{|B|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \tag{4.5}$$

where m is the number of matching characters, if they are the same and not farther than

$$\left\lfloor \frac{\max(|A|,|B|)}{2} - 1 \right\rfloor,$$

and t is the number of matches in different sequence order, divided by 2.

Equation 4.6 is an extension of the Jaro distance that gives favorable ratings to strings that match from the beginning.

$$d_w(A,B) = d_j(A,B) + (lp(1 - d_j(A,B))) \tag{4.6}$$

where l is the length of common prefix at the start of the string, and p is a constant scaling factor ($p = 0.1$).

TF-IDF-based Distance

This distance measure relies on the fact that, if a workflow is often used, it should be recommended. This measure calculates the Inverse Document Frequency of each item of the sequences in the previous experiments, measuring if the item is common or rare in the sequences. Let p be an item of a experiment workflow, s a experiment workflow, and S the set of previous experiments of the framework. The Inverse Document Frequency of each item is obtained by dividing the total number of sequences N_S by the number of sequences that contains the item, and taking the logarithm of the quotient (Equation 4.7).

$$idf(p, S) = \log \frac{N_S}{|s \in S : p \in s|} \tag{4.7}$$

The Term Frequency of an item in a experiment workflow ($tf(p, s)$) is represented by the raw frequency of the item in the sequence. Therefore the feature vector of a sequence is the product of this two statistics for each item of the sequence, as shown in Equation 4.8.

$$tf - idf(p, s, S) = tf(p, s) \times idf(p, S) \tag{4.8}$$

With the feature vector of two sequences, the distance between them is calculated using the Euclidean distance.

To perform the recommendation, the framework uses the workflow that is being built in the ongoing experiment configuration (the *query* workflow) to search for similar workflows in previous experiments. Figure 4.12 shows the “Recommend” button, that initiates the recommendation system, with the simple workflow experiment built in the framework.

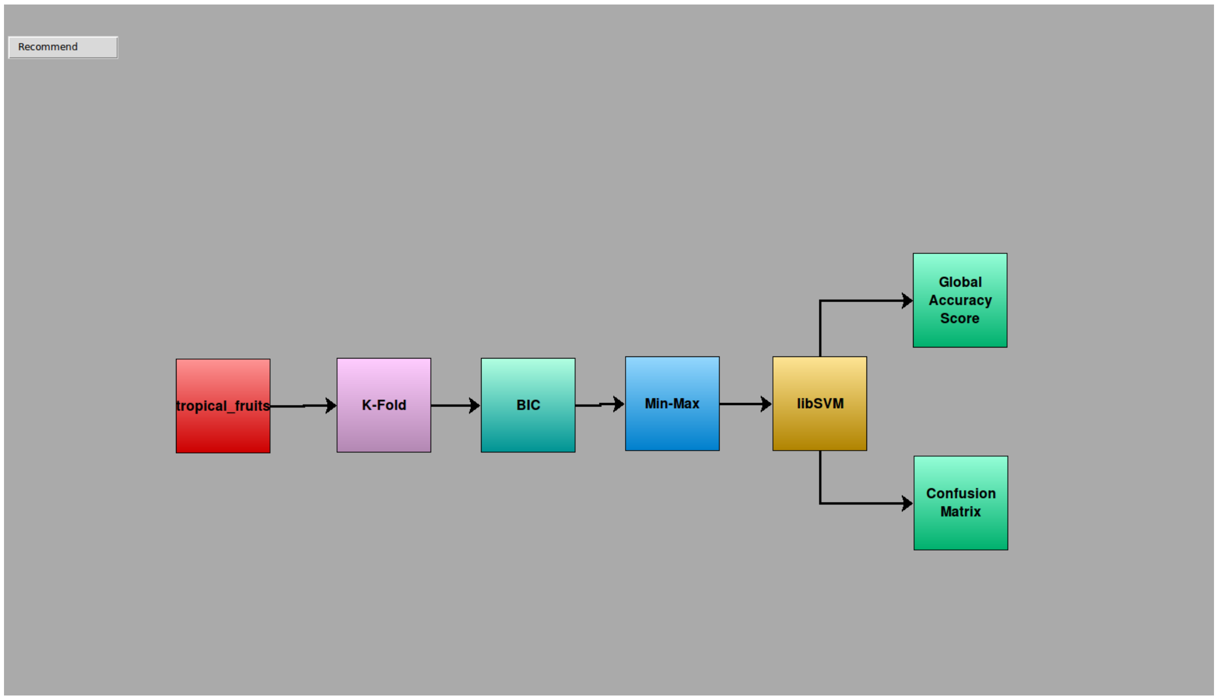
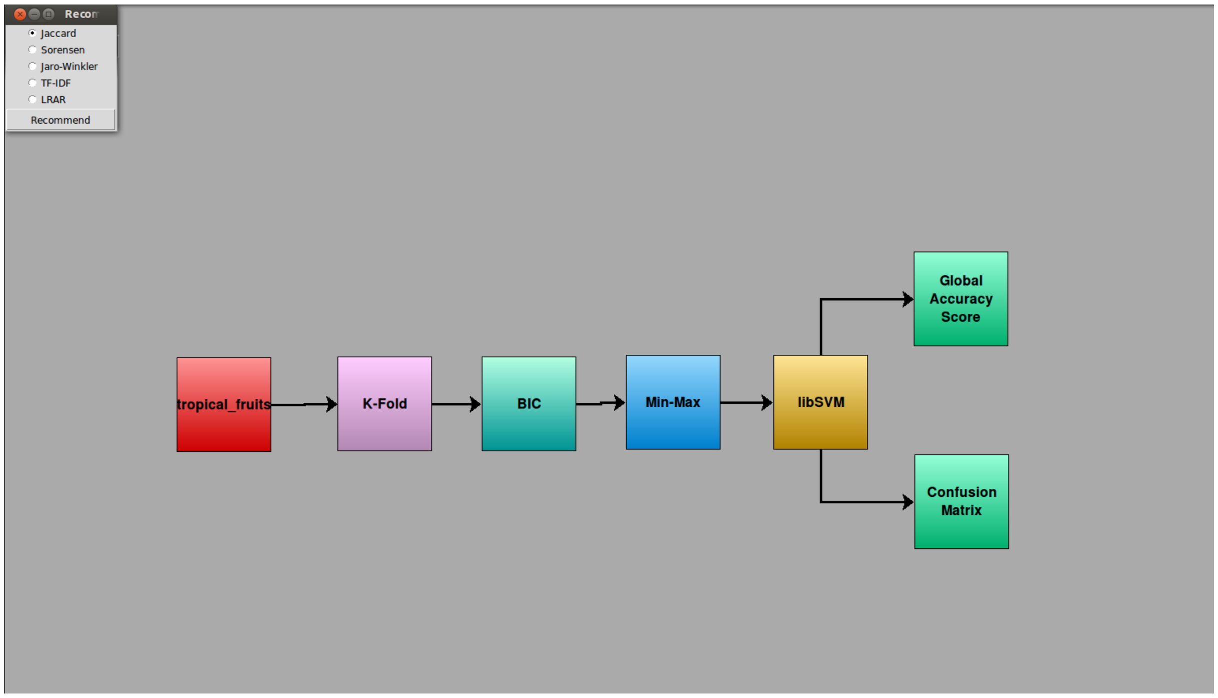


Figure 4.12: "Recommend" button must be used to begin the Recommendation Module of the framework. The Recommendation Module will use the experiment under configuration (the workflow in the center of the screen) as a query for searching for existing similar experiments.

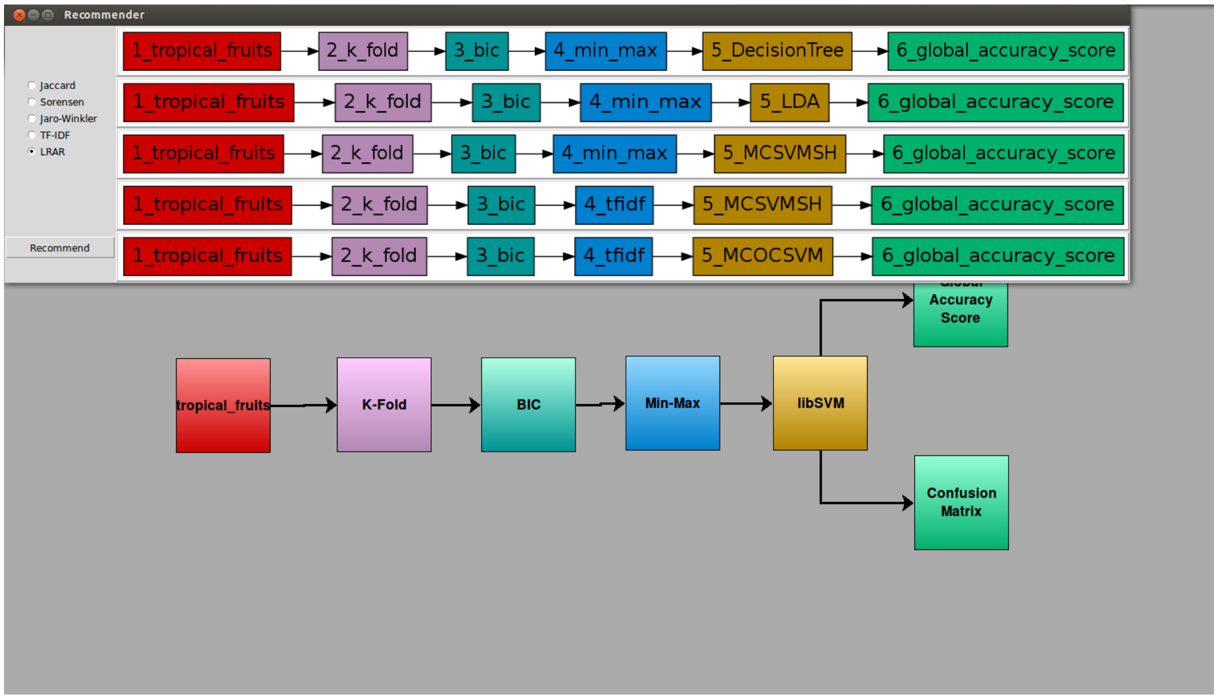
When the user clicks on the "Recommend" button, the framework opens a window with a list of recommendation plugins implemented in the framework. These plugins are responsible for ranking the previous workflows of the framework according to the query, and show them to the user. Figure 4.13 shows the list of plugins in the framework.



[✓ Jaccard] Sorensen Jaro-Winkler TF-IDF LRAR

Figure 4.13: “Recommender” window showing the plugin options that are implemented in the Recommendation Module.

When the user selects the plugin and clicks in the “Recommend” button in the new window, the framework starts the execution of the recommendation module. At the end of the execution, the framework lists five workflow experiments that are similar to the workflow that is being built in the framework, according to the selected method, as shown in Figure 4.14.


Recommender
1_tropical_fruits → 2_k_fold → 3_bic → 4_min_max → 5_DecisionTree → 6_global_accuracy_score
1_tropical_fruits → 2_k_fold → 3_bic → 4_min_max → 5_LDA → 6_global_accuracy_score
1_tropical_fruits → 2_k_fold → 3_bic → 4_min_max → 5_MCSVMSH → 6_global_accuracy_score
1_tropical_fruits → 2_k_fold → 3_bic → 4_tfidf → 5_MCSVMSH → 6_global_accuracy_score
1_tropical_fruits → 2_k_fold → 3_bic → 4_tfidf → 5_MCOCSVM → 6_global_accuracy_score
Recommend
tropical_fruits
K-Fold
BIC
Min-Max
libSVM
Accuracy Score
Confusion Matrix

○ Jaccard
○ Sorensen
○ Jaro-Winkler
○ TF-IDF
● LRAR

Figure 4.14: The five most similar existing experiments ranked according to the recommendation method selected. In this example, we can see that the LRAR method recommends very similar experiments, only changing the normalizer and the classifier.

4.3.2 Experiments

Our experiments aim to address two different research questions: (i) *Which similarity measure is more appropriate for ranking workflows modeled as a sequence of activities?* and (ii) *Is the use of learning-to-rank methods a suitable research venue for ranking workflows?*

In order to address the first question, we performed experiments in the recommendation system using four similarity measures (Jaccard, Sørensen, Jaro-Winkler, and a TF-IDF-based measure). For the second question, we performed experiments with the Learning to Rank using Association Rules (LRAR) with the objective of comparing its accuracy performance with the methods that do not use any learning mechanism.

To perform these experiments, we had to define a ground truth that indicates the relevance of experiments (workflows). To obtain this ground truth, we invited five specialists in machine learning experiments to label workflows as relevant or not for some queries. For this, we randomly created 1,000 workflow experiments, and selected 18 of those workflows as queries.

For each query workflow, we applied the four similarity measures (Jaccard, Sørensen, Jaro-Winkler, and a TF-IDF-based measure) with the 1,000 entries. For each measure, we ranked and selected the closest 20 workflows to the query, and presented these ranked workflows to all those specialists, so that they could label which ones are relevant for the query, as shown in Figure 4.15.

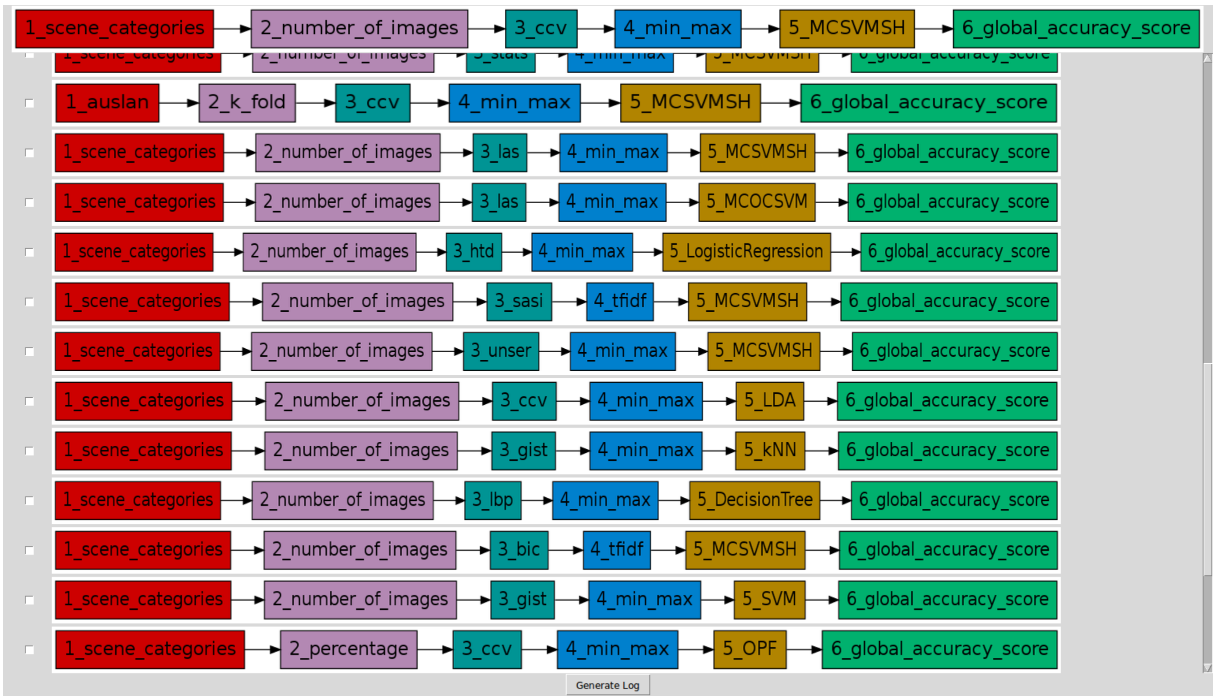

1_scene_categories → 2_number_of_images → 3_ccv → 4_min_max → 5_MCSVMSH → 6_global_accuracy_score
1_auslan → 2_k_fold → 3_ccv → 4_min_max → 5_MCSVMSH → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_las → 4_min_max → 5_MCSVMSH → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_las → 4_min_max → 5_MCOCSVM → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_htd → 4_min_max → 5_LogisticRegression → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_sasi → 4_tfidf → 5_MCSVMSH → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_unser → 4_min_max → 5_MCSVMSH → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_ccv → 4_min_max → 5_LDA → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_gist → 4_min_max → 5_kNN → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_lbp → 4_min_max → 5_DecisionTree → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_bic → 4_tfidf → 5_MCSVMSH → 6_global_accuracy_score
1_scene_categories → 2_number_of_images → 3_gist → 4_min_max → 5_SVM → 6_global_accuracy_score
1_scene_categories → 2_percentage → 3_ccv → 4_min_max → 5_OPF → 6_global_accuracy_score
Generate Log

Figure 4.15: Interface shown to the specialists. At the top of the window we have the query workflow, at the bottom there is a button to generate a file with the labels of the specialists, and between them a list of the workflows to be labeled.

To compare the results of different similarity methods with the LRAR method, we applied a majority voting scheme based on the labels provided by each specialist. One workflow is labeled as relevant if most of the specialists agree on that. At the end, we have the ground truth for 18 queries merging the results of the four similarity methods. These labels are then used to train the LRAR method.

With 18 queries, we split these queries into 5 folds, where 4 folds contains the training set of the LRAR method, and the fifth fold contains the test queries. The training set of the LRAR is composed of the queries of 4 folds, and each query has no more than 80 workflows (the 20 closest workflows for each similarity method). For each workflow, its features in the learning-to-rank method are the distance between the workflow to its query calculated by the similarity methods, and the relevance of the workflow is the majority voting of the relevance labeled by each user for each query. The LRAR method implemented uses a minimum confidence value θ and a minimum support value σ to limit the amount of rules created. We executed the LRAR method with 7 values of confidence $(0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5)$ and 5 values of support $(0.01, 0.05, 0.1, 0.15, 0.2)$. The P@5 value for the variation of the confidence and support are shown in Figure 4.16.

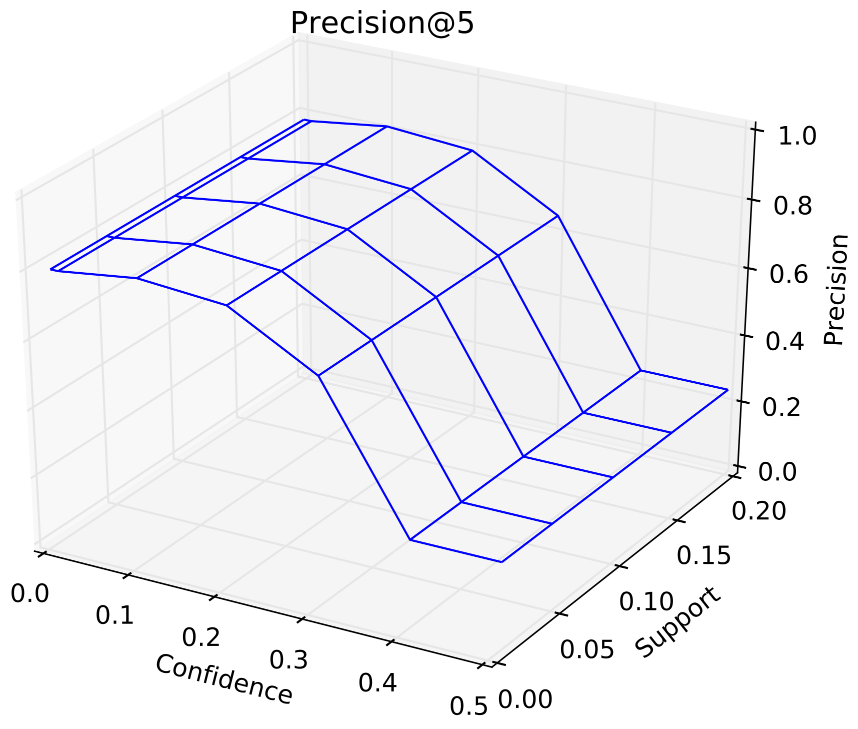


Figure 4.16: Precision of the LRAR method for each combination of confidence and support.

We can see in Figure 4.16 that the variation of the support value did not affect the result of the precision. However, a great increase in the value of the minimum confidence limits too much the power of the LRAR method, and the smaller values add noise of weak rules in the precision. To use the LRAR method, we selected the best confidence and support ($\min_\theta = 0.1$ and $\min_\sigma = 0.1$).

With the best result of the LRAR method, we can compare the precision (P@5, P@10, and P@20) of each query for the five implemented methods (Jaccard, Sørensen, Jaro-Winkler, TF-IDF-based, and LRAR), grouping them according to the folds. The results for the precision measure is shown in Table 4.1.

Fold 1	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.60	0.70	0.60	0.25	**0.75**
mean_P@10	**0.42**	**0.42**	**0.42**	0.25	0.40
mean_P@20	0.23	**0.29**	0.23	0.21	0.20

Fold 2	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.75	**0.90**	0.75	0.50	**0.90**
mean_P@10	**0.60**	0.57	**0.60**	0.43	0.55
mean_P@20	0.44	**0.52**	0.44	0.30	0.28

Fold 3	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.60	**0.85**	0.60	0.20	**0.85**
mean_P@10	0.50	**0.55**	0.50	0.25	0.48
mean_P@20	0.30	**0.42**	0.30	0.25	0.24

Fold 4	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.53	**0.67**	0.53	0.13	**0.67**
mean_P@10	**0.40**	**0.40**	**0.40**	0.27	**0.40**
mean_P@20	0.22	**0.27**	0.22	0.22	0.20

Fold 5	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.60	**0.87**	0.60	0.33	**0.87**
mean_P@10	**0.57**	**0.57**	**0.57**	0.37	0.53
mean_P@20	0.32	**0.38**	0.32	0.27	0.27

Mean of Folds	Jaccard	Jaro-Winkler	Sørensen	TF-IDF-based	LRAR
mean_P@5	0.62	0.80	0.62	0.28	**0.81**
mean_P@10	**0.50**	**0.50**	**0.50**	0.31	0.47
mean_P@20	0.30	**0.38**	0.30	0.25	0.24

Table 4.1: Precision of each similarity measure and LRAR for each fold.

We can conclude, by Table 4.1, that the Learning to Rank using Association Rules (LRAR) has the best performance, with precision $P@5$ of 81%, followed by the Jaro-Winkler similarity measure with $P@5$ of 80%. We noted that the relevant workflows are those that are very similar to the query, specially maintaining the early steps of the machine learning experiments (Database, Feature Extraction). These early steps are selected as they represent the core of the experiment according to the specialists.

Once we have the precision of all methods, we applied the Student's t-test and the Wilcoxon test to confirm if the results of the methods are significantly different for each other. We compared the methods in pairs, using a confidence of 95%. The results are shown in Figure 4.17.

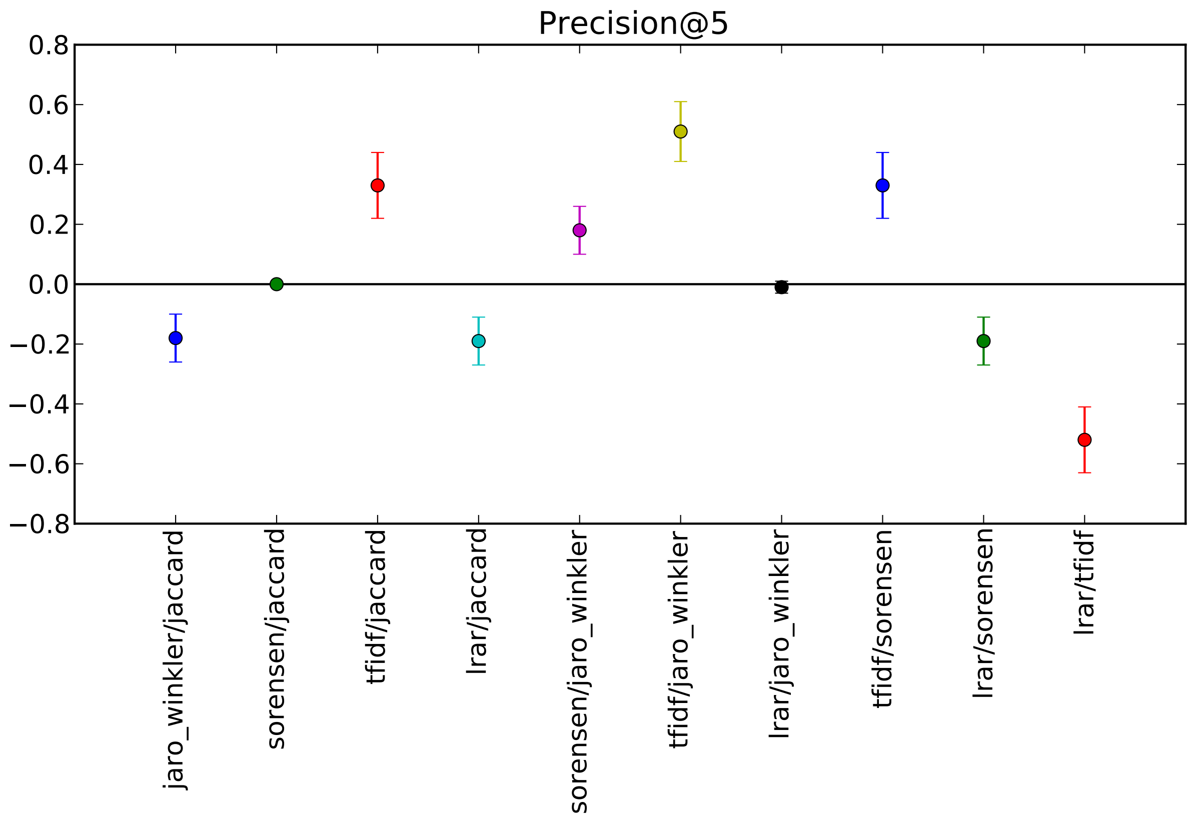


Figure 4.17: Student's t-test for the Precision@5, comparing all recommendation methods. Dots below the horizontal line indicate that the first method in the corresponding pair of the x-axis is better. Dots above the line indicate the opposite. If the error bar touches the horizontal line, there is no statistical difference between the two methods being compared. The Wilcoxon test confirmed the results obtained in the t-test.

The Student's t test in Figure 4.17 shows that the two methods with the best performance (LRAR and Jaro-Winkler) are not significantly different from each other, but significantly different from the others. The Wilcoxon test confirms the Student's results, with p-values above 0.05 in the Jaro-Winkler and LRAR comparison.

It is worth to mention that the LRAR method is consistent with the user-generated ground-truth. It recommends relevant workflows. The accuracy measures shown in Table 4.2 highlight this property of LRAR.

Folds	Accuracy
Fold-1	0.88
Fold-2	0.81
Fold-3	0.80
Fold-4	0.86
Fold-5	0.89
Mean	0.85

Table 4.2: Accuracy of the LRAR method.

Chapter 5

Conclusions

In this chapter, we present our contributions and discuss possible research directions for future work.

5.1 Contributions

Nowadays, we have to handle large and complex data sets that are difficult to process using some of the existing data analysis tools. To extract knowledge from this data, we usually perform machine learning experiments. There are several libraries and machine learning frameworks in the literature, however, they have some flaws, as they are not flexible for being extended with novel methods, and they often do not identify and reuse successful solutions devised in the past. In this work, we addressed these two flaws directly.

We have proposed a workflow-based framework for designing, deploying, executing, and recommending machine learning experiments. An important contribution of this work is the implementation of a tool that implements the proposed framework. This tool, as explained in the previous chapters, is able to provide a standardized environment for performing machine learning experiments. The tool makes it easy to evaluate different feature descriptors, normalizers, classifiers, fusion approaches in a wide range of tasks involving machine learning.

Another contribution is the evaluation of similarity measures and a learning-to-rank method in a recommendation scenario, in which it makes the recommendation of machine learning experiments modeled as a sequence of activities. We compared the performance of four similarity measures (Jaccard, Sørensen, Jaro-Winkler, and a TF-IDF-based measure) and the learning-to-rank method LRAR. Among the similarity measures, Jaro-Winkler had the best performance, with a precision $P@5$ of 80%, and the LRAR method obtained precision $P@5$ of 81%. With these precision values, we applied the Student's t test and the Wilcoxon test to confirm that these two methods are not significantly different from each other. A good result in the recommendation system can help beginner users, and also experienced ones, to design more effective machine learning experiments, presenting possible workflows used previously in the framework.

5.2 Future Work

Several research venues can be addressed for future work. Some of them are listed below:

- We propose the use of other recommendation measure, aside the distance between the workflows, such as the global accuracy and the number of false positives of an experiment.
- We propose the study of other *learning-to-rank* techniques, like RankSVM [50, 62], AdaRank [129] or RankBoost [43]. Another strategy concerns the use of rank aggregation approaches to combine ranked lists defined by different similarity functions.
- Another important research direction concerns the use of graph-based methods to compare workflows using machine learning techniques. With that, it is possible to improve the recommendation and it is possible to recommend pieces of a workflow, improving some steps of the machine learning experiment.
- In different applications, the design of an appropriate machine solution may be a time-consuming task. We propose the investigation of strategies for automatically designing a workflow-based solution for a given problem. We propose the use of evolutionary techniques to search for suitable workflows for a given target application. For this, we will generate random workflow experiments, select the ones with better performance (e.g., greater accuracy or fewer false positives) to reproduction, breed new workflows, evaluate these new experiments, and replace some of them from the population with the breed ones.
- To improve the usability of the framework, we propose the investigation of novel visualization approaches for guiding the user in the design of a workflow experiment. The objective is to provide an overview regarding the behavior of each step of an experiment. One starting point would be the use of similarity trees as proposed in [90].
- We also plan to incorporate other plugins so that the tool can be used for designing and executing more complex machine learning experiments. In special, we would like to incorporate meta-recognition methods [102], expanding the range of tasks to be performed in the framework.

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 22(2):207–216, June 1993.

[3] Selim Aksoy and Robert M Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582, 2001.

[4] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. *Computing Research Repository*, 2010.

[5] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423–424. IEEE, 2004.

[6] Naomi S Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, 1992.

[7] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search*. Pearson Education Ltd., Harlow, England, 2 edition, December 2010.

[8] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[9] Louis Bavoil, Steven P. Callahan, Patricia J. Crossno, Juliana Freire, Carlos Eduardo Scheidegger, Cláudio T. Silva, and Huy T. Vo. Vistrails: Enabling interactive

multiple-view visualizations. In *16th IEEE Visualization Conference*, pages 135–142. IEEE, IEEE Computer Society, October 2005.

[10] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. Knime: The konstanz information miner. In Christine Preisach, Hans Burkhardt, Lars Schmidt-Thieme, and Reinhold Decker, editors, *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 319–326. Springer Berlin Heidelberg, 2008.

[11] Daniel Billsus, Clifford A. Brunk, Craig Evans, Brian Gladish, and Michael Pazzani. Adaptive interfaces for ubiquitous web access. *Communications of the ACM*, 45(5):34–38, May 2002.

[12] Yu Bo and Qi Luo. Personalized web information recommendation algorithm based on support vector machine. In *The 2007 International Conference on Intelligent Pervasive Computing*, pages 487–490. IEEE, IEEE Computer Society, October 2007.

[13] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013.

[14] Johan Bollen, Michael L. Nelson, Gary Geisler, and Raquel Araujo. Usage derived recommendations for a video digital library. *Journal of Network and Computer Applications*, 30(3):1059 – 1083, August 2007.

[15] Carl R Boyd, Mary Ann Tolson, and Wayne S Copes. Evaluating trauma care: the triss method. *Journal of Trauma-Injury, Infection, and Critical Care*, 27(4):370–378, 1987.

[16] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, July 1998. Morgan Kaufmann Publishers Inc., Morgan Kaufmann Publishers Inc.

[17] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees.* Wadsworth, 1984.

[18] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, volume 119

of *ACM International Conference Proceeding Series*, pages 89–96, New York, NY, USA, August 2005. ACM.

[19] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[20] Steven P. Callahan, Juliana Freire, Emanuele Santos, Carlos Eduardo Scheidegger, Cláudio T. Silva, and Huy T. Vo. Managing the Evolution of Dataflows with VisTrails. In Roger S Barga and Xiaofang Zhou, editors, *Proceedings of the 22nd International Conference on Data Engineering Workshops*, page 71. IEEE, IEEE Computer Society, April 2006.

[21] Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. Sasi: a new texture descriptor for content based image retrieval. In *Proceedings of the International Conference on Image Processing*, volume 2, pages 137–140, 2001.

[22] Abdurrahman Çarkacıoğlu and Fatoş Yarman-Vural. Sasi: a generic texture descriptor for image retrieval. *Pattern Recognition*, 36(11):2615–2633, 2003.

[23] Sílvio César Cazella and Eliseo Berni Reategui. Sistemas de recomendação. In *XXV Congresso da Sociedade Brasileira de Computação*, 2005.

[24] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[25] Ruey-Ming Chao, Jen-Tu Huang, and Chin-Wen Yang. The study of knowledge service-oriented recommendation mechanism - a case of e-learning platform. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2228–2233 Vol. 4. IEEE, August 2005.

[26] Savvas Chatzichristofis and Yiannis Boutalis. Cedd: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *Computer Vision Systems*, pages 312–322. Springer, 2008.

[27] Hung-Chen Chen and Arbee LP Chen. A music recommendation system based on music data grouping and user interests. In *Proceedings of the 10th ACM CIKM International Conference on Information and Knowledge Management*, volume 1, pages 231–238. ACM, November 2001.

[28] Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[29] Thomas Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[30] Ryan R Curtin, James R Cline, Neil P Slagle, William B March, Parikshit Ram, Nishant A Mehta, and Alexander G Gray. MLPACK: A scalable C++ machine learning library. *Journal of Machine Learning Research*, 14(1):801–805, March 2013.

[31] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[32] Filipe de O. Costa, Michael Eckmann, Walter J. Scheirer, and Anderson Rocha. Open set source camera attribution. In *Proceedings of the 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 71–78, Aug 2012.

[33] Filipe de O. Costa, Ewerton Silva, Michael Eckmann, Walter J. Scheirer, and Anderson Rocha. Open set source camera attribution and device linking. *Pattern Recognition Letters*, 39(0):92 – 101, 2014.

[34] Daniel de Oliveira, Kary A.C.S. Ocaña, Eduardo Ogasawara, Jonas Dias, João Gonçalves, Fernanda Baião, and Marta Mattoso. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. *Future Generation Computer Systems*, 29(7):1816 – 1825, 2013.

[35] Daniel de Oliveira, Eduardo S. Ogasawara, Fernanda Araujo Baião, and Marta Mattoso. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*, pages 378–385, July 2010.

[36] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA, April 2008. ACM.

[37] The Economist. Data, data everywhere. `http://www.economist.com/node/15557443`, February 2010.

[38] Johan Eker, Jorn W Janneck, Edward A. Lee, Jie Liu, Xiaojun Liu, Jozsef Ludvig, Sonia Sachs, Yuhong Xiong, and Stephen Neuendorffer. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[39] Weiguo Fan, Michael D. Gordon, and Praveen Pathak. Genetic programming-based discovery of ranking functions for effective web search. *Journal of Management Information Systems*, 21(4):37–56, Spring 2005.

[40] Fábio Augusto Faria, Jefersson Alex dos Santos, Anderson Rocha, and Ricardo da Silva Torres. Automatic classifier fusion for produce recognition. In *Proceedings of the 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 252–259, Aug 2012.

[41] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006.

[42] Ronald Aylmer Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8(4):376–386, 1938.

[43] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, November 2003.

[44] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[45] Marcos André Gonçalves, Edward A. Fox, Layne T. Watson, and Neill A. Kipp. Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. *ACM Transactions on Information Systems*, 22(2):270–312, April 2004.

[46] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter*, 11(1):10–18, November 2009.

[47] Abhay S. Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 91–98, New York, NY, USA, July 2008. ACM.

[48] Amber L Hartman, Sean Riddle, Timothy McPhillips, Bertram Ludäscher, and Jonathan A Eisen. Introducing waters: a workflow for the alignment, taxonomy, and ecology of ribosomal sequences. *BMC bioinformatics*, 11(1):317, 2010.

[49] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2nd edition, 2008.

[50] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large Margin Rank Boundaries for Ordinal Regression. In *Advances in Large-Margin Classifiers*, chapter 7, pages 115–132. MIT Press, January 2000.

[51] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[52] Shohei Hido, Seiya Tokui, and Satoshi Oda. Jubatus: An open source platform for distributed online machine learning. In *NIPS Workshop on Big Learning*, 2013.

[53] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201. ACM Press/Addison-Wesley Publishing Co., ACM/Addison-Wesley, May 1995.

[54] Chao-Bing Huang and Quan Liu. An orientation independent texture descriptor for image retrieval. In *Proceedings of the International Conference on Communications, Circuits and Systems*, pages 772–776, July 2007.

[55] Jing Huang, S.R. Kumar, M. Mitra, Wei-Jing Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, Jun 1997.

[56] Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 65–73. ACM, ACM, June 2002.

[57] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[58] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[59] Mohsen Jamali and Martin Ester. *TrustWalker*: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 397–406, New York, NY, USA, July 2009. ACM.

[60] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[61] Matthew A Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

[62] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, July 2002.

[63] Daniel S. Kaster, Claudia B. Medeiros, and Heloisa V. Rocha. Supporting modeling and problem solving from precedent experiences: the role of workflows and case-based reasoning. *Environmental Modelling & Software*, 20(6):689 – 704, 2005.

[64] Donald Ervin Knuth. *Computers & Typesetting: The TEXbook*, volume A. Addison Wesley, Reading, Massachusetts, 1986.

[65] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 165–172, New York, NY, USA, October 2011. ACM.

[66] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, pages 1137–1143, New York, NY, USA, 1995. ACM.

[67] Vassili Kovalev and Stephan Volmer. Color co-occurrence descriptors for querying-by-example. In *Proceedings of the International Conference on Multimedia Modeling*, pages 32–38, Oct 1998.

[68] Ludmila I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.

[69] Anísio Lacerda, Marco Cristo, Marcos André Gonçalves, Weiguo Fan, Nivio Ziviani, and Berthier Ribeiro-Neto. Learning to advertise. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 549–556. ACM, ACM, August 2006.

[70] Pat Langley and Herbert A. Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64, November 1995.

[71] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[72] Shuchuan Lo and Chingching Lin. Wmr–a graph-based algorithm for friend recommendation. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 121–128, Washington, DC, USA, December 2006. IEEE Computer Society, IEEE Computer Society.

[73] Etienne Lord, Mickael Leclercq, Alix Boc, Abdoulaye Baniré Diallo, and Vladimir Makarenkov. Armadillo 1.1: An original workflow platform for designing and conducting phylogenetic analysis and simulations. *PLoS ONE*, 7(1):e29903, 01 2012.

[74] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 340–349. AUAI Press, July 2010.

[75] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Distributed graphlab: A framework for machine learning in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, April 2012.

[76] 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.

[77] Fariborz Mahmoudi, Jamshid Shanbehzadeh, Amir-Masoud Eftekhari-Moghadam, and Hamid Soltanian-Zadeh. Image retrieval based on shape similarity by edge orientation autocorrelogram. *Pattern Recognition*, 36(8):1725–1736, 2003.

[78] Claudia Bauzer Medeiros, José de Jesús Pérez Alcázar, Luciano A. Digiampietri, Gilberto Zonta Pastorello Jr., André Santanchè, Ricardo da Silva Torres, Edmundo Roberto Mauro Madeira, and Evandro Bacarin. Woodss and the web: Annotating and reusing scientific workflows. *ACM SIGMOD Record*, 34(3):18–23, September 2005.

[79] Claudia Bauzer Medeiros, Gottfried Vossen, and Mathias Weske. Wasa: A workflow-based architecture to support scientific database applications. In Norman Revell and AMin Tjoa, editors, *Database and Expert Systems Applications*, volume 978 of *Lecture Notes in Computer Science*, pages 574–583. Springer Berlin Heidelberg, 1995.

[80] Franck Michel. How many public photos are uploaded to flickr every day, month, year? `https://www.flickr.com/photos/franckmichel/6855169886/`, April 2014.

[81] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, 2006.

[82] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 263–266, New York, NY, USA, January 2003. ACM, ACM.

[83] Bradley N. Miller, Joseph A. Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Transactions on Information Systems*, 22(3):437–476, July 2004.

[84] Javier A Montoya-Zegarra, Neucimar Jerônimo Leite, and Ricardo da S. Torres. Rotation-invariant and scale-invariant steerable pyramid decomposition for texture image retrieval. In *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 121–128, Oct 2007.

[85] Thomas Natschläger, Felix Kossak, and Mario Drobics. Extracting knowledge and computable models from data-needs, expectations, and experience. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, volume 1, pages 493–498. IEEE, July 2004.

[86] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–54, November 2004.

[87] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, Jul 2002.

[88] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[89] Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*, volume 128. UC Berkeley, August 1999.

[90] Jose Gustavo Paiva, Laura Florian, Hélio Pedrini, Guilherme P. Telles, and Rosane Minghim. Improved similarity trees and their application to visual data classification. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2459–2468, Dec 2011.

[91] Joao P Papa, Alexandre X Falcão, Paulo AV Miranda, Celso TN Suzuki, and Nelson DA Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. In *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 337–348. MCT/INPE, 2007.

[92] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *Proceedings of the 4th ACM International Conference on Multimedia*, MULTIMEDIA '96, pages 65–73, New York, NY, USA, 1996. ACM.

[93] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

[94] Daniel Carlos Guimarães Pedronette. Uma plataforma de serviços de recomendação para bibliotecas digitais. Master's thesis, Universidade Estadual de Campinas, March 2008.

[95] Foster Provost and Tom Fawcett. Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data*, 1(1):51–59, February 2013.

[96] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, October 1994. ACM.

[97] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1989.

[98] Anderson Rocha, Daniel C. Hauagge, Jacques Wainer, and Siome Goldenstein. Automatic fruit and vegetable classification from images. *Computers and Electronics in Agriculture*, 70(1):96 – 104, January 2010.

[99] Arun A Ross and Rohin Govindarajan. Feature level fusion of hand and face biometrics. In *Defense and Security*, pages 196–204. International Society for Optics and Photonics, 2005.

[100] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[101] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, August 2002. ACM.

[102] Walter J. Scheirer, Anderson Rocha, Ross J. Michaels, and Terrance E. Boult. Meta-recognition: The theory and practice of recognition score analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1689–1695, 2011.

[103] Walter J. Scheirer, Anderson Rocha, Ross J. Micheals, and Terrance E. Boult. Robust fusion: Extreme value theory for recognition score normalization. In *Proceedings of the 11th European Conference on Computer Vision*, pages 481–495, New York, NY, USA, September 2010. ACM.

[104] Walter J. Scheirer, Anderson Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, July 2013.

[105] Laura A. Seffino, Claudia Bauzer Medeiros, Jansle V. Rocha, and Bei Yi. WOODSS - a spatial decision support system based on workflows. *Decision Support Systems*, 27(1-2):105–123, November 1999.

[106] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217. ACM Press/Addison-Wesley Publishing Co., ACM/Addison-Wesley, May 1995.

[107] Ya-Yueh Shih and Duen-Ren Liu. Hybrid recommendation approaches: Collaborative filtering via valuable content information. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 217b–217b, 2005.

[108] Thorvald Sørensen. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Ist Application to Analyses of the Vegetation on Danish Commons*. Det Kongelige Danske Videnskabernes Selskab. Munksgaard, 1948.

[109] Kent A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 160–163, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[110] Renato O. Stehling, Mario A. Nascimento, and Alexandre X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *Proceedings of the 11th ACM CIKM International Conference on Information and Knowledge Management*, CIKM '02, pages 102–109. ACM, November 2002.

[111] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77 – 89, 1997.

[112] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[113] Bo Tao and Bradley W. Dickinson. Texture recognition and image retrieval using gradient indexing. *Journal of Visual Communication and Image Representation*, 11(3):327–342, September 2000.

[114] Mohammad A. Tayebi, Mohsen Jamali, Martin Ester, Uwe Glässer, and Richard Frank. Crimewalker: A recommendation model for suspect investigation. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 173–180, New York, NY, USA, October 2011. ACM.

[115] Roberto Torres, Sean M. McNee, Mara Abel, Joseph A. Konstan, and John Riedl. Enhancing digital libraries with techlens+. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '04, pages 228–236, New York, NY, USA, June 2004. ACM, ACM.

[116] Michael Unser. Sum and difference histograms for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):118–125, January 1986.

[117] Adriano Veloso, Humberto Mossri de Almeida, Marcos André Gonçalves, and Wagner Meira Jr. Learning to rank at query-time using association rules. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, ACM, July 2008.

[118] Jason Venner and Steve Cyrus. *Pro Hadoop*, volume 1. Springer, 2009.

[119] Bruno S. C. M. Vilar, Claudia Bauzer Medeiros, and André Santanchè. Towards adapting scientific workflow systems to healthcare planning. In *HEALTHINF*, pages 75–84, 2013.

[120] Jacques Wainer, Mathias Weske, Gottfried Vossen, and Claudia Bauzer Medeiros. Scientific workflow systems, 1996.

[121] Tom White. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 1st edition, 2009.

[122] Adam Williams and Peter Yoon. Content-based image retrieval using joint correlograms. *Multimedia Tools and Applications*, 34(2):239–248, August 2007.

[123] Graham J. Williams. *Data Mining with Rattle and R: The art of excavating data for knowledge discovery.* Use R! Springer, 2011.

[124] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990.

[125] Workflow Management Coalition. Terminology and Glossary Document Number WFMC-TC-1011. Technical Report 3.0, Workflow Management Coalition, February 1999.

[126] Bing Wu, Luo Qi, and Xiong Feng. Personalized recommendation algorithm based on svm. In *International Conference on Communications, Circuits and Systems*, pages 951–953. IEEE, July 2007.

[127] Peng Wu, Bangalore S. Manjunath, Shawn Newsam, and Hyundoo Shin. A texture descriptor for browsing and similarity retrieval. *Signal Processing: Image Communication*, 16(1):33–43, 2000.

[128] Yan-Wen Wu, Qi Luo, Min Liu, Zheng-Hong Wu, and Li-Yong Wan. Research on personalized service system in e-supermarket by using adaptive recommendation algorithm. In *International Conference on Machine Learning and Cybernetics*, pages 4507–4510, August 2006.

[129] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[130] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and Hans-Peter Peter Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56–69, January 2004.

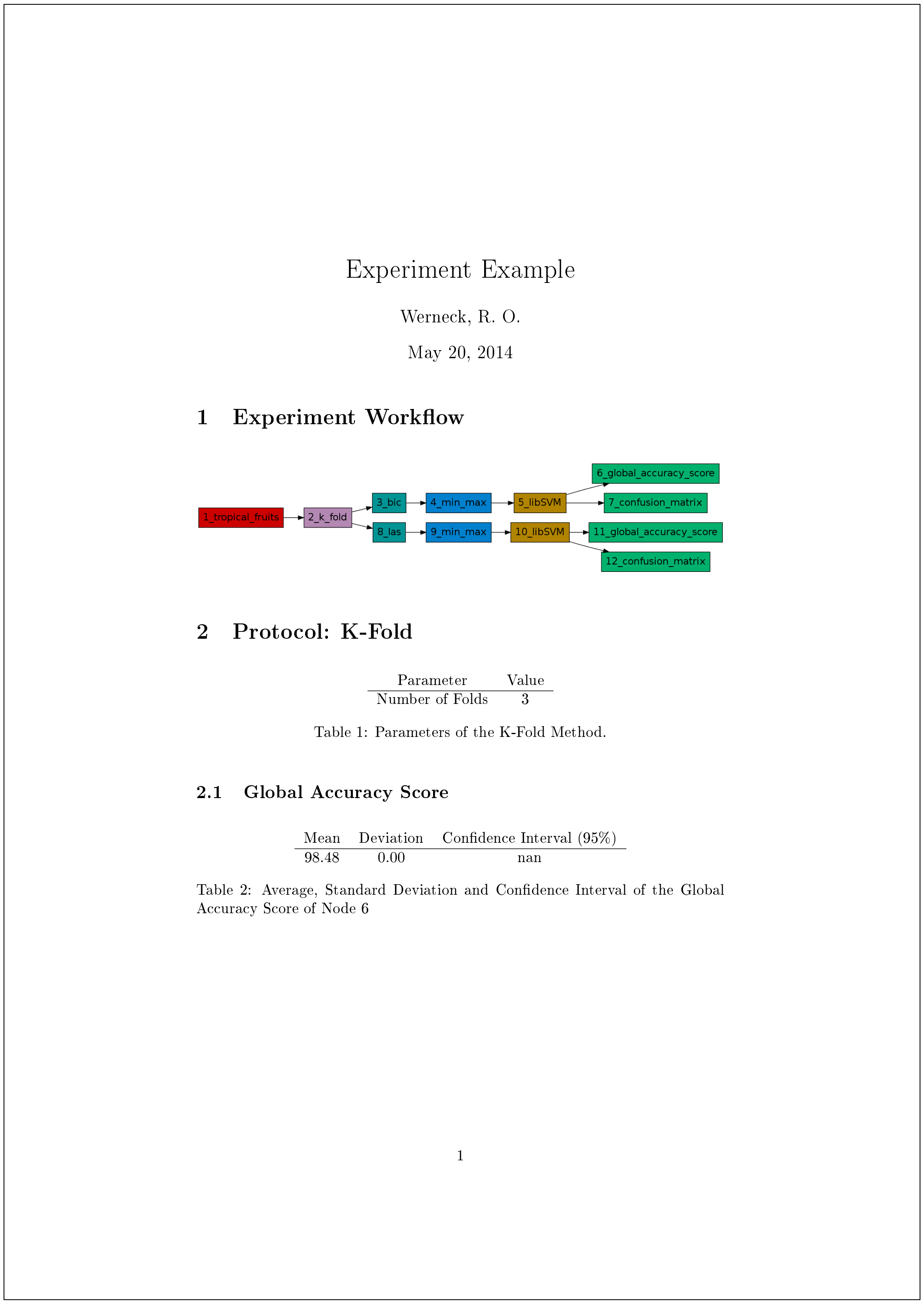
[131] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 271–278. ACM, July 2007.

[132] Massimiliano Zanin, Pedro Cano, Javier M Buldú, and Oscar Celma. Complex networks in recommendation systems. In *Modern Topics of Computer Science*, pages 120–124. World Scientific and Engineering Academy and Society, January 2008.

Appendix A

Experiment Result

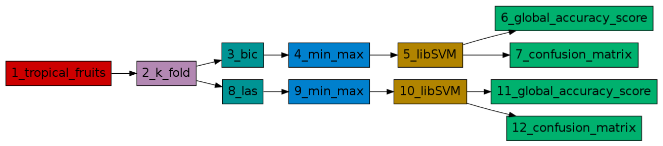


Experiment Example

Werneck, R. O.

May 20, 2014

1 Experiment Workflow



2 Protocol: K-Fold

Parameter	Value
Number of Folds	3

Table 1: Parameters of the K-Fold Method.

2.1 Global Accuracy Score

Mean	Deviation	Confidence Interval (95%)
98.48	0.00	nan

Table 2: Average, Standard Deviation and Confidence Interval of the Global Accuracy Score of Node 6

1

Mean	Deviation	Confidence Interval (95%)
74.86	0.00	nan

Table 3: Average, Standard Deviation and Confidence Interval of the Global Accuracy Score of Node 11

2.2 Confusion Matrix

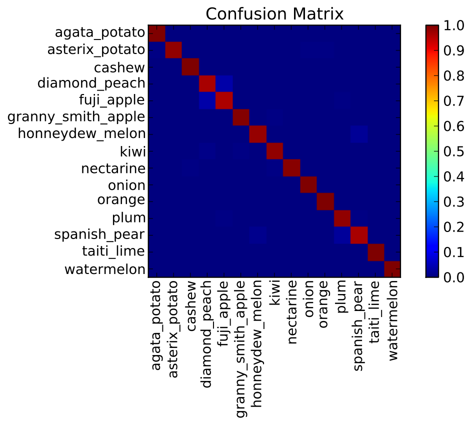


Figure 1: Average Confusion Matrix of Node 7.

	agata_potato	asterix_potato	cashew	diamond_peach	fuji_apple	granny_smith_apple	honneydew_melon	kiwi	nectarine	onion	orange	plum	spanish_pear	taiti_lime	watermelon
agata_potato	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
asterix_potato	0.55	98.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.55	0.55	0.00	0.00	0.00	0.00
cashew	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
diamond_peach	0.00	0.00	0.00	96.21	3.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fuji_apple	0.00	0.00	0.00	3.77	95.75	0.00	0.00	0.00	0.00	0.00	0.00	0.47	0.00	0.00	0.00
granny_smith_apple	0.00	0.00	0.00	0.00	0.00	99.35	0.00	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00
honneydew_melon	0.00	0.00	0.00	0.00	0.00	0.00	97.93	0.00	0.00	0.00	0.00	0.00	2.07	0.00	0.00
kiwi	0.00	0.00	0.00	1.17	0.00	0.58	0.00	98.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nectarine	0.00	0.00	0.40	0.00	0.00	0.00	0.00	0.40	99.19	0.00	0.00	0.00	0.00	0.00	0.00
onion	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
orange	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
plum	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.38	0.00	0.00	98.11	0.76	0.00	0.00
spanish_pear	0.00	0.00	0.00	0.00	0.00	0.00	1.26	0.00	0.00	0.00	0.00	2.52	96.23	0.00	0.00
taiti_lime	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
watermelon	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

Table 4: Average Confusion Matrix of Node 7.

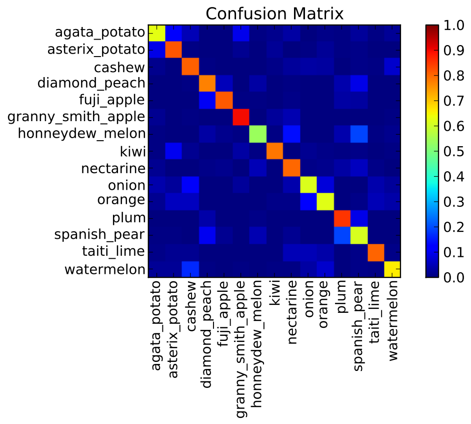


Figure 2: Average Confusion Matrix of Node 12.

	agata_potato	asterix_potato	cashew	diamond_peach	fuji_apple	granny_smith_apple	honneydew_melon	kiwi	nectarine	onion	orange	plum	spanish_pear	taiti_lime	watermelon
agata_potato	62.69	10.95	4.98	0.00	0.00	9.45	0.50	1.49	3.48	0.00	1.00	0.50	2.49	0.50	1.99
asterix_potato	9.34	81.87	0.55	0.00	0.00	1.65	0.00	2.20	0.00	0.55	2.20	0.55	1.10	0.00	0.00
cashew	1.43	0.00	80.48	0.48	0.00	0.00	0.00	0.95	2.38	3.33	2.86	0.00	0.95	0.48	6.67
diamond_peach	0.00	0.00	0.47	77.25	5.21	0.00	3.32	0.00	0.47	0.00	0.00	4.27	9.00	0.00	0.00
fuji_apple	0.00	0.00	0.00	10.38	81.60	0.47	0.47	0.00	0.94	0.00	0.00	3.30	2.83	0.00	0.00
granny_smith_apple	1.94	0.00	0.65	0.00	0.00	89.68	0.00	2.58	4.52	0.00	0.00	0.00	0.00	0.00	0.65
honneydew_melon	0.69	0.00	0.00	3.45	1.38	0.69	53.79	0.69	13.79	0.00	0.00	4.14	19.31	0.69	1.38
kiwi	0.58	9.94	1.75	0.00	0.00	3.51	0.58	78.36	0.00	1.75	1.17	0.58	1.17	0.00	0.58
nectarine	1.62	0.00	0.00	0.81	1.21	0.00	4.86	0.00	79.76	0.81	1.21	3.24	5.67	0.81	0.00
onion	4.00	2.67	10.67	0.00	0.00	2.67	0.00	0.00	4.00	61.33	8.00	0.00	0.00	4.00	2.67
orange	1.94	5.83	5.83	0.00	0.00	0.00	0.00	0.97	0.97	12.62	62.14	0.00	0.97	4.85	3.88
plum	0.00	0.00	0.00	3.79	0.38	0.00	1.14	0.00	0.38	0.00	0.00	85.23	9.09	0.00	0.00
spanish_pear	0.63	0.63	0.00	10.06	1.89	0.00	4.40	0.00	1.89	0.00	0.00	19.50	61.01	0.00	0.00
taiti_lime	0.94	1.89	1.89	0.00	0.00	0.00	0.00	0.00	4.72	4.72	3.77	0.00	0.94	80.19	0.94
watermelon	2.08	2.08	16.67	0.52	0.00	0.52	0.00	0.52	0.52	3.12	6.77	0.00	1.04	0.52	65.62

Table 5: Average Confusion Matrix of Node 12.

Appendix B

Algorithms

Algorithm 1: split_train_test(*collection* `coll`, *classes* `classes`, *train_test_method* `plugin`, *parameters* `param_values`):

```
1 Import plugin;
2 train_test ← plugin.train_test(coll, classes, param_values);
3 for each train and test set in train_test do
4     Save training set into file;
5     Save testing set into file;
6 end
7 Return train_test;
```

Algorithm 2: write_tex(*train_test_method* `plugin`, *parameters* `param_values`):

```
1 Import plugin;
2 tex_text ← plugin.write_tex(plugin, param_values);
3 Return tex_text;
```

Algorithm 3: *train_test*

Definition: Function responsible for splitting the objects of a collection into train and test sets.

Input:

- `collection`: python dictionary containing, for each object, the classes and feature vectors associated with it;
- `classes`: python dictionary containing, for each class, its objects;
- `param_values`: parameters of the train and test method (dictionary with the parameters of the XML file).

Output:

- `list_train_test`: python list containing the training set and the testing set:
  - `training set`: python list containing all objects belonging to the training set;
  - `testing set`: python list containing all objects belonging to the testing set;

```
1 Perform the split of the collection according to the plugin.
```

Algorithm 4: *write_tex*

Definition: Function responsible for writing into the result TeX file the parameters of the train and test method.

Input:

- `plugin`: name of the method being written into the TeX file;
- `param_values`: parameters of the train and test method (dictionary with the parameters of the XML file).

Output:

- `tex_string`: string in the TeX format to be written into the TeX file.

```
1 Write a LaTeX table containing the parameters of the plugin of the Train and Test
  Module.
```

Algorithm 5: **extract_features**(*collection* `coll`, *classes* `cla`, *feature_extraction_method* `plugin`, *parameters* `param_values`):

```
1  for object obj in coll do
2      if obj already extracted then
3          Load its feature vector;
4      end
5  end
6  Import plugin;
7  for object obj in coll do
8      feature ← plugin.extract(obj, param_values);
9      Save feature into file;
10 end
11 Return file path;
```

Algorithm 6: *extract*

Definition: Function responsible for performing the extraction of the feature vector

Input:

- `object`: path of the object being processed (string);
- `param_values`: parameters of the descriptor (dictionary with the parameters of the XML file).

Output:

- `object_name`: name of the object (string);
- `object_class`: class of the object (string);
- `feature`: feature vector (list of floats).

```
1 Extract feature from object;
2 feature ← fv_transform(feature);
```

Algorithm 7: *fv_transform*

Definition: Function responsible for transforming the feature vector in the standard of the framework.

Input:

- `string`: variable containing the feature vector extracted from the object (string).

Output:

- `list`: feature vector in the defined standard (list of floats).

```
1 Receive feature vector from the extract function;
2 Convert the feature vector to the defined standard of the framework;
```

Algorithm 8: normalize_features(*collection* `coll`, *train_test_sets* `train_test`, *normalization_method* `plugin`, *parameters* `param_values`):

```
1 Import plugin;
2 for Each set of training and testing do
3     Normalize the training set;
4     Save the training set into file;
5     Normalize the testing set;
6     Save the testing set into file;
7 end
8 Return train and test file paths;
```

Algorithm 9: *normalize*

Definition: Function that normalizes the feature vector according to the parameters of the normalizer method. The feature vector can be normalized according to its own values, or can depend on the values all feature vectors of the database.

Input:

- `object`: an object of the collection;
- `collection`: collection of objects;
- `objects_train`: objects in the training set;
- `param_values`: parameters of the normalizer (dictionary with the parameters of the XML file);
- `train_param`: parameters used in the training step of the normalization.

Output:

- `object_name`: name of the object (string);
- `object_class`: class of the object (string);
- `fv_norm`: feature vector normalized (list of floats);
- `train_param`: parameters of the training step of the normalization, to avoid the training phase in the normalization of the next objects.

```
1 if train_param is empty then
2     Train parameters with objects_train;
3 end
4 Normalize feature vector of object according to the train_param;
```

Algorithm 10: classification(*files with feature vectors* `fv_paths`, *train_test_sets* `train_test`, *classification_method* `plugin`, *parameters* `param_values`)

```
1 Import plugin;
2 for Each set of training and testing do
3     test_classification, model_file ← plugin.classify(fv_paths,
      training set, testing set, param_values);
4     Save test_classification in file;
5 end
6 Return test_classification;
```

Algorithm 11: *classify*

Definition: Function responsible for performing the classification of the testing set according to the model learned from the training set.

Input:

- `fv_paths`: paths of the files containing the feature vector of all objects;
- `train_set`: python list containing all objects belonging to the training set;
- `test_set`: python list containing all objects belonging to the testing set;
- `param_values`: parameters of the classification method;

Output:

- `test_set`: python list containing all objects belonging to the test set, with its paths and feature vector;
- `list_class`: python list with the ground truth of the test set;
- `list_result`: python list containing, for each index, the predictions of the classification as a list with the probability of each class;
- `model_path`: path to the saved model trained in the classification.

```
1 Build a classifier model with the training set;
2 Save model into file;
3 Predict the testing set according to the classifier model;
```

Algorithm 12: fusion(*collections* `list_collections`, *train_test_sets* `list_train_test`, *fusion_method* `plugin`, *parameters* `param_values`)

```
1 Import plugin;
2 new_collection, new_train_test ← plugin.fusion(list_collections,
  list_train_test, param_values);
3 Return new_collection, new_train_test;
```

Algorithm 13: *fusion*

Definition: Function responsible for performing the fusion of any type of module of the framework.

Input:

- `list_collections`: python list containing python dictionaries with the information of objects and feature vectors of the inputs links in the fusion module;
- `list_train_test`: python list containing the training and testing set of all inputs in the fusion module;
- `param_values`: parameters of the fusion method being applied;

Output:

- `result_collection`: python dictionary with the result of the fusion method according to each object;
- `result_train_test`: python list with the result of the fusion of training and testing sets;

```
1 Perform the fusion (train and test, feature vector, or result of classification.)
  according to the selected method;
2 Save the result of the fusion;
```

Algorithm 14: evaluation(*collection* `objects`, *train_test_set* `train_test`, *evaluation_method* `plugin`, *parameters* `param_values`)

```
1 Import plugin;
2 train, test ← train_test evaluation ← plugin.evaluate(objects,
  train_test, param_values);
3 plugin.string_file(evaluation);
4 plugin.write_tex(evaluation);
5 Return evaluation;
```

Algorithm 15: *evaluate*

Definition: Function responsible for computing the output representation according to the evaluation method.

Input:

- `objects_classification`: python dictionary containing, for each data path, the classes and the classification predictions associated to it;
- `test_set`: python list containing all data paths belonging to the test set;
- `param_values`: parameters of the output method;

Output:

- `evaluation`: result of the output method;

```
1 Performs the computations of the selected evaluation method.
```

Algorithm 16: *string_file*

Definition: Function responsible for writing the result of the output method to a file for each execution of the framework.

Input:

- `result`: result of the *evaluation* function;

Output:

- `string_result`: a string representation of the result to be written into the file;

```
1 Creates a string with the evaluation result of the experiment execution.
```

Algorithm 17: *write_tex*

```
Definition: Function responsible for processing the result of the output method
            into a format to be written into the result TeX file.
Input:
    • evaluation_path: path to the file containing the results of all executions of the
      framework for this evaluation method;
    • classes: python list with the classes of the database;
Output:
    • tex_string: string with the formated output to be written into the result TeX
      file;
1 As the evaluation_path has the result of all executions of the framework, the
  write_tex has the job to calculate the average result and write it into the TeX file.
```

Algorithm 18: recommendation(*workflows* previous_workflows, *workflow* test_workflow, *recommendation_method* plugin)

```
1 Import plugin;
2 extra ← previous_workflows;
3 for workflow train_workflow in previous_workflows do
4     list_distances ← plugin.distance(test_workflow, train_workflow,
      extra);
5 end
6 sorted_distances ← sort(list_distances);
7 Return sorted_distances;
```

Algorithm 19: *distance*

Definition: Function responsible for calculating the distance between two sequences of workflows.

Input:

- `seq1`: First sequence representing a workflow;
- `seq2`: Second sequence representing a workflow;
- `extras`: Dictionary containing any extra parameter to calculate the distance between the sequences;

Output:

- `distance`: Float value with the distance between the sequences;

```
1 According to the define function, calculate the distance between seq1 and seq2.
```

Appendix C

XML Document

```xml
<?xml version="1.0" ?>
<experiment author="Werneck R. O." date="2014-05-20" hour
   ↪ ="11:57:50" id="Experiment Example" executions="1" number
   ↪ ="1" openset="False">
        <module module="database" id="1" name="tropical_fruits"
           ↪ parameters="{}"/>
        <module module="train_test_method" id="2" name="k_fold"
           ↪ parameters="{'Number of Folds': 3}"/>
        <module module="descriptor" id="3" name="bic" parameters
           ↪ ="{'Bins': 128}"/>
        <module module="normalizer" id="4" name="min_max"
           ↪ parameters="{'Max': 1.0, 'Min': 0.0}"/>
        <module module="classifier" id="5" name="libSVM"
           ↪ parameters="{'Kernel': 'Linear', 'C': 1.0, 'degree':
           ↪  3, 'Probabilities': False, 'Cross-Validation': 3, '
           ↪ gamma': 0.0}"/>
        <module module="evaluation_measure" id="6" name="
           ↪ global_accuracy_score" parameters="{}"/>
        <module module="evaluation_measure" id="7" name="
           ↪ confusion_matrix" parameters="{}"/>
        <module module="descriptor" id="8" name="las" parameters
           ↪ ="{}"/>
        <module module="normalizer" id="9" name="min_max"
           ↪ parameters="{'Max': 1.0, 'Min': 0.0}"/>
        <module module="classifier" id="10" name="libSVM"
           ↪ parameters="{'Kernel': 'Linear', 'C': 1.0, 'degree':
           ↪  3, 'Probabilities': False, 'Cross-Validation': 3, '
           ↪ gamma': 0.0}"/>
        <module module="evaluation_measure" id="11" name="
           ↪ global_accuracy_score" parameters="{}"/>
        <module module="evaluation_measure" id="12" name="
           ↪ confusion_matrix" parameters="{}"/>
        <links>
                <link id="1">
                        <out>2</out>
                </link>
                <link id="2">
                        <in>1</in>
                        <out>3</out>
                        <out>8</out>
                </link>
                <link id="3">
                        <in>2</in>
                        <out>4</out>
                </link>
                <link id="4">
                        <out>5</out>
                        <in>3</in>
                </link>
                <link id="5">
                        <out>6</out>
                        <out>7</out>
                        <in>4</in>
                </link>
                <link id="6">
                        <in>5</in>
                </link>
                <link id="7">
                        <in>5</in>
                </link>
                <link id="8">
                        <in>2</in>
                        <out>9</out>
                </link>
                <link id="9">
                        <in>8</in>
                        <out>10</out>
                </link>
                <link id="10">
```

```
                        <out>11</out>
                        <in>9</in>
                        <out>12</out>
                </link>
                <link id="11">
                        <in>10</in>
                </link>
                <link id="12">
                        <in>10</in>
                </link>
        </links>
</experiment>
```

Listing C.1: XML of the experiment built in Figure 4.8.