Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
option Data Science

AnHALytics: a Workflow for Text Mining

Jessica DE SOUZA REINALDO

September, 2017

Research project performed at INRIA

Under the supervision of:
Didier Chassignol and Achraf Ahzar

Defended before a jury composed of:
Massih-Reza Amini
Achraf Azhar
External Expert

September 2017

Abstract

AnHALytics is a framework for text mining of technical and scientific documents coming from very large repositories. Anhalytics retrieves, extracts important data, and enriches these files using information on the article itself and other knowledge bases (e.g. Wikipedia).

The Anhalytics workflow is executed by hand, which causes some issues related to the lack of fault tolerance and of a more flexible workflow execution, as well as the difficulty to maintain and guarantee the scientific reproducibility.

In order to provide a solution to this, we have made a study on the main workflow workbenches and how they would perform with an application such as anhalytics. At the end of this study, we have selected one workbench in which we performed a simulation of an application similar to anhalytics and observed its behavior when compared to a scripting solution.

The simulation results have shown that this kind of application can be highly improved with the integration of scientific workflows.

As the simulation results have shown, scientific workflows – created at first to be used with applications from the bioscience domain – can also be used with text mining applications and present similar results, giving the application considerable improvements when compared to the classic scripting version of the workflow.

Résumé

AnHALytics est un *framework* pour le *text mining* de documents techniques et scientifiques provenant de très grandes bases des données. Anhalytics récupère, extrait des données importantes et enrichit ces documents en utilisant des informations sur l'article scientifique lui-même et d'autres bases de connaissances (ex Wikipedia).

Le workflow de anhalytics est exécuté à la main, ce qui provoque des problèmes liés au manque de tolérance aux pannes et à une exécution plus flexible des processus du *workflow*, ainsi qu'à la difficulté de maintenir et garantir la reproductibilité scientifique.

Afin de fournir une solution, une étude a été conduite sur les principaux programmes de workflow, pour évaluer comment ils fonctionneraient avec une application similaire à anhalytics. À la fin de cette étude, un de ces programmes a été sélectionné pour faire une simulation d'une application similaire à anhalytics à fin d'observer son comportement par rapport à une solution de script.

Les résultats de la simulation ont montré que ce type d'application peut être fortement amélioré grâce à l'utilisation des workflows scientifiques.

Comme les résultats de la simulation l'ont montré, les *workflows* scientifiques – créés pour être utilisés avec les applications du domaine de la biologie – peuvent également être utilisés avec des applications de *text mining* et présentent des résultats similaires, donnant à la demande des améliorations considérables par rapport à la version *scriptée* classique du flux de travail.

Acknowledgement

I would like to express my gratitude to my home university in Brazil, Federal University of Rio Grande do Sul, for giving me the opportunity of being in this double degree internship and for giving me the skills I needed to follow my studies abroad.

I also want to thank my host university, Grenoble INP - Ensimag, for providing me this fantastic and enriching experience and for presenting me this whole new domain to explore. I want to thank Inria, for having me as an intern for my master's project, and specially thank those who helped in my stay here: Didier Chassignol, Stephane Ribas, and Achraf Ahzar.

I am also thankful to the members of the jury, for taking their time to read and evaluate my work.

Contents

— 1 —
Introduction

AnHALytics is a platform whose goal is to extract information and enrich scientific documents that are obtained from platforms such as HAL [1] and ISTEX [2] by using a combination of text mining algorithms. Anhalytics is a complex application that involves several different processes that range from gathering the documents from a database to obtaining information from a knowledge base. Currently, the anhalytics workflow is executed by hand, which brings some issues to it, such as difficulty to ensure fault tolerance and reproducibility, as well as a less efficient use of resources. We have researched possible solutions to automate the anhalytics workflow in a way that it would improve its results both quantitatively and qualitatively. In order to do this, we have studied the state of the art of business and scientific workflows.

1.1 Workflows

The use of software workflows for the automation of processes started in the late 70's with the rise of business workflows. These types of workflows are aimed at processes that are very heavy on control flow and that use data only as auxiliary to the control flow.

As this type of workflow was not generic enough to all kinds of applications, in the 90's a new type of workflow came into the picture: scientific workflows. The domain of scientific workflows emerged alongside with the ever growing need of automation of processes coming from specially the life sciences. They come to fill the gap in a very specific context because the user of such workflows is a specialist in his own field but do not necessarily knows how to deal with complex scripting schemes very well.

1.2 AnHALytics

Anhalytics is a framework aimed at extracting information while at the same time enriching scientific documents in order to improve their indexation and retrieval. Anhalytics is composed of a quite complex workflow that contains text mining processes all linked together to achieve its goal.This workflow is, at the moment, being manually executed by the user. Our main goal is to find a way to automatize the execution of the anhalytics workflow, while at the same

[1] https://hal.inria.fr
[2] http://www.istex.fr

time obtaining more fault tolerance and the capability of resuming the execution of a faulty workflow.

Given the characteristics inherent to the anhalytics framework, we can do that with the help of scientific workflows, as anhalytics deals with large amounts of data, and is much more focused on the interactions and processes applied into the data than in a complex control flow.

1.3 Experiments and results

In order to check whether or not scientific workflows would bring improvements to the anhalytics workflow, we implemented two simulation scenarios to emulate the behavior of anhalytics under a scientific workflow platform and anhalytics executed by hand, as its current version.

We started by studying the most well known and utilized workbenches for scientific workflows , then compared them under several criteria which are important to the context in which anhalytics is executed.From this comparison, we have decided that the workbench that would be better suited for the anhalytics platform was [RapidMiner].

The simulation results have shown a good improvement in the version that was integrated with [RapidMiner]. The benchmarks demonstrate a performance gain, with a lower execution time.As for the qualitative results, the reproducibility is easier in the [RapidMiner] version because of the execution logs provided by default, which makes it easier to track and analyze previous executions.

1.4 Conclusion and further work

The use of scientific workflows to automate text mining applications that are comprised by a workflow of several processes can provide significant improvements specially in terms of reproducibility and efficiency achieved by a distributed execution. The field of scientific workflows still has a long way to improve and to be more adapted to the general scenario, as it is still very focused on applications targeted at the biosciences and biotechnology domains.

— 2 —
AnHALytics

AnHALytics [1] is a text mining application that aims at facilitating the access and indexing of scientific repositories by the research community.

It achieves this by normalizing the scientific documents to a standardized format, as well as adding information to scientific documents in order to enrich them and make the indexation and the retrieval easier and more efficient. This enrichment concerns specially the key terms and key categories of the document, as well as a process of disambiguation of the author and the title of the piece itself, which improves the retrieval.

Lopez [21] describes the following parameters used by AnHalytics to enrich scientific documents:

- keyword: a phrase that is usually selected by the author, and it is generally located in the header of the document.
- keyterm: the most discriminant phrase in a document. Each keyterm is associated to a relevance score, which corresponds to the probability that a reader of the document would select a given keyterm as a key phrase. Keyterms are extracted with grobid-keyterm.
- key category: obtained from the set of Wikipedia categories that are the most relevant to a document given its corpus. These parameters are categories associated with the keyterms (that were extracted with grobid-keyterm), and NERD performs the disambiguation.
- concept: an annotation produced by the nerd component. As nerd has a more semantic approach, it annotates mentions in the text with their concepts.
- key concept: the most discriminant concepts (i.e., wikipedia article) to characterize a document, given its corpus.

Lopez continues by defining a *term* as being described by three different scores: phraseness, informativeness, and keywordness. The *phraseness* measures lexical cohesion, that is, the degree to which a sequence of words can be considered a phrase. *Informativeness* measures the degree to which a term is representative of a document. This is a very standard measure in document classification, called TF-IDF (Term Frequency - Inverse Document Frequency). Lastly, the *keywordness* measures the degree to which a term is selected as a keyword. It is given simply by the frequency of a keyword in the global corpus.

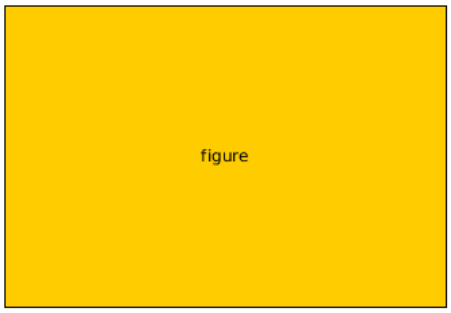
[1] https://github.com/anHALytics

The extraction of these terms and keywords is responsibility of the *Grobid* process. Grobid actually has three subprocesses that perform slightly different tasks : grobid-quantities, grobid-keyterm, and grobid-nerd.

Besides the term extraction and enrichment of the scientific documents, anhalytics is also responsible for harvesting these documents from their original repositories and storing them in local databases, and for creating a standardized TEI file with all the information added by this enrichment, which will also facilitate the indexing of these files afterwards.

Figure 2.1 shows the anhalytics execution workflow.

Figure 2.1: A grobid quantities example



The core processes of the anhalytics workflow are the grobid processing and the annotations. We will describe these processes in more detail in section 2.1.

In section 2.2 we describe how the execution of anhalytics takes place.

And at last , in section 2.3, we discuss some issues and drawbacks of the current implementation of the anhalytics execution workflow and some alternatives to solve the problem.

2.1 Main processes

2.1.1 Grobid

Grobid [2], which stands for *GeneRation Of BIbliographic Data*, is a data extraction algorithm based on conditional random fields (CRF) algorithm, which allows grobid to automatically extract and restructure raw and heterogeneous content into a standardized format.

More specifically, grobid is responsible for the extraction of keywords in scientific documents.

One of the main obstacles of the extraction of information in scientific documents is the fact that these files are not in an standardized format and some of them do not even provide information that can be easily extracted.Grobid is usually successful in extracting information from documents, but mistakes or problems are not impossible to come by. The information extracted is mostly concentrated in the header of the file, such as the title, the authors and their affiliations, and key words indicated in the document. This is the main information of a scientific document, and which allows it to be cited and indexed in library systems.

At the end of grobid processing of a document, the data that was extracted from the document is appended in the standard TEI file that was obtained from the original file.

[2]https://github.com/kermitt2/grobid

Alongside Grobid, the anhalytics platform also provides other tools to improve the information extraction and the document enrichment with annotations: NERD and grobid quantities

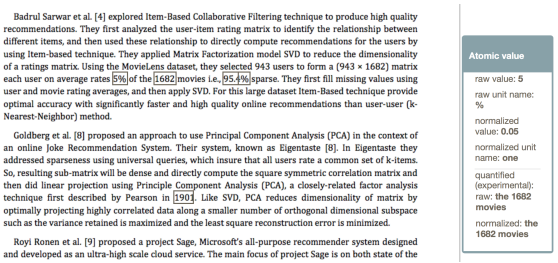
2.1.2 grobid-quantities

GROBID-quantities[3] aims at recognizing expressions of measurements in textual documents. These measurements can be later converted to SI units.

The measurements will be grouped according to the different types found in the document (e.g., years, pressure, mass) and will be shown to the user by using different colors to specify these groups.

Figure 2.2 shows an example of the result of executing grobid quantities on a document.

Figure 2.2: A grobid quantities example



2.1.3 NERD

Named Entity Recognition and Disambiguation (NERD) is the part of the anhalytics workflow that deals with entity disambiguation by using knowledge bases, more specifically, wikipedia. NERD is a process that gives semantical meaning to the terms that were extracted with grobid.

It uses wikipedia data to train the examples and then supervised machine learning algorithms to perform the disambiguation.

A browser accessible nerd console is also available, and it takes a PDF file or raw text as an input. Entities are recognized and displayed in different colors according to their types, which can be several different things, such as personal names, cities and countries, chemical elements, etc. The training is performed with wikipedia articles.

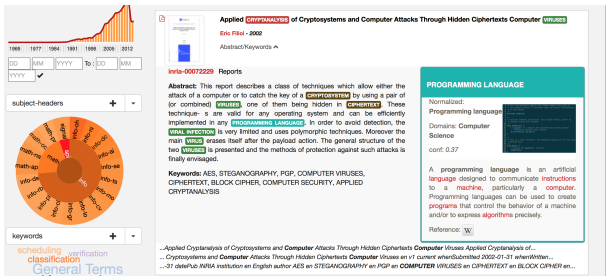
NERD performs the following tasks:

- entity recognition and disambiguation against wikidata and wikipedia in a raw text, partially annotated segment
- entity recognition and disambiguation against wikidata and wikipedia at document level, for example a pdf with layout positioning and structure aware annotations
- search query disambiguation - bellow disambiguation of the search query "concrete pump sensor" in the service test console
- weighted term vector disambiguation (a term being a phrase)
- interactive disambiguation in text editing mode

Figure 2.3 shows an example of the result of executing nerd on a document.

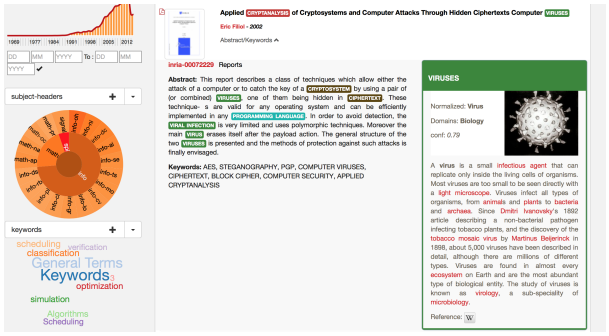
[3]https://github.com/kermitt2/grobid-quantities

Figure 2.3: A grobid NERD example



But the tool does not always get the context right. Figure 2.4 shows a case in which the meaning extracted by the knowledge bases of nerd is not the same as the one used by the author of the article.

Figure 2.4: A grobid NERD example



Besides the term extraction and enrichment of the scientific documents, anhalytics is also responsible for harvesting these documents from their original repositories and storing them in local databases, and for creating a standardized TEI file with all the information added by this enrichment, which will also facilitate the indexing of these files afterwards.

2.2 Executing AnHALytics

From the point of view of the execution, the anHALytics framework is divided in two parts: the anHALytics-core and the anHALytics-frontend. The anHALytics-core is the server part; here, documents are harvested and analyzed by grobid (grobid-keyterm and grobid-nerd). The frontend part is the client side of the framework, where a user can control the execution that happens in the backend.

A user can access the anhalytics-frontend through his web browser, where he can then search documents from main scientific databases. These documents are annotated and separated in categories that were learned by executing the AnHALytics server on the database.

2.3 Issues with the current solution

One of the main problems of the AnHALytics platform at the moment is that all the processes that we described in this chapter are manually executed, which takes a great deal of human

time and makes the platform more susceptible to errors and mistakes. Besides, by manually executing the whole AnHALytics workflow, the platform is also not capable of providing fault tolerance, error handling, or workflow resuming/restarting.

One of the possible solutions for this issue is the use of automated workflows.

Automated workflows are capable of executing the processes in an automatic fashion, which means that there is less user interference during the execution. Usually automated workflow applications will also provide some level of fault tolerance and of resuming a workflow process after a failure. The two most well known types of automated workflows are business and scientific workflows.

We will describe these two in greater detail in chapter 3.

— 3 —
Workflows

A workflow is the automation of a process, containing a well-defined sequence of tasks, control and data flows and dependencies, and participation of users. The design of a workflow is usually achieved by using a workflow system.

There were traditionally two types of workflows systems [8] – one focused on *business* processes and other for *functional* style computation of data. Later on, *scientific* workflows came as a third option that lies in between the two traditional ones, with properties for both data and control flows, which are characteristics of functional and business workflows, respectively.

The most known and broadly used type of workflow is the business workflow. Even the workflow management coalition [14], one of the main references in the domain, defined workflows in terms of business workflows, as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules." As for scientific workflows, this definition does not exactly apply, once it is not usually focused on participants and actions, but rather on activities and data flow.

One of the reasons to use workflows is that they present several advantages when compared to other common solutions. Workflows can, for instance, automate and manage processes that were being controlled by a script.

Ludascher et al. [23] cite some advantages of using workflows over scripting approaches: (i) scripting languages lack built-in provenance recording facilities, and (ii) it is difficult to use a single script to automate a process spanning multiple computer nodes, heterogeneous communication

Workflows are much more scalable than scripts, and they also provide functionalities such as fault tolerance and management on the fly of the execution.

Scaling Workflows are becoming larger and larger each day, so we are dealing with growing amounts of data. Scalability of a workflow is very important.

What needs to be scaled: (i) data, (ii) number of workflows, (iii) number of resources involved, and (iv) number of participants

Workflow requirements and desiderata According to McPhilips et al. [26], a workflow should provide:

- Well-formedness: Workflow systems should make it easy to design and validate workflows.

- Clarity: Workflow systems should make it easy to create self-explanatory workflows.
- Predictability: Purpose of products of workflow are easy to see and understand.
- Recordability: The system should make it easy to see what a workflow did when it ran.
- Reportability: Workflow systems should make it easy to see if a result makes sense scientifically.
- Reusability: Workflow systems should make it easy to design new workflows using existing workflows.
- Scientific data modeling: Workflow systems should provide data modeling schemes that let users represent their data in terms that are meaningful to them
- Automatic optimization: Workflow systems should take responsibility for optimizing workflow performance

Gil et al. [12] describe some workflow application requirements:

- Collaborations: with the ever growing computation of large amounts of data in distributed locations, these resources usually belong not to an individual, but to several, hence collaboration is necessary.
- Reproducibility: allowing scientists to reproduce the method used to obtain a result is at the core of the scientific method. Reproducibility is highly dependent on provenance information.
- Flexible environments: scientists must be able to perform both common analysis as well as be able to set up their own methods in the workflow.

Abstract and Concrete workflows Yu et al. [42] state that when it comes to resource allocation, workflows can come in two different models (also called *specification*): *abstract* and *concrete*.

In the abstract model, the workflow is described in abstract form, without specifying grid resources. In the concrete model, the workflow is described in concrete form, allocating tasks to specific resources.

An abstract workflow is a workflow in which the activities are independent of the physical resources used to execute them. As this workflow is just an abstraction and not linked to anything tangible, it is also said to be portable. Usually the resources will be allocated to a workflow on the fly, dynamically, as these resources are shared among several users and are difficult to allocate to a given process ahead of time.

The concrete workflow is the executable workflow, that is, a workflow with all the necessary resources allocated and mapped to specific grid or cluster unities to run their tasks, as well as all the necessary data movement that happens between the workflow activities

In the following sections we will get into more detail on business and scientific workflows. In section 3.3 we present some important differences between business and scientific workflows. Finally, in section 3.4 we discuss how the characteristics of these workflow types could be used in the context of the anhalytics workflow process.

3.1 Business Workflows

The field of business workflows have its origins traced back to the 70's and 80's with the office automation systems of that time. Today they are mostly used in web services for service choreography.

Business workflows are focused on control-flow and events. and aim to automate organization processes. These processes usually involve different roles that manipulate resources and are mainly control flow-driven. Data flowing through a business workflow is usually lightweight control data, so this type of workflow is more suited for dealing with complex control flow than with data flow.

As these types of workflows are very focused on control flow and participation of each role, so they provide branching primitives such as loop and iteration by default.

The user of a business workflow is usually a computer science expert, and he is the one that builds the workflow according to the needs of his client, the person or company interested in the results of the workflow execution.

The main goal of a business workflow is to check which processes can be automated in order to save resources – which could be human time, computation time, physical resources, and money.

The most famous business workflow platforms is Business Process Management (BPM). Van der Aalst et al. [36] define BPM as supporting business process processes using methods, techniques, and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information.

Business workflows are usually represented by using Petri nets. Van der Aalst et al. [35] give three reasons why it is a wise choice to select Petri-net-based workflow: (i) formal semantics (despite graphical nature), (ii) state-based instead of event-based, and (iii) abundance of analysis techniques

3.2 Scientific Workflows

Scientific workflows are workflow frameworks aimed at solving the problems of scientific research and e-science. The term *e-science* was coined to describe computationally and data intensive science. The use of workflows in this domain is growing because it is difficult to manage computationally heavy processes by hand, without automation.

According to Ludascher et al. [23] , the main goals of scientific workflows are to save "human" cycles and to save machine cycles. This means that scientific workflows need the hard work to be performed by the machines rather than by the users. A user of a scientific workflow wants to automate a scientific process and not worry much about how this process takes place.

Zhao et al. [43] state that scientific workflow systems are engaged and applied to the following aspects of scientific computations:

- Describing complex scientific procedures
- Automating data derivation processes
- High performance computing to improve throughput and performance
- Provenance management

Ludascher et al. [23] also define some scientific workflow concepts and system features:

- Integrated workflow environment
- Workflow preparation and execution support (parameter seeps to simplify multiple runs)
- "Smart reuse" is desirable
- Runtime monitoring
- Fault tolerance ("smart resume"), with checkpointing or logging, for instance.

These features are usually very similar for any kind of workflow, but are of great importance in scientific workflows.

Life cycle The scientific workflow life cycle consists of the following steps: (i) workflow design (usually reuse pre-existing workflows or refine them), (ii) workflow preparation (selection of sources and setting parameters), (iii) workflow execution (input data is consumed), and (iv) post-execution analysis (evaluate data products and interpret results)

Reproducibility According to Gil et al. [12], one of the most important aspects of scientific workflows is that they should allow the reproduction of the steps to obtain the final results to be easy. Scientific reproducibility implies that someone can follow the general methodology, relying on the same initial data, and obtain equivalent results.

3.2.1 Requirements and desiderata

Ludascher et al. [22] describe some requirements and desiderata of scientific workflows:

- Seamless access to resources and services: remote execution of jobs and access to databases
- Service composition and reuse, and workflow design: how to design actors to perform simple tasks and how to reuse these actors in other workflows
- Scalability: workflows can deal with a large number of nodes to execute a task or with large amounts of data to process or analyze
- Detached execution: workflows should be able to run in the background on a remote server and not need to be connected to the user application during all the execution time
- Reliability and fault tolerance: workflows should be made more reliable by using fault tolerance mechanisms
- User-interaction: users might be required to interact with workflows at several steps of the execution, like choosing where to run a task or to resume the execution in a certain node after a failure occurred
- Smart re-runs: allows to change a parameter in an actor without the need to re-execute the workflow from the beginning, only re-execute the nodes affected by this change. Smart re-run also comprises checkpointing, so the workflow can be executed from the last checkpoint if a failure occurs.

- Smart (semantic) links: a scientific workflow workbench should assist the design by suggesting which components fit together and datasets that might be fed to each actor. a scientific workflow system has to be able to capture some sort of semantics within the data it is dealing with

- Data provenance: an experiment run in a scientific workflow should be reproducible. so it is important to keep track of what has been done / executed in a workflow, such as its results and which actors were and in what sequence. In this context comes data provenance, that comprises information of what has been run in the workflow. Data provenance gives semantic information about data (because it can track down how it was produced and why).

3.3 Scientific vs Business Workflows

Scientific and business workflows share a lot of similarities, but they also differ on several aspects. We will talk about the points where these two workflow standards differ and where they coincide in this section. Most of the points raised in this section were based on evaluations made by Yildiz et al. [40] and Barga et al. [3].

General purpose The goals, concerns and the nature of scientific and business workflows are intrinsically different. While in business workflows the main goal is to have a set of business rules and interactions between the roles, the goal of a scientific workflow is to facilitate a scientific process by automating it. Business workflows usually focus on the roles of several actors and how they interact with each other and the activities they have to perform, while scientific workflows usually only have one "actor"(the scientist), and it is more focused on dealing with data

The processes of the scientific field are much more dataflow-driven and rely on the analysis of the data generated by the execution of the workflow (which is inherently much more complex than the data in business workflows) rather than in the overall control / coordination dependency scheme that is common in business workflow scenarios.

As it is more focused on data, it is more common that the user of a scientific workflow needs to run a given workflow multiple times, changing the input data in some of its tasks. This is the main difference of business and scientific workflows: how they deal with data.

Data flow Business workflows deal mainly with lightweight control data, so the data flow is not a very important aspect for this type of workflow.

Scientific workflows usually deal with large amounts of complex data. The data that results from each step is important (because it contains intermediate results), and should be correctly processed and stored.

Control flow Business workflows provide basic control flow methods such as sequence, branching and choice.

A scientific workflow is usually represented by a directed acyclic graph (DAG), which makes it more difficult to represent these simple primitives, as DAGs do not allow loops in the basic structure.

In scientific workflows, the control flow is modeled in a (mainly) dataflow environment. This makes dataflow and control-flow "entangled" in scientific workflows, which makes then harder to design and to understand.

Scientific workflows might need a more expressive control flow mechanism than that of business workflows in order to express both data and execution concurrency.

Goals of the User Users of business workflows want to know which tasks can be optimized in order to reduce costs, while users of scientific workflows (that is, scientists) want to know details about the scientific process that generates the final results.

Process modeling Scientific workflows usually model process using dataflow information, while business workflows usually aim at expressing a certain order of the execution of the tasks (control flow).

Dependency In business workflows, dependency is related to the partial order between two tasks in the workflow. In short, the dependency in business workflows is related to the control flow.

In scientific workflows, on the other hand, dependency is related to the data relationship (data dependency) between two tasks in the workflow, but can also depend on the control flow, because data and control flow are entangled in scientific workflows.

In conclusion, in business workflows the dependencies between activities are dependencies in the control flow, while in scientific workflows dependencies are related to control flow, data flow, or both.

Model of computation In business workflows, the model of computation is similar to imperative programming. This is a very rigid approach, that couples modeling and execution. This is not the best way to describe all processes, so approaches that allow a more flexible modeling have been proposed too. In these approaches, the modeling is a mixture between imperative programming and declarative programming.

In scientific workflows, the model of computation is dataflow oriented, so the execution of a task is data-driven. A task is "activated" and will execute only when its input data is available. This model does not impose an order of execution between the tasks, only data dependency. However, some control structures might be needed, for instance to express conditionality.

Data and control flow The main difference between business and scientific workflows lies in how they deal with data. As it is more focused on data, it is more common a scientific workflow needs to be run a given workflow multiple times, with different data in some of its tasks.

Who builds the workflow Business workflows are generally constructed by professionals in information technology, while scientific workflows are build by scientists themselves, who are usually not experts in that domain.

Provenance Information In business workflows, the goal is to know the parts of the process that can be optimized in order to reduce costs. In scientific workflows, there is much more concern about the intermediate steps, that are data resulting of a scientific process.

Comparison Table 3.1 shows the comparison between business and scientific workflow.

Table 3.1: Differences between Scientific and Business Workflows

CHARACTERISTIC	WORKFLOW	
	Business	**Scientific**
Implementation vs modeling	Process	Executability
Goals	Outcome is known since the beginning	Outcome may confirm or invalidate hypothesis
Users and roles	Responsible for distributing work to human actors	Largely automated, requiring little to no human intervention
Dataflow vs control flow	A $\rightarrow$ B represent control flow	A $\rightarrow$ B represents data flow (A produces data and B consumes it)
Reusability	Does not allow	Allows (and encourages)
Computation	Service innovation	Dataflow computations
Models of computation	Petri nets	Varied (can be job or dataflow oriented)

3.4 Discussion

In chapter 2, we discussed the AnHALytics platform and some issues with its current implementation.

We argued that a possible solution would be to use automated workflows to automatize the processes that are currently executed by hand. In this chapter, we talked about the two most well known types of automated workflows: business and scientific, pointing out their characteristics and to which applications they are more suitable.

Knowing that AnHALytics is a platform that focus mainly on the processing of *data*, and analyzing the pros and cons of each type of automated workflow, it is clear that the most appropriate type of workflow for automatizing its processes is a scientific workflow.

The field of scientific workflows has been gaining a lot of attention in the last decade, and several different workflow systems have been proposed, each with its own characteristics and aimed at different types of scientific applications. In the next chapter we will describe the most well known of these workflow systems, as well as compare them according to some important topics related to the AnHALytics framework in order to find the one that would be most adapted for its needs.

— 4 —
Scientific Workflow Management Systems

Workflow management systems aim to automate the execution of processes [40]. The automation of a process involves the scheduling, control and monitoring of the tasks that comprise it. A workflow system provides means to model, re-engineer and automate the execution of processes.

In chapter 3, we described the standards for business and scientific workflows and their differences and similarities. In this section, we will present some of the most well known systems for creating and managing scientific workflows. For each of them, we will provide a brief description and at the end of the chapter we included a section in which they are compared based on some characteristics that are fundamental to our application.

Yildiz et al. [41] stated that "the main advantage of workflow management systems is the modeling capabilities that allow users, who may not be specialized in programming, to implement their processes using the comprehensible design primitives such as activities, dependencies, and control structures."

Our goal is to find the workflow system that is more adapted to our application needs.

4.1 Kepler

According to Zhao et al. [43], Kepler [1] is a scientific workflow framework based on Ptolemy II, which is a framework to design heterogeneous and concurrent systems.

Kepler uses *cinema* as a metaphor for how it sees the components of a workflow: each workflow has a director and several actors. Directors specify when an event occurs while actors specify what occurs. There are different types of directors, each of them more ??specified to work in a different scenario.

Curcin et al. [8] classify Kepler's core directors in four different types:

- SDF (synchronous data flow): it has fixed rates for token production and consumption. The order of execution is statically determined from the model.

- PN (process network): it is a derestricted version of SDF, where the actor is invoked only when the data arrives, although there is no need to have all data to invoke the actor. This leads to a more dynamic environment, where actors can be executing in parallel.

[1] https://kepler-project.org/

- CT (continuous time): it introduces the notion of timestamps on tokens and it is usually described in terms of differential equations to approximate functions and schedule the executions.

- DE (discrete event): it also uses timestamps on the tokens to measure average waiting times.

Kepler does not have a functional composition – it is the director that controls the order in which the actors will execute. Each director can be classified as strict, loose or loosest, based on the semantics. Embedding in Kepler are allowed only when an inner director is at least as strict as an outer director.

In order to give a semantic meaning to what an actor does, the actor has to be directed. It is the director that controls, for instance, how many times an actor is to be executed or if it can be executed in parallel with other actors.

Some highlights of Kepler: (i) web service extensions, (ii) grid and other extensions, and (iii) actor-oriented modeling

4.2 Pegasus

Pegasus (Planning for Execution in Grids) [2], as its name implies , is a tool designed to map workflows onto grid resources that can execute them.

Pegasus does not provide a GUI, so workflows must be created by describing their scheme in the DAX (Directed Acyclic Graph XML) language. Pegasus provides libraries to DAX generators in Perl, Python, and Java. This DAX description is portable and it does not depend on local files or information to execute. The workflow created with the DAX language is only an abstract form of the workflow.

According to Deelman et al. [9], to map an abstract workflow into a concrete (executable) one, three things have to be found: resources to execute the tasks, data used in the workflow, and the necessary software.

Pegasus is also capable of performing workflow reduction when mapping an abstract workflow into its concrete form. For instance, if a file is the input to a task and it has been already computed, Pegasus will use this file and not recompute the task that created it (and for which the file is an output).

The concrete workflow produced by Pegasus can be executed by DAGMan. It generates a submit file that contains information about the allocation of tasks to resources and the order in which the tasks should be executed.

According to Ludascher et al. [22], Pegasus is (at that moment) semi-dynamic, because abstract workflows are mapped to concrete workflows only when they are given to Pegasus.

Pegasus uses three different information catalogs:

- site catalog: description of the sites where each workflow job is going to be executed

- transformation catalog: description of the executables used by the workflow (location, operating system, etc)

- replica catalog: description of the location of each input file used in the workflow

[2] https://pegasus.isi.edu/

4.3 Taverna

According to Wolsterncroft et al. [37], Taverna is designed to combine distributed webservices and local tools into complex analysis pipelines, which can be executed in clusters, grids, etc or even on the local machine.

Taverna is composed [8] by the Taverna workbench, the SCUFL language and the FreeFluo enactment engine.

The workbench is composed of the graphical user interface and several components that the user can use (by dragging and dropping into the workflow canvas) to build his workflow. The workflow a user builds is going to be stored in the scufl representation language. The workflow is represented as a DAG. When the user wants to run workflows he has build, they will be run with the Freefluo engine. SCUFL, the language in which Taverna workflows are represented has some limitations, though. Scufl is a dataflow language with only some control flow primitives. The only control structure Taverna has is a conditional construct, similar to the case construction. Loops can be achieved only in a limited form.

Usually Taverna workflows are composed from a mixture of distributed web services, local scripts and other services (for instance, R scripts). The biggest advantage of using distributed systems is that all the computation overhead of the execution happens in a remote location, and not the local machine, so there is no need to install any tools locally. This is specially evident for the execution in the cloud, where the cloud server can easily provide the user with all the tools he needs to execute his workflow from the start.

4.4 Triana

Cursin et al. [8] describe Triana [3] as being a visual workflow-based problem solving environment developed at Cardiff university.

A workflow component in Triana is called *unit*, and units can connect with each other by directed cables.

Triana is a dataflow system, but can provide support for control flow by using some special messages that can trigger control between units.There are also special nodes for branching and looping, and they can be combined with other functional units to build more sophisticated forms of control flow.

One of the main disadvantages of Triana is that it has not had any recent updates. The GitHub repository that hosts the Triana development code was last updated almost three years ago, with the last stable version launched more than four years ago.

4.5 RapidMiner

RapidMiner is a data science / data mining platform. Its workbench is called RapidMiner Studio, and it offers the users a very modern GUI in which they can build their workflows. It provides several tools to work with the data and to create workflows that focus on the relationships and the transference of data.

[3] http://www.trianacode.org/

Jungermann et al. [15] described the information extraction in RapidMiner [4] as being a plugin that converts documents containing natural language texts to machine-readable form in order to extract interesting information and relationships between them.

The process-view in RapidMiner presents a modular view of the experiment. it usually consists of four stages: (i) input stage, (ii) preprocessing stage, (iii) learning stage, and (iv) evaluation stage.

Radoop

One of the issues with Hadoop is that it is not very user-friendly because it lacks a graphical user interface.

Prekopcsak et al. [31] developed Radoop, an extension to RapidMiner which allows it to work with Hadoop.

Radoop aims at dealing with the problem of big data (and for that, distributed computation) while hiding the complexity of data analytics through a high level framework such as RapidMiner.

4.6 Knime

Knime [5] [4], also known as Konstanz Information Miner, is an environment for the interactive execution of a data pipeline. It allows the user to create (and visualize) the analysis flow by building blocks connected through pipes that represent the data flow between those blocks.

The user can model and visualize workflows. A workflow is consists of nodes and connections between these nodes; the nodes process data and the connections carry the data among the nodes.

A node in Knime processes all its data before forwarding it. This means that the process can be stopped and easily resumed and also that new nodes can be added to the workflow scheme after the node in execution and it will not be necessary to rerun the previous nodes of the workflow again.

Workflows in Knime are directed acyclic graphs (DAGs). Besides the nodes that compose the workflow *per se*, there is also the workflow manager. The workflow manager holds the power to allow (or not) the addition of new nodes and connections between nodes in the workflow schema.

Besides the built-in nodes provided by Knime, a user can also create his own workflow nodes by using Knime plugins. New nodes are created by extending one of the abstract classes for nodes, so the user must know how to code in order to build new nodes from scratch.

A workflow in Knime is represented by a DAG, and for this reason, loops cannot be represented. To overcome this problem, Knime provides two special nodes: loop start and loop-end. These nodes, unlike regular nodes, are not reset when the loop executes and they can exchange information with each other.

[4] https://rapidminer.com/

[5] https://www.knime.org/

4.7 Discussion / Comparison

In this section we will present a comparison study between the scientific workflow systems previously described in this chapter. This comparison takes into account some important criteria to the execution of an application such as anhalytics, mainly focusing on a distributed scenario.

4.7.1 Workflow management and deployment

Kepler Kepler has a monitor manager that, for each element of the workflow, allows the user to indicate the style and the events to be monitored [17]. This is, however, a experimental module and might not work perfectly.

Pegasus In Pegasus, the management of the workflow being run is done with logs, which can be of three different types [34]:

- Pegasus mapper log
- DAGman log
- Job logs (aka kickstart records)

DAGman writes a log file in near real time during execution. This log file contains the status of each job in the workflow, as well as the scripts associated with it.

Taverna Taverna workflows can be annotate (by the user or configurable to be automatically annotated). However, Taverna does not capture provenance of a workflow definition, and assumes that the scientist is managing this through versioning (names, git, etc). Each time a workflow is changed, an internal identifier it has also changing, which allows the user to keep track of these changes and workflows that have the same ancestry [32].

Triana Triana can be used alongside stampede to provide better management and deployment tools. Besides that, Triana does not have any kind of logging or management tools or information [34].

Knime Knime does not have a logging scheme *per se*, but it has some tools that can store some intermediate execution information about the nodes of a workflow.

RapidMiner RapidMiner has a log table which can store all kinds of information, such as parameters, execution time, etc. The information stored in the log table can also be analyzed in form of graphs in RapidMiner's result workspace [19].

4.7.2 Workflow execution

Kepler Kepler supports multiple levels of distributed execution, on the workflow, subworkflow and atomic actor levels. It supports master-slave, map/reduce, and grid distributed execution, among others [30].

Pegasus Pegasus can run, from the point-of-view of the user, workflows across multiple heterogeneous resources distributed in a wide area, while at the same time shielding the user from the grid specific details [10]. Both data storage and the workflow execution itself can be distributed.

Taverna Taverna workflows can be executed in distributed web services, using services on a grid or on a cloud [37].

Triana Triana is mainly focused on executing workflows on the grid or as web services. It has two distribution policies: parallel and pipeline, as well as two modes of execution: dynamic and static [6]. In the dynamic approach, the workflows are sent to services that can execute any subworkflow and communicate with other Triana services they are connected to. In the static approach, a user can chose to launch a group unit as a specific remote service, so Triana units or groups may be deployed as Web services.

Knime Knime has no free option for distributed execution. It has three main frameworks, Knime Cluster Executor, Knime Server, and Knime Spark Executor, but they are all commercial products [38].

RapidMiner RapidMiner has an extension called Radoop which allows it to be integrated with the Hadoop framework more easily. The main method of distributed execution in RapidMiner is by using Radoop to run tasks on Hadoop, which is the most well known platform for distributed execution focused on the map/reduce model, very efficient in distributing the execution of big data applications [31].

4.7.3 Deployment of a workflow while in execution

Kepler Kepler has an execution monitoring that keeps track of all the activities performed in a workflow as well as the data it is using [17].

Pegasus Pegasus has a monitoring framework called NetLogger that runs under Stampede (Synthesized Tools for Archiving, Monitoring Performance and Enhanced DEbugging). The goal of Stampede is to enable real-time debugging and performance and behavior analysis of workflows [34].

Taverna During a workflow execution, Taverna displays status information. For each Scufl processor, the last event is displayed alongside with detailed information (if any), which can include the progress through an iteration and retry information. This status information also provides more details about intermediate inputs and outputs [28].

Taverna also has a workflow monitor web application that performs periodic monitoring of workflow activities and provides information about their current status [39].

Triana Triana can be extended to use Stampede Monitoring, just like Pegasus. When integrating Triana and Stampede, each task within a task graph is run locally [34].

Knime Knime has a monitoring tool called NodeMonitor. Using the debug feature of NodeMonitor allows the user to check the execution times for each node of a workflow.

RapidMiner Logging and Monitoring are utilities available for RapidMiner + Radoop. Monitoring contains information on the cluster's status, health, version, etc. It provides detailed information on the applications and jobs running in the cluster, as well as logs and information about the MapReduce jobs. These utilities are also available whenever RapidMiner is used with the spark web UI. If the user wants to execute jobs on the cloud, RapidMiner also provides a tool called CloudMonitor that monitors the jobs executing on the cloud [20].

4.7.4 Fault tolerance and error handling

Kepler Kepler has some specialized actors to deal with errors and exceptions, such as the so called "exception catching actors". Kepler also has the rerun capability based on the provenance of the information collected by the system [1].

Pegasus Pegasus provides the following fault tolerance mechanisms [11]:

- job retry: jobs and data transfers are automatically retried in case of failures.
- data movement reliability
- failed workflow recovery / rescue DAGs: the workflow will restart from the point of failure
- workflow re-planning: in case of workflow failures, users can re-plan the workflow and
- continue the computation at a different resource

Taverna Taverna provides fault tolerance mechanisms such as dynamic service substitution and retry [28].

Triana Triana provides fault tolerance (prevention and recover capabilities) at task level, workflow level and user level. It can also easily detect faults on hardware level, but not so much on the OS level [7].

Knime Knime provides try/catch nodes and basic error handling. Its mechanisms for fault tolerance are not as sophisticated as the ones of some other workflow management systems.

RapidMiner Using only RapidMiner, in order to restart from an intermediate step, the user needs to save the data to the disk, which can be done through the GUI. Using Radoop alongside RapidMiner allows to use the fault tolerance provided by Hadoop [31].

4.7.5 Reproducibility

Most workflow management systems provide reproducibility through data provenance. To obtain data provenance, these systems log each step of the workflow with great detail, in order to make that workflow easily reproducible by anyone, thus asserting scientific reproducibility.

For scientific workflows, provenance can be of three different types [32]:

1. Provenance of the workflow definition
2. Provenance of a workflow run
3. Provenance of data

Each workflow system might capture a different set of these provenance types.

Kepler Kepler has an add-on module that allows the user to record a workflow execution history. It is simply a button in the GUI which the user can activate and deactivate as he sees fit. Whenever a user changes some parameters of an actor, the workflow might be able to perform a smart re-run, which takes data dependencies into account and only execute the parts of the workflow that were affected by the changes [27].

Pegasus Pegasus keeps track of what has been done (provenance) including the locations of data used and produced, and which software was used with which parameters. By default, all jobs in Pegasus are launched using the Kickstart wrapper that captures runtime provenance of the job and helps in debugging. Provenance data is collected in a database, and the data can be queried with Pegasus specific tools or directly using SQL [29].

Taverna The Taverna Engine (workbench + command line) enacts the workflows and collects the provenance of workflow runs, including individual processor iterations and their inputs and outputs. It is also able to provide some level of provenance of workflow data through some tools in myGrid, such as annotations [16].

Triana Triana provides data provenance through workflow annotation, which can happen manually (user annotates the workflow) or automatically. Provenance data recorded includes: date and time composed, date and time of the execution, details of the resource on which a service was executed, and intermediate data products generated [24].

Knime At any point of the workflow, the user can inspect data, model, etc., which provides intermediate results of the workflow. Knime also provides logging which is divided in several levels such as DEBUG, INFO, WARN and ERROR. The verbosity of this log can be controlled by the user beforehand [18].

RapidMiner In RapidMiner, there is little built-in support for recording and accessing the information of how data outputs have been produced by different versions of a flow. Most logging is focused on logging the Radoop extension in order to access and monitor the cluster's web interfaces [13] . RapidMiner also provides a log operator that can be used to store the values a user has chosen.

4.7.6 Adaptability

Most workflow management systems allow the execution both locally and on distributed environments (be it a cluster, a grid, or a cloud). Generally, this is not transparent to the user; the user needs to state his or her preference for where the workflow is going to be run.

Kepler Extensions allow Kepler to be executed in a distributed environment, such as clusters, grids, and clouds. Kepler is not targeted to a particular distributed execution requirement. The distributed execution in Kepler can happen at the workflow level, subworkflow level, director level or actor level. Because of these wide variations, this is not transparent to the user.

Pegasus In Pegasus, user created workflows can be easily run in different environments without alteration. Pegasus currently runs workflows on top of Condor, grid infrastructures, and clusters. The same workflow can run on a single system or across a heterogeneous set of resources.

Taverna In Taverna, the distributed execution happens in the form of webservices. A workflow (or subworkflow) is seem as a service. Taverna also provides local and distributed pipeline execution. In order to use Taverna's distributed resources, the user needs to be using Taverna Server.

Triana The user needs to select the resources for execution. A user will, for example, select a group of tasks he wants to execute in parallel. The main unit of Triana distributed execution are those group tasks; data is distributed accordingly.

Knime Knime only allows distributed execution with the help of a paid extension, which means that the open source and free version does not allow / adapt to distributed environments, only single user machine.

RapidMiner RapidMiner has the Radoop extension, so the behavior for execution (be it distributed or not) will be similar to using Hadoop while using this extension. The user has to describe the available machines in a file, and any tasks that can be distributed are going to be allocated to each of these machines. Without this extension, RapidMiner can only execute the workflows locally.

4.7.7 Challenge when comparing to scripting solutions

For all workflow management systems, there is a more difficult step at the beginning, which involves having to translate everything that was coded and managed with scripts into the different workflow language or environment each of these tools utilize. After doing this, it is actually easier to develop new features and manage the execution (which comprises also data management and error handling) while using a workflow management platform than while using scripts.

4.7.8 Scalability

All of the workflow systems we studied are said to scale well in distributed environments, being able to deal with up to thousands of workflow elements.

Kepler Kepler has specialized actors to deal with grid architectures / environments and can also be integrated with Hadoop and Spark [2].

Pegasus Pegasus can easily scale well to both the size of the workflow and the resources. Pegasus can run workflows ranging from just a few computational tasks up to one million tasks. The number of resources involved in executing a workflow can scale as needed without any impediments to performance [29]. Among the workflow systems we have considered, Pegasus is the best for distributed execution.

Taverna Taverna was build to be scalable through the SCAPE project, aimed at building a scalable platform for planning and executing workflows. The workflows are deployed on a large scale (using Hadoop) and executed over large, distributed and heterogeneous objects [33].

Triana Triana workflows can be executed on the grid and in P2P by selecting resources for execution, so it scales well for the number of nodes. Triana can also be executed on the cloud if an appropriate extension is used in the workflow [24].

Knime The Knime distributed execution module is paid, and the system does not scale well with local execution (limited ram, processors, and disk space).

RapidMiner RapidMiner uses Radoop, and extension that allows users to use Hadoop inside of RapidMiner very easily; so, in theory, it should scale as well as Hadoop [24]. However, the distributed version of RapidMiner is paid, so in the free version only one thread is used to run the workflow.

4.8 Discussion

UPDATE!!

In chapter 2, we discussed the AnHALytics platform and how automatizing its processes could improve the overall performance of the framework. In chapter 3 we presented the two most used and well known types of automated workflow types, and we argued that scientific workflows would be a better fit for the automation of anhalytics, according to its characteristics and general needs. In this chapter, we presented some of the most well known and widely used scientific workflow platforms and compared them using some criteria that is of great importance in the context of the anhalytics application.Among the workflow systems described herewe believe that the anhalytics framework would work better with [RapidMiner].

Taverna, and Kepler could also be good fits for anhalytics, but they are focused on bio-science applications. Implementing data mining algorithms in these workbenches would be a lot more expensive in terms of time and work than with RapidMiner. Besides the domain, RapidMiner is also the workbench with the best documentation among all of those we have studied and the one that is more often updated.

[RapidMiner] provides operators that implement data mining algorithms, fault tolerance, process logging, and easy execution management. However, only its core is free and open source. Distributed execution is not available in the free version, so the criterion of scalability should not be the first priority when using RapidMiner. Even with this drawback, we consider that it is a good fit for anhalytics.

In the next chapter we discuss how we can use this scientific workflow system to improve the anhlaytics workflow process.

— 5 —
Implementation and Evaluation

solution to solve the problem previously stated
DRAFT

5.1 Some remarks on the proposed solution

At first we planned on implementing the anhalytics framework over one of the workflow workbenches, but this was proven to be more difficult than we expected, mostly because virtually all these workbenches lack documentation on the integration between the workflow system and an existing application. We have spent some time trying to find a solution to this, but to no avail.

Because of the nature of the scientific workflow systems we have studied so far, implementing a large application such as anhalytics over one of these workbenches is going to require a lot of technical work. First of all, apart from RapidMiner, all the other systems are aimed at the biosciences domain, which means that most of their operators/actors operators/actors available off-the-shelf are exclusive to this domain. In order to implement a text mining application over such a system would mean that all algorithms would have to be transformed in Java actors or operators, or used as a external scripting actor. In such a scenario, the whole project would have to be thought over ?? from the beginning, needing a lot of refactoring and reorganization of the code.

In RapidMiner, several data mining algorithms and tools are available either on the standard version or in the Text Mining extension, which is free and easily installed. But using RapidMiner would also imply?? the need of refactoring the code, with the advantage that in this case a good part of the tools needed are available as default operators, so they would not have to be (re-)implemented.

As time was an important factor during an internship, we decided it would be best to implement a simulation using two different scenarios instead. These scenarios aim to reflect the behavior of the execution of the anhalytics platform in its current form (with all the processes being executed by hand) and the execution of anhalytics being managed by a scientific workflow system.

5.2 Proposed solution

As stated in the previous section, the process of integrating anhalytics with scientific workflow workbench goes beyond the scope of this internship, as it deals with a lot of technicalities

and would require more time to redesign the system. Two scenarios are proposed in order to simulate the behavior of the anhalytics platform in its current state and with a scientific workflow.

These two implementation are presented in more detail in the next subsection.

5.2.1 Simulation

The simulation scenario is composed by a series of tasks that involve web crawling and the application of text mining algorithms, both important steps in the anhalytics workflow.

The following tasks were implemented:

- Read an excel file containing URLs to TEI documents from the HAL platform.
- Web crawling to obtain the content of these documents.
- Extraction of the English abstract.
- Tokenization and removal of stop words in these documents.
- Cosine similarity and tf-idf metrics calculated for the English abstract of these documents.

The scenario that simulates the current state of anhalytics was implemented in Python, while the scientific workflow scenario was implemented in RapidMiner.

For each of the documents obtained from the HAL platform, we extracted its English abstracts and calculated the cosine similarity and the tf-idf measures when compared to the other documents.

In the scripting version, we needed to use several Python libraries to implement the algorithms we needed, such as libraries to exclude the stop words from the original text and to read and parse the xls and xml files. With RapidMiner, all the functions we needed were either available off the shelf or implemented in the text mining extension, which is free and can be easily installed from the extension manager.

The simulation schemes are presented in the next subsections.

Script

The scripting scenario aims to simulate the behavior of the current implementation of the anhalytics platform.

Figure 5.1 shows an scheme with the simulation scenario implemented with Python scripts.

Python is one of the most used programming languages in data mining applications, so it has libraries that implement several data mining algorithms.To implement this scenario, we used the library *nltk* for the tf-idf algorithm, *urllib* for gathering documents from their URLs, and *xlrd* to read the xls file containing a list of the URLs.

Scientific workflow

The scientific workflow scenario was implemented in RapidMiner. As RapidMiner is focused on data science applications, it provides several operators that implement text mining algorithms. Consequently, in this scenario we only needed to use RapidMiner's operators, no external scripts were needed.

Figure 5.1: Simulation scenario: scripting

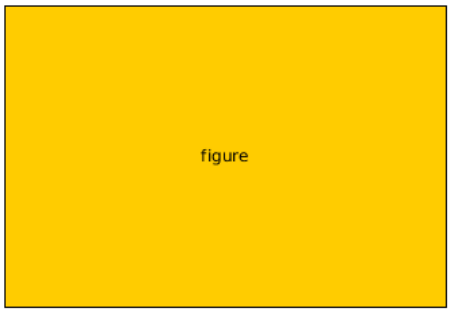
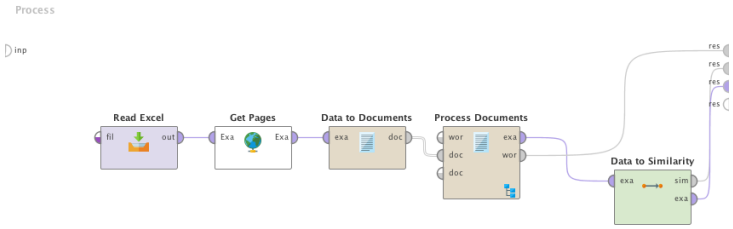


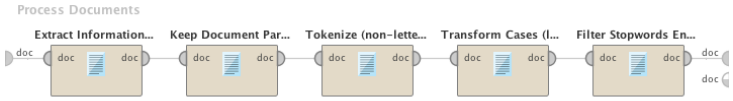
Figure 5.2 shows the scheme implemented in RapidMiner.

Figure 5.2: Simulation scenario: scientific workflow



The task "Process Documents" is a subworkflow. Its complete scheme is shown in figure 5.3

Figure 5.3: Inside the "Process Documents" operator



— 6 —
Evaluation

evaluation or validation of the proposed solution (or guides to it)

DRAFT

The goal of both scenarios is to simulate a very simplified version of the behavior of the anhalytics process.

The two simulation scenarios are evaluated under the same criteria, some of which was already discussed in section 4.7. We used the following criteria to evaluate the solutions:

- Reproducibility
- Fault tolerance
- Execution time
- Execution management (resuming, pausing)

All these criteria, with the exception of execution time, can be considered as qualitative. For this reason, we will discuss how they can be achieved in each scenario.

6.1 Reproducibility

Cacioppo et al. [5] argue that *reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator*.

Marcus [25] links reproducibility with the credibility in scientific experiments, and states that *without credibility, others can't/won't build on our work, and as a result, the pace of scientific advance is slowed*.

Reproducibility is one of the most important criteria we need to evaluate, as it is at the very basis of the scientific process.

In the scripting scenario, any means to achieve reproducibility have to be manually implemented, as none is available by default. Most of modern programming languages have libraries that allow the user to create their own logs for an execution, but still it would be the user's choice to decide what to put into the log.

In the scientific workflow scenario, there are tools that can be used to achieve reproducibility, such as detailed logging and validation of a process workflow. The workflows generated are also portable, which makes it easier for a different observer to analyze the results obtained.

It is also important to note that RapidMiner provides a rich view of the results after the execution is finished, including statistics and charts to visualize the results. This is also a factor that is related to reproducibility, because it is much more simple to analyze and compare the results of the execution.

6.2 Fault Tolerance

Fault tolerance is another important criterion to be evaluated, as it ensures that the work previously executed will not be lost if a technical problem happens in the middle of the execution process.

The scripting solution does not provide fault tolerance in itself, but could be executed in an environment that guarantees fault tolerance.

It is on the basis of scientific workflows to provide fault tolerance. In the case of RapidMiner it is provided by its Radoop extension. Radoop is a Hadoop implementation for RapidMiner, so it provides fault tolerance. In the implemented scenario we do not use Radoop, so the resources for fault tolerance were not being used.

6.3 Execution Time

One of our goals with the automation of anhalytics would be to reduce the execution time – mainly by optimizing the number of threads used to execute the application. As anhalytics seem to be an application very prone to distribution, as each document is individually treated at each time, we expect the execution time to be reduced with the use of a scientific workflow controlling its execution.

The table 6.1 shows the results for execution time for the simulation scenarios.

Table 6.1: Execution time in the simulation scenarios

Task	SCENARIO	
	Script	**Scientific workflow**
Text harvesting	0	0
Text mining	0	0
Transfer to database	0	0

6.4 Execution Management

The execution management, which comprises specially the resuming and pausing of the execution, is also an important criterion we need to evaluate.

The scripting solution does not provide any form of execution management, because programming languages do not provide this resource. The closest thing to an execution management that this solution can provide is adding breakpoints in some parts of the code. The

execution stops at each breakpoint and the user can check the state of the variables, and then manually continue the execution until the next breakpoint.

In the scientific workflow scenario we also have the option of using breakpoints for debugging, but the platform allows a user to pause/resume the execution of the workflow by pressing a button.

6.5 Discussion

According to the evaluation results presented in this chapter,

Table 6.2 shows an overview of the points discussed in the previous subsections.

Table 6.2: Overview of criteria in the simulation scenarios

Criterion	SCENARIO	
	Script	**Scientific workflow**
Reproducibility	It has to be manually implemented	Easier to achieve with the automatic logs and portability of solution
Fault Tolerance	It has to be manually implemented	Execution can be restarted from last point
Execution time	0	0
Execution management	Does not provide it	Provides breakpoints and pausing/resuming of the workflow

— 7 —
Conclusion

DRAFT

We presented an overview of the field of workflows to automate processes, as well as discussed the differences between the two most important types of workflows – business and scientific.

We presented the anhalytics platform and explained the issues with the current solution. We discussedhow we could use automated workflows to improve anhalytics, and performed a thorough study on which type of workflow and which workbench would be more appropriate to automate anhalytics, taking into consideration the inherent characteristics of anhalytics and our priorities in terms of quantitative and qualitative results of this implementation.

Then we presented a simulation scenario which aims to represent the anhalytics process workflow, and we implemented it in two ways: one was integrated into the scientific workflow, and the other was manually executed (scripting).

An evaluation of the two simulation scenarios was presented, taking into account the most important criteria concerning the execution of the anhalytics workflow. We discussed the results obtained with the evaluation, which have shown a considerable advantage in the scenario that uses the scientific workflow approach.

The simulation results have shown that the scenario implemented on RapidMiner [talk about results].

For our purposes one of the most important criterion is the reproducibility, which is hard to guarantee in the current anhalytics version, but with the use of scientific workflows it can be achieved much more easily.

Taking these results in consideration and extrapolating it to the real life implementation, we strongly believe that the automation of anHALytics would also show improvements in the criteria under which it was evaluated, specially reproducibility and easier execution management.

The natural next step within the context of this research project is the implementation of the studied concepts directly on the anhalytics platform – that is, to integrate anhalytics with the chosen scientific workflow workbench, Kepler. Although the implementation was proven to be more difficult than we expected, we believe that the results show that the additional effort pays off, in delivering an application that will be more reliable and efficient in the long run. ??

Concerning scientific workflows and the remarkable growth in data science research in the last few years, we believe that new platforms aimed at these types of applications are to be expected in the near future. RapidMiner, which is the only workbench we studied that provided tools off the shelf for implementing machine learning algorithms, seems to indicate that this could be a trend in the next years, which will strongly facilitate the development and

deployment of data science applications.

The results obtained with this study are not limited to the anhalytics platform or even to text mining applications. Our results demonstrate that using scientific workflows to organize complex process flows can improve the execution in several aspects and it is well worth the additional work.

Bibliography

[1] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In *International Provenance and Annotation Workshop*, pages 118–132. Springer, 2006.

[2] 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.

[3] Roger Barga and Dennis Gannon. Scientific versus business workflows. In *Workflows for e-Science*, pages 9–16. Springer, 2007.

[4] Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime-the konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1):26–31, 2009.

[5] John T Cacioppo, Robert M Kaplan, Jon A Krosnick, James L Olds, and Heather Dean. Social, behavioral, and economic sciences perspectives on robust and reliable science. 2015.

[6] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming scientific and distributed workflow with triana services. *Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, 2006.

[7] Alexandru Costan, Corina Stratan, Eliana-Dina Tirsa, Mugurel Ionut Andreica, and Valentin Cristea. Towards a grid platform for scientific workflows management. *arXiv preprint arXiv:0910.0626*, 2009.

[8] Vasa Curcin and Moustafa Ghanem. Scientific workflow systems-can one size fit all? In *Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International*, pages 1–9. IEEE, 2008.

[9] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In *undefined*, pages 11–20. Springer, 2004.

[10] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[11] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, et al. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015.

[12] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *Computer*, 40(12), 2007.

[13] Fenno F Terry Heath III and Richard Hull. Analytics process management: A new challenge for the bpm community. In *Business Process Management Workshops*, pages 175–185, 2014.

[14] David Hollingsworth and UK Hampshire. Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19, 1995.

[15] Felix Jungermann. Information extraction with rapidminer. In *Proceedings of the GSCL SymposiumâSprachtechnologie und eHumanities*, pages 50–61. Citeseer, 2009.

[16] Simon Jupp, James Eales, Simon Fischer, Sebastian Land, Rishi Ramgolam, Alan Williams, and Robert Stevens. Combining rapidminer operators with bioinformatics services. a powerful combination. In *RapidMiner Community Meeting and Conference*. Shaker, 2011.

[17] Kepler Execution Monitor. `https://kepler-project.org/developers/incubation/kepler-execution-monitoring/archive/kepler-execution-monitoring`. Accessed: 2017-07-05.

[18] KNIME Quickstart Guide. `http://www.mi.fu-berlin.de/wiki/pub/ABI/KnimeSec/KNIME_quickstart.pdf`. Accessed: 2017-07-05.

[19] Log (RapidMiner Studio Core) . `https://docs.rapidminer.com/studio/operators/utility/logging/log.html`. Accessed: 2017-07-05.

[20] Logging and Monitoring. `https://docs.rapidminer.com/radoop/troubleshooting/logging-and-monitoring.html`. Accessed: 2017-07-05.

[21] Patrice Lopez and Laurent Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 248–251. Association for Computational Linguistics, 2010.

[22] 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.

[23] Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. Scientific workflows: Business as usual? In *International Conference on Business Process Management*, pages 31–47. Springer, 2009.

[24] Shalil Majithia, Matthew Shields, Ian Taylor, and Ian Wang. Triana: A graphical web service composition and execution toolkit. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 514–521. IEEE, 2004.

[25] Emilie Marcus. Credibility and reproducibility. *Structure*, 23(1):1–2, 2015.

[26] Timothy McPhillips, Shawn Bowers, Daniel Zinn, and Bertram Ludäscher. Scientific workflow design for mere mortals. *Future Generation Computer Systems*, 25(5):541–551, 2009.

[27] Luc Moreau. *Provenance and Annotation of Data: International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*, volume 4145. Springer Science & Business Media, 2006.

[28] Tom Oinn, Mark Greenwood, Matthew Addis, M Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, et al. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[29] Pegasus - Workflow Management System. `https://pegasus.isi.edu/overview/`. Accessed: 2017-07-05.

[30] Marcin Płóciennik, Tomasz Żok, Ilkay Altintas, Jianwu Wang, Daniel Crawl, David Abramson, Frederic Imbeaux, Bernard Guillerminet, Marcos Lopez-Caniego, Isabel Campos Plasencia, et al. Approaches to distributed execution of scientific workflows in kepler. *Fundamenta Informaticae*, 128(3):281–302, 2013.

[31] Zoltan Prekopcsak, Gabor Makrai, Tamas Henk, and Csaba Gaspar-Papanek. Radoop: Analyzing big data with rapidminer and hadoop. In *Proceedings of the 2nd RapidMiner community meeting and conference (RCOMM 2011)*, pages 1–12. Citeseer, 2011.

[32] Provenance Management. `http://www.taverna.org.uk/documentation/taverna-2-x/provenance/`. Accessed: 2017-07-05.

[33] SCAPE. `http://www.taverna.org.uk/introduction/related-projects/scape/`. Accessed: 2017-07-05.

[34] Karan Vahi, Ian Harvey, Taghrid Samak, Daniel Gunter, Kieran Evans, David Rogers, Ian Taylor, Monte Goode, Fabio Silva, Eddie Al-Shakarchi, et al. A case study into using common real-time workflow monitoring infrastructure for scientific workflows. *Journal of grid computing*, 11(3):381–406, 2013.

[35] Wil MP Van Der Aalst. Three good reasons for using a petri-net-based workflow management system. In *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPICâ96)*, pages 179–201. Citeseer, 1996.

[36] Wil MP Van Der Aalst, Arthur HM Ter Hofstede, and Mathias Weske. Business process management: A survey. In *International conference on business process management*, pages 1–12. Springer, 2003.

[37] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, page gkt328, 2013.

[38] KNIME Performance Extensions. `https://www.knime.org/knime-performance-extensions`. Accessed: 2017-07-05.

[39] Workflow Monitor. `http://www.taverna.org.uk/download/associated-tools/workflow-monitor/`. Accessed: 2017-07-05.

[40] Ustun Yildiz, Adnene Guabtni, and Anne HH Ngu. Business versus scientific workflows: A comparative study. In *Services-I, 2009 World Conference on*, pages 340–343. IEEE, 2009.

[41] Ustun Yildiz, Adnene Guabtni, and Anne HH Ngu. Towards scientific workflow patterns. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, page 13. ACM, 2009.

[42] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Record*, 34(3):44–49, 2005.

[43] Yong Zhao, Ioan Raicu, and Ian Foster. Scientific workflow systems for 21st century, new bottle or new wine? In *Services-Part I, 2008. IEEE Congress on*, pages 467–471. IEEE, 2008.