Mastering Data-Intensive Collaboration and Decision Making

FP7 - Information and Communication Technologies

Grant Agreement no: 257184
Collaborative Project
Project start: 1 September 2010, Duration: 36 months

D3.1.1 – The Dicode Data Mining Framework (initial version)

Due date of deliverable:	31 May 2011
Actual submission date:	31 May 2011
Responsible Partner:	FHG
Contributing Partners:	FHG, NEO, CTI, UOL

Nature: ■Report ☐Prototype ☐ Demonstrator ☐Other

Dissemination Level:

■ PU : Public
☐ PP : Restricted to other programme participants (including the Commission Services)
☐ RE : Restricted to a group specified by the consortium (including the Commission Services)
☐ CO : Confidential, only for members of the consortium (including the Commission Services)

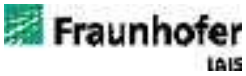
Keyword List: Scalable high performance data mining, data mining towards sense-making, collaboration support

The Dicode project (dicode-project.eu) is funded by the European Commission, Information Society and Media Directorate General, under the FP7 Cooperation programme (ICT/SO4.3: Intelligent Information Management).

The Dicode Consortium

Research Academic Computer Technology Institute (CTI) (coordinator), Greece

University of Leeds (UOL), UK



Fraunhofer-Gesellschaft zur Foerderung der angewandten Forschung e.V. (FHG), Germany

Universidad Politecnica de Madrid (UPM), Spain

Neofonie GmbH (NEO), Germany

Image Analysis Limited (IMA), UK

Biomedical Research Foundation, Academy of Athens (BRF), Greece

Publicis Frankfurt Zweigniederlassung der PWW GmbH (PUB), Germany

Document history			
Version	Date	Status	Modifications made by
1	10-05-2011	First draft	Axel Poigné, FHG
2	19-05-2011	Second draft (sent to internal reviewers)	Axel Poigné, FHG Jörg Kindermann, FHG Manuel Meßner, NEO
3	24-05-2011	Reviewers' comments incorporated (sent to SC)	Axel Poigné, FHG Ahmad Ammari, UOL Nikos Karacapilidis, CTI
4	29-05-2011	SC's comments incorporated	Axel Poigné, FHG Jörg Kindermann, FHG Manuel Meßner, NEO Manolis Tsagarakis, CTI Guillermo de la Calle Velasco, UPM
5	31-05-2011	Final (approved by SC, sent to the Project Officer)	Axel Poigné, FHG Nikos Karacapilidis, CTI

Deliverable manager

- Axel Poigné

List of Contributors

- Jörg Kindermann, FHG
- Doris Maassen, NEO
- Manuel Meßner, NEO
- Axel Poigné, FHG
- Stefan Rüping, FHG

List of Evaluators

- Ahmad Ammari, UOL
- Nikos Karacapilidis, CTI

Summary

This deliverable reports on the design and specification of the architecture of the Dicode Data Mining Framework, which is considered as an instance of the Service-Oriented Architecture. This approach is justified from both the user's and the developer's point of view. The related components of the Service-Oriented Architecture are sketched and considered in the context of Dicode. Then, the particular data mining components of the proposed framework are discussed. Finally, experimental results motivating some of the choices made so far are presented in the Appendices, together with examples of abstract service descriptions that are considered as blueprints for the proposed framework.

Table of Contents

1 Introduction

The Dicode Data Mining Framework covers architectural considerations as well as a preliminary list of data mining algorithms to be provided. The architecture shall combine maximum flexibility and development independence with a high degree of interoperability. Dicode proposes a Service-Oriented Architecture for achieving these goals. This choice is justified below from both the users' and the developers' perspective. Particular aspects of Service-Oriented Architecture, being relevant for the Dicode Data Mining Framework, are discussed in Section 2. Individual components, such as specific data mining algorithms, exploitation of available data mining frameworks and other contributing technologies are outlined in Section 3. Finally, Appendix A presents experimental results that motivate the choices made in Section 3, while Appendix B sketches preliminary definitions of Dicode Data Mining Services.

1.1 User Perspective

Users will use the Dicode Data Mining Framework either by using individual data mining algorithms or by composing several components of the framework as a *workflow*. The latter presumes that the respective components are interoperable and easy to compose. Interoperability implies that results obtained by one component are in a format acceptable as input by the component supposed to execute subsequently or that intermediate components are available that transform the output into an acceptable input format. The user must be supported in that information about the compatibility of components should be available and recommendations should be given about which components may be applied subsequently to output data obtained so far or about which transformations into suitable input formats are available.

There are several strategies to compose components into workflows:

- *Use of Prefabricated Workflows* – A frequently used workflow is prefabricated and provided as a component. The workflow may be instrumented, in that particular parameters can be set and particular subcomponents can be chosen. In the latter case, we speak of a *workflow pattern*.
- *On-the-fly Workflow Composition* – The idea is that a user interactively steps through a workflow, either guided (i.e. next steps are recommended) or freely (i.e. the user may choose which component to apply next to which data). Guidance may for instance be given by stepping through a series of web pages using links to go to the next component or by using drag-and-drop functionalities on a web page that offers widgets to access input data, components, and (intermediate) results.
- *Use of a Workflow System* – There are several workflow systems such as Taverna[1], Kepler[2], Triana[3] and Vistrails[4] to name a few. Workflow systems typically consist of a workflow *editor* and a workflow *enactor*, the latter being responsible for controlling the execution of the workflow edited.

These three strategies correspond to three types of users:

[1] http://www.taverna.org.uk/
[2] https://kepler-project.org/
[3] http://www.trianacode.org/
[4] http://www.vistrails.org/

- A user who just wants to use a standardised procedure to achieve results as fast and reliable as possible. Such a user is best served with prefabricated workflows.
- A user who needs some freedom for experimentation but refrains to design a full-flavoured workflow and does not care for some repetitive action. Such a user is best served with on-the-fly workflow composition.
- A user who develops and improves a workflow over some period of time and who might want to share and discuss the workflow with the community. myExperiment[5], for instance, is an example for a community platform of this kind.

The Dicode Data Mining Framework aims to support all these types of users. While a specific externally provided workflow system may be chosen, a likely candidate being Taverna, Dicode will develop a communication and interaction platform to support the other options.

Finally, all three strategies sketched above can reuse components being services. They differ only in the interaction principles (what is called *platform logic* in Section 2.4.3), which also justifies the use of a Service-Oriented Architecture. The scenarios are very distinct with regard to the interaction with the users, each with its particular advantages and disadvantages.

1.2 Developer Perspective

Developers want to satisfy user requirements with a minimum of effort, using the same code basis in the different contexts sketched above. The natural granularity is that of components that hide the computational devices delivering the functionality. From the developers' perspective, a Service-Oriented Architecture with services as components provides the grade of granularity and flexibility needed to serve all purposes.

As discussed in the state-of-the-art survey (deliverable D2.1), computational demands of data mining on storage and speed ask for potentially very different technological solutions and platforms. Experimentation with different technological solutions (for a summary, see Appendix A) confirms the findings of the state-of-the-art survey. The integration of these solutions also demands for an abstraction that hides these differences. Again, service-orientation is a suitable abstraction mechanism.

2 The Data Mining Framework as Service-Oriented Architecture

2.1 Why Service-Oriented Architecture

The different workflow strategies outlined in the previous section spot different interaction mechanisms but should be implemented on top of the same code basis to reduce design and implementation efforts. This demands for a flexible but standardised component paradigm that support reuse. The paradigm of Service-Oriented Architecture seems at present to be most appropriate to satisfy these requirements, offering an established technology with many resources already available and providing the right degree of granularity and flexibility. Available workflow systems such as Taverna or Kepler can embed services as components within their workflows and web services can – of course – be embedded into web-

[5] http://www.myexperiment.org/home

based portals. Therefore, Dicode will use the paradigm of service orientation as architectural basis of the Dicode Data Mining Framework (see also deliverable D2.2, Section 6.4).

2.2 Particular Requirements with regard to Data Mining

The choice of a Service-Oriented Architecture is somewhat generic and does not particularly reflect the concerns of a data mining framework. There are a couple of potential characteristic features that distinguish data mining algorithms from typical service-oriented applications:

- Data mining algorithms may run for a long time, from hours up to days;
- Data mining algorithms may need a huge amount of storage, and
- Some data mining algorithms need user interaction; for instance, interaction to decide whether an approximation is sufficient or whether more iteration is needed (this is possibly the most characteristic feature).

Again, these features are probably common in a number of other algorithms but ask for specific service interfaces to monitor the underlying processes and interact with them.

2.3 Aspects of Services

Several aspects of services are discussed below setting the border conditions for developing services within Dicode Data Mining Framework (and for the Service Framework for Dicode in general). The discussion reflects the respective sections of deliverable D2.1 (Section 6).

2.3.1 Service Description

The OASIS draft proposal "Reference Architecture for Service Oriented Architecture" OASIS-SOA-RA[6] captures – in a rather precise way - the elements of Service Description that defines the information needed to use, deploy, manage, and otherwise control a service.

The elements of the OASIS Service Description are illustrated in Figure 1:

- *Service Functionality* describes what can be expected when interacting with a service, in that it defines the interfaces, operations, and parameter types of the service syntactically and semantically. The operations produce the desired Real World Effects.
- The *Service Interface* is the means for interacting with a service. It includes the specific protocols, commands, and information exchange in terms of messages by which actions are initiated that result in the real world effects as specified through the Service Functionality portion of the service description. The *Information Model* determines the structure and semantics of messages, while the *Action Model* of a service is the characterization of the actions that may be invoked against the service and the message exchange pattern used to perform an action. The *Process Model* captures the behavioural aspects of the interactions in which a service can engage to achieve a (business) goal. This includes, for instance, the temporal relationships and temporal properties of actions and events.
- *Service Reachability* enables service to locate and interact with one another. An *Endpoint* determines the logical location of a service to which a message can be send according to some protocol in order to invoke an action.

[6] http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.pdf

- *Polices and contracts* are part of the service description. Service contracts should be comprised of functional components such as type, functionality, operations, invocation methods, location, pre- and post conditions, and participation in orchestration or choreographies, as well as non-functional components such as security constraints, Quality of Service, semantics, transactional properties[7], and service level agreements. In other words, contracts and policies constitute meta-information about a service.

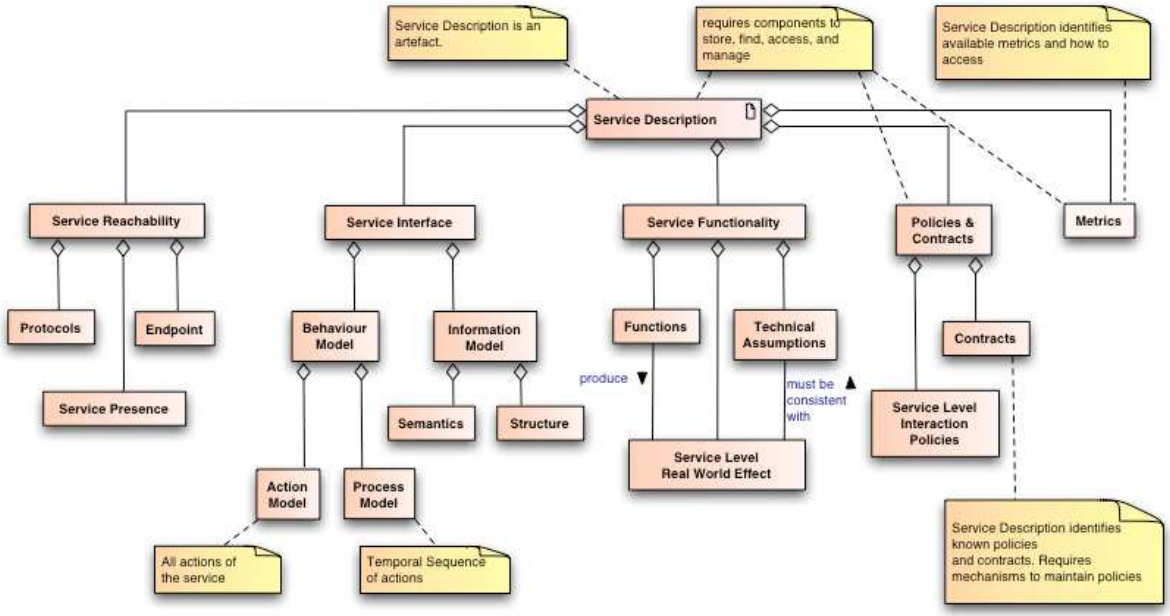


Figure 1: OASIS-SOA-RA class diagram of a service description

2.3.2 Service Development Process

Service development strategies[8] distinguish between abstract service specifications developed independently of any middleware technology (typically in form of UML diagrams) and concrete service specifications being implemented on a specific service platform (see Figure 2).

In detail, the development phases considered are:

- The *analysis phase*, resulting in requirements.
- The *abstract design phase*, which leads to platform-neutral specifications. These specifications define the informational requirements (information model), functional requirements (abstract service specifications), and non-functional requirements (specification of the quality of service (QoS).
- The *concrete design phase*, which maps the abstract specifications to a chosen concrete *service platform*, i.e. an infrastructure needed to use and interoperate services.
- The *engineering phase*, where the platform-specific components are organised into service networks taking into account the QoS requirements and translating them into operational policies.

[7] Whether it is capable of acting as part of a larger transaction.
[8] See, e.g., http://www.eu-orchestra.org/

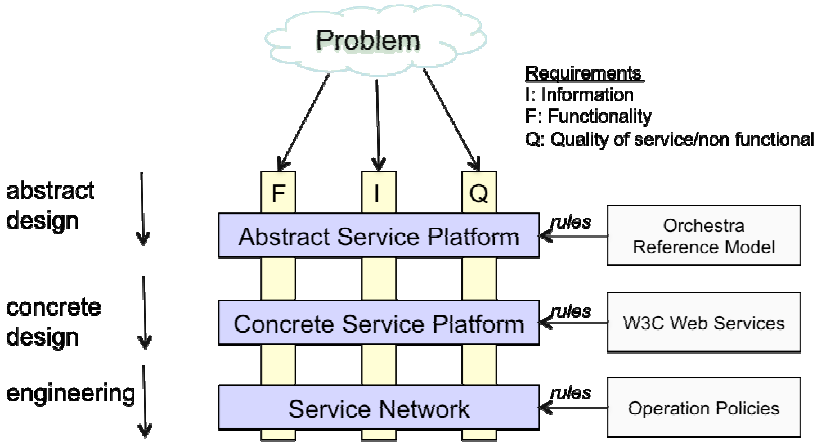


Figure 2: Abstract and concrete service platforms

In an agile process, as foreseen in Dicode (see deliverable D2.2, Section 8), development will be an evolutionary cyclic process aiming to establish a usable code basis as soon as possible. This may even imply that development starts with a concrete design aiming for an abstract design only later when some stability has been achieved, or even omitting an abstract design, but the general principle shall still apply and justifications about design decisions should be given at each step within the development.

2.3.3 Service Platform

A platform defines the languages and mechanisms needed for specifying the information model, the action model, and the process model for a service network. This includes:

- *Interface Language:* Formal language used for specification of service interfaces
- *Execution Context:* Specification of the Execution Context as an agreement between service providers and consumers that contains information about, e.g., preferred protocols, semantics, policies, and other conditions and assumptions that describe how a service can and may be used.
- *Data Formats:* An agreement on the usage of standard data formats and specific/proprietary data formats may be part of platform specification.

Specific aspects may be covered as well such as User Management, Authentication and Authorisation.

In order to ensure a basic interoperability of services, the languages and components of a platform should be specified precisely, meaning that the specific version of a language or version of a library is indicated, e.g. that the Web Service Description Language (WSDL) version 1.1 is used (and not WSDL 2.0).

2.3.4 SOAP versus REST

Service Orientated Architecture (SOA)[9] reflects two traditions: Remote Procedure Calls (RPC) and information hiding as promoted by Object-Oriented Architecture (OOA), which results in the idea that interfaces consist of operations and that data can only be accessed and changed by applying these operations. The well-known SOAP protocol[10] provides a number of standards[11] supporting the SOA perspective.

[9] See http://en.wikipedia.org/wiki/Service-oriented_architecture
[10] http://en.wikipedia.org/wiki/SOAP
[11] http://en.wikipedia.org/wiki/List_of_web_service_specifications

A second tradition (of database origin) leads to a different – more recent – architectural style: Resource-Oriented Architecture (ROA)[12] or RESTlike[13] architecture. Here, a "service" offers a set of resources, each of which supports the *same* interface that only consists of operations familiar from HTTP, namely: GET, PUT, DELETE, POST. The contrast is obvious: information hiding is abandoned in favour of direct access to resources (data), while indirect access to data via operations is replaced by direct access operations.

The advantages and disadvantages of the two paradigms have been extensively discussed[14]. The main benefit of SOAP is standardisation (at the price of some overhead), while the main benefit of REST is flexibility and being more lightweight (at the price of missing standards). Dicode will optionally target both paradigms (see Section 2.4.1).

2.3.5 On Standards

Most important standards concerning services and metadata are ISO 19119[15] and ISO 19115[16]. The WS-*[17] specifications provide a set of quasi-standards by W3C in the context of SOAP/WSDL based web services.

2.3.6 Security Standards

Security is one aspect of confidence in the internality, reliability, and confidentiality of a system. It focuses on avoiding inappropriate access to resources and on accidentally or intentionally misuse of the infrastructure.

ISO/IEC 27002 characterizes the following key security concepts:

- *Confidentiality:* Protection of privacy of participants in their interactions: messages should not be readable to third parties, but the degree of visibility if messages are exchanged and of the participant's identity to third parties can be defined
- *Integrity:* Protection of altering exchanged information
- *Availability:* Concerns the reliability of a system, in other words, if the system offers the service for which it is designed, and the security concept needed to respond to active threats to the system
- *Authentication:* Concerns the means of identifying the participants in an interaction
- Authorisation: Concerns the means of legitimacy of the interaction, the exchanged actions must be explicitly or implicitly approved
- *Non-repudiation:* Concerns the accountability of participants: participants should not, at a later time, successfully deny having participated in the interactions.

2.4 Bearings on the Dicode Data Mining Framework

2.4.1 General

Dicode optionally targets both SOAP based as well as REST based service platforms (see deliverable D2.1, Section 6.4). The rationale is to obtain a maximum of flexibility in serving both

[12] See http://en.wikipedia.org/wiki/Resource-oriented_architecture,
[13] See http://en.wikipedia.org/wiki/Representational_State_Transfer
[14] See http://www.prescod.net/rest/rest_vs_soap_overview/
[15] http://www.iso.org/iso/catalogue_detail.htm?csnumber=39890
[16] http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020
[17] See http://de.wikipedia.org/wiki/WS-*

paradigms. The targeting will be on demand. Costs shall be reduced by using wrappers for adapting the same code basis for the different targets (see Section 2.4.3).

2.4.2 Service Description

Dicode will provide functional specifications of services in terms of abstract descriptions. For examples, see Appendix B. For SOAP based services, (possibly semantically annotated) WSDL specifications shall be used formally to specify interfaces, information and behaviour models as well as service reachability. For RESTful services, WASDL may be used but Dicode may agree on an ad-hoc formalism.

2.4.3 Service Development

Minimal requirement is the provision of a platform-dependent service specification, preferably complemented by a platform-neutral specification. Platform-dependent specifications shall conform to the requirements specified in Section 2.4.4. Platform-neutral specifications shall be in UML format and correspond to rules of ISO 19119 "Service Specification" (use of a UML tool will satisfy the latter condition).

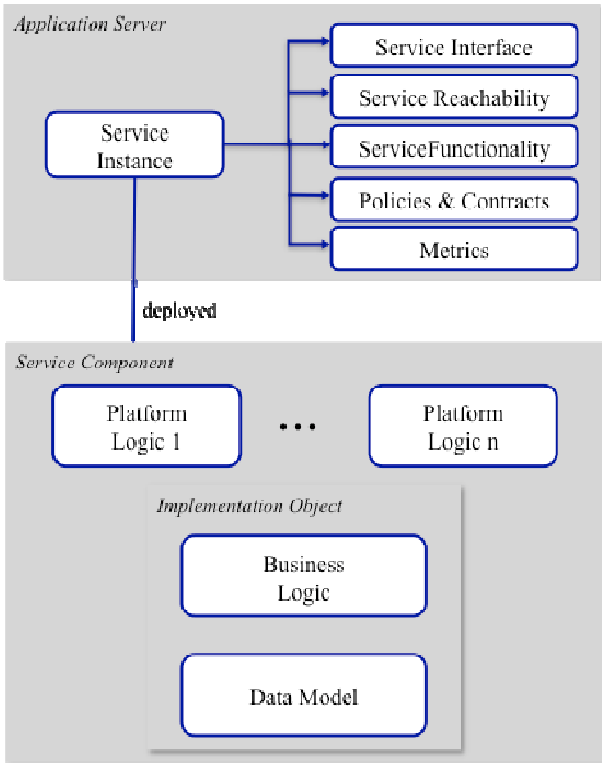


Figure 3: Service Instance and Service Component

The software implementation of a service (specification) is referred to as a *service component* (see Figure 3), the deployment of which is a *service instance*. Service instances interact with other service instances. To do so, a service instance needs to expose its description including information about reachability, functionality, etc.

A service component shall consist of an *implementation object* and one or more *platform logics*. The implementation object represents the software components behind the service. Usually it comes in two tiers, a *data model* and the *business logic*. Platform logic is a façade (as design pattern) of the implementation object (given as, for instance, a piece of Java code) with regard to a specific implementation platform (SOAP or REST, if both are supported). Separa-

tion between platform logic and implementation object is meant to support reuse, thus preserving investments in software.

2.4.4 Service Platform

General

Services foreseen in Dicode will be Web services.

Interface language

Interface language is WSDL (Web Service Description Language) or SAWSDL if semantic information is incorporated. Preferred is version 2.0 since this version covers RESTful web services as well. This choice is up to a final decision. If WSDL1.x is chosen, an acceptable format for RESTful web services has to be defined.

Execution context

The message exchange protocol may be, for instance, SOAP 1.2 with HTTP as transport protocol. The message style may be defined in terms of a MIME type, for example "document/literal non-wrapped".

For security, the basic mechanism may be SOAP Message Security for encryption of SOAP messages. As a requirement, the execution context may state that session information must be included in the SOAP header.

RESTful services might require HTTP Basic or HTTP Digest or SSL certificate-based mutual authentication for propagating identity credentials.

Schema language for definition of data format

Prototypical schema language is XML ("Extended Schema Language"), version 1.0[18]. There exist many restrictions of the general scheme. Dicode will require specification of the schema languages to be used. These should preferably be common to the use case scenarios whenever feasible. As an alternative, in particular for inter-service data exchange, JSON[19] or CSV[20] may be used.

Schema Mapping

The general idea of schema encoding is that the class definitions in UML are mapped to type and element declarations in a schema language, so that the objects in the instance model can be mapped to corresponding element structures in the schema language, e.g. XML) document. Standard schema encodings ensure interoperability between services. Schema encodings shall be defined in Dicode whenever appropriate.

2.4.5 Security

At present stage, security requirements are not fully clear. It is anticipated that in some scenarios authentication and authorisation will be needed. Shibboleth[21] may provide the suitable means to satisfy security requirements within Dicode.

[18] http://www.w3.org/TR/2004/REC-xml-20040204/
[19] http://en.wikipedia.org/wiki/JSON
[20] http://en.wikipedia.org/wiki/Comma-separated_values
[21] See http://shibboleth.internet2.edu/: "The Shibboleth System is a standards-based, open source software package for web single sign-on across or within organizational boundaries. It allows sites to

2.5 Service Types

A *service type* is a category of related services that use a common schema. Below is a list of service types that are relevant for Dicode. Data mining services will typically be considered as processing services complemented by notification and monitoring services, by catalogue (discovery) and access services for data acquisition, and by visualisation services.

Service	Short Description
Catalogue Service	Supports the ability to publish, query, and retrieve descriptive information for resources data, independent of a specific meta-information standard
Access Service	Allows interoperable read and write access on feature instances available in an Service Network
Ontology Access Service	Read access to the specification of a logical ontology, export or import of complete specifications into an ontology store.
Annotation Service	Automatically generation of specific meta-information from various sources and relation with semantic descriptions
Processing Service	Common interface for services offering processing operations by initiating the calculation and managing the outputs to be returned to the client.
Notification Service	Allows to send messages to a client, which previously has been registered to listen for certain events
Service Monitoring Service	Provides an overview about Service Instances currently running within a Service Network
Visualisation Service	Common interface for services offering visualisation of data in form of diagrams, charts, or other visual formats.
Authentication Service	Verifies genuineness of principals using a set of given credentials
Authorisation Service	Gives a compliance value as response to a given authorisation context
User Management Service	Creates and maintain subjects including groups of principals as entities that need authentication.

Table 1: Service types

A number of these service types, most notably catalogue and access services, have standard specifications and implementation rules (look e.g. the widely used OGC standards[22]). The Dicode Data Mining Framework shall adhere to these standards whenever feasible.

Regarding applications and services integration, we envisage the use of a service-oriented approach, also supported by the Dicode Service Ontology (DSO) (see deliverable D2.1, Section 6.4).

make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner".

[22] http://www.opengeospatial.org/standards

3 Components of the Data Mining Framework

3.1 Frameworks

There are several frameworks available for data mining. The Dicode Data Mining framework will take advantage of these frameworks by using the functionalities offered but will encapsulate these functionalities in terms of services, thus hiding the internal workings of the different frameworks.

We have performed a series of preliminary experiments featuring various data mining algorithms and exploring the potential of several frameworks. In particular, we tested a sequential implementation of a text mining algorithm (LDA[23]) against the Mahout implementation, a sequential text mining pre-processing workflow against Mahout, and two Map/Reduce implementations of the k-means clustering algorithm (Twister against Mahout). The results are presented in Appendix A. Generally speaking, the intention of these experiments was to gain insight in the characteristics and the usefulness of these frameworks for Dicode.

3.1.1 Mahout[24]

Apache Mahout currently has a very active development community. It has been reported to be used as part of the spam filtering pipeline at Yahoo![25], for matching couples at Speeddate[26], as well as a part of the recommendation modules at AOL and Foursquare[27].

Mahout's goal is to provide scalable machine learning libraries for the Java[28] programming language. Mahout implements most of its algorithms on top of Apache Hadoop[29] using the Map/Reduce paradigm. Therefore, it is perfectly suited for massive data that is stored in a Hadoop Cluster.

3.1.2 Twister[30]

Twister is a Map/Reduce implementation that has been described in deliverable D2.1 (Section 3.4.1, page 25). It uses pub/sub messaging for all the communication/data, eliminating the need for a specialised distributed file system. If the computational resources required by an algorithm are large in relation to the data volume, a considerable speed-up can be observed in comparison to Hadoop. An example is given in Appendix A.3).

3.1.3 RapidMiner[31]

RapidMiner provides many data mining and machine learning procedures including data pre-processing and visualization. These can be nested to construct complex data mining processes. It integrates learning schemes and attribute evaluators of the machine learning environment WEKA and statistical modelling schemes of the R-Project. There is little support yet for distributed data mining.

---

[23] Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3, 993–1022.
[24] http://mahout.apache.org/
[25] http://www.slideshare.net/hadoopusergroup/mail-antispam
[26] http://ww.speeddate.com, https://cwiki.apache.org/MAHOUT/powered-by-mahout.html
[27] https://cwiki.apache.org/MAHOUT/powered-by-mahout.html
[28] http://www.oracle.com/us/technologies/java/index.html
[29] http://hadoop.apache.org/
[30] http://www.iterativemapreduce.org/
[31] http://rapid-i.com/content/view/181/190/lang,en/

A very recent project called Radoop integrates Hadoop in RapidMiner by providing a Hadoop extension for RapidMiner. Since Mahout uses Hadoop, it will then be possible to use Mahout class library from within RapidMiner.[32]

3.1.4 Weka[33]

Weka is an alternative to RapidMiner also providing many machine learning and data mining algorithms. As for RapidMiner, little support for distributed data mining seems to be available.

3.1.5 R[34]

R is a programming language and development environment for statistical computing and as a *de facto* standard for developing statistical software. In Dicode use case 1, R is used extensively but on the level of scripts (see Section 3 of deliverable D2.2). Dicode will split these scripts into reusable services using Rserve[35]as enactment machine. Rserve is a TCP/IP server that allows other programs to use the functionality of R. This allows combining R-based algorithms with other services. First experiments show the feasibility, for instance, of Taverna to interact with Rserve[36] (see Appendix A.3A.5).

Several packages support distributed computing in R. For instance, the R-Package GridR[37] allows submitting R functions for execution on remote computers, clusters or grids. Further, several data mining packages are available for Text Mining (such as Rattle[38] and tm[39]).

3.2 Particular Data Mining Services

This section lists a number of individual data mining services that will be provided by Dicode. At present, the list is preliminary and by no means exhaustive.

3.2.1 Twitter Harvester

This service is located at the front of the production process. It federates two different possibilities to obtain tweets (e.g. status updates) from Twitter using Twitter's Streaming API[40] and its Search API[41]. Results from both APIs get stored into HBase[42] and a RESTful API is provided to control Dicode's Twitter Harvester (see Figure 4).

Our component Twitterstream establishes a connection to Twitter's sampling streaming service, which current sampling rate is ~1% of all public status updates. Twitter's algorithm for its sampling streaming service is as follows: "The status id modulo 100 is taken on each public status. [...] Modulus value 0 is delivered" (to the stream). "Over a significant period, a 1%

[32] http://prekopcsak.hu/index.php?slug=radoop
[33] http://en.wikipedia.org/wiki/Weka_(machine_learning)
[34] http://en.wikipedia.org/wiki/R-Project
[35] http://www.rforge.net/Rserve/
[36] http://rosuda.org/Rserve/
[37] http://cran.r-project.org/web/packages/GridR/
[38] http://rattle.togaware.com/
[39] http://cran.r-project.org/web/packages/tm/index.html
[40] http://dev.twitter.com/pages/streaming_api
[41] http://dev.twitter.com/doc/get/search
[42] http://hbase.apache.org/

[…] sample of public statuses is approached. This algorithm, in conjunction with the status id assignment algorithm, will tend to produce a random selection."[43]

While Twitterstream provides a random sample of status updates, Twittertracker obtains tweets about specific topics using Twitter's Search API. Therefore, service users will be able to configure a set of queries via Twitter Harvester's RESTful API. Twittertracker fires those queries regularly using a simple round robin scheduling and respecting Twitter's dynamic rate limits[44].

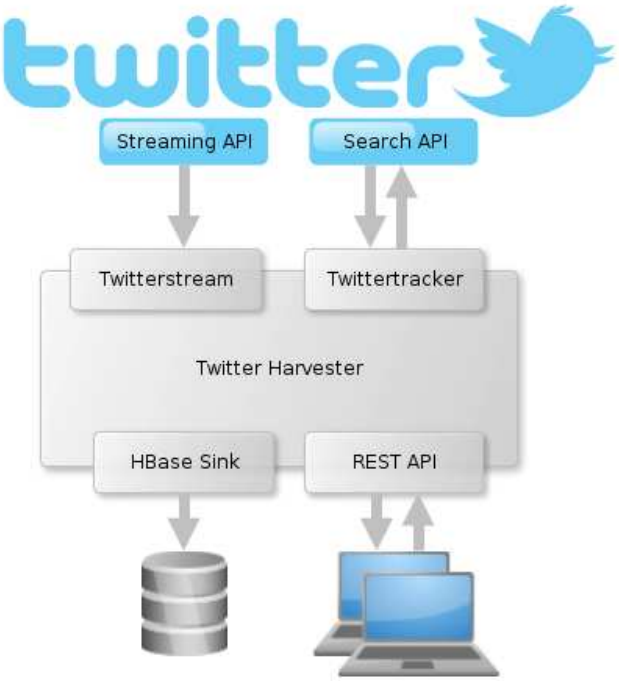


Figure 4: Dicode's Twitter Harvester

3.2.2 Data pre-processing

Data pre-processing features several operations. These will be part of more complex components to be provided as services. They can be switched on or off by sending parameters to the respective components. Data pre-processing operations may be very specific for a particular data source or they may be of more general nature (being reusable for many data sources). In the latter case, such operations will be made available as publicly accessible operations of services. The data pre-processing operations will address issues such as spam-filtering and the reduction of spelling errors. This will be achieved by simple pattern matching. Other pre-processing operations are concerned with assigning part-of-speech tags to the texts, segmentation of texts into sections or sentences, data transformation between different analysis services (e.g. transforming numerical attribute into categorical, transforming continuous attribute into discrete, etc), or attribute reduction to reduce dimensionality of data.

3.2.3 Automatic Tagging

The auto tagging service analyses text passages to determine relevant catchphrases. Therefore the service dissects the given text using linguistic and statistic methods in order to find

[43] http://dev.twitter.com/pages/streaming_api_concepts#sampling
[44] http://dev.twitter.com/pages/rate-limiting

candidates for catchphrases. The service will classify those candidates using different kind of features, e.g. "title is more important than body text, length, or part of speech" or whether Wikipedia contains a candidate. The service will be provided by Dicode. We are not aware of an existing service of this kind.[45]

3.2.4 Named Entity Recognition

Terms like proper names, geographical locations, enterprise names, or commercial products are identified in the texts and assigned their correct class label (like "name", "location", etc.). It may be necessary not only to annotate named entities in the classical sense, but also to identify key phrases that are typical for the domain. We will implement a workflow to annotate named entities based on technology developed by FHG[46].

3.2.5 Sentiment analysis

This service identifies text passages that include sentiments, and – if possible – also indicates the nature (positive versus negative or further categories) of the sentiment. Sentiments can be identified according to contextual regularities, much in the same way as named entities (of course, there are also sophisticated ways to express one's sentiments between the lines; the more complicated cases of sentiment expression are not yet accessible to automated analysis). The development will be based on FHG technology. Currently, no publications exist.

3.2.6 Emotion Detection

This service detects some types of emotions, like satisfaction, anger, surprise, etc. This is still leading edge research, so the results are not fully predictable. The development will be based on FHG technology. Currently, no publications exist.

3.2.7 Relation Extraction

Simple semantic relations are identified from the typical usage of terms and syntactic constructs in texts. Here, we detect certain prototypical relations between persons, products, and brands that are related to emotions. The results in this case are also in part depending on the research to be done in Dicode. The development will be based on FHG technology.[47]

3.2.8 Opinion Mining

The results obtained from the previous services are combined to extract opinions that the authors may have with respect to products, brands, or related topics. The development will be based on FHG technology. Currently, no publications exist.

3.2.9 Subgroup Discovery

Subgroup Discovery is concerned with detecting patterns in data that are infrequent but very informative in large data sets. It allows identifying possibly interesting local correlations and dependencies.

---

[45] For instance, the service is not available in OpenNLP (http://incubator.apache.org/opennlp/) or GATE (http://gate.ac.uk/)

[46] Pilz, A. and Paaß, G, Named entity resolution using automatically extracted semantic information, In Proceedings of LWA 2009, Darmstadt.

[47] Reichartz, F. and Korte, H. and Paass, G.: Dependency tree kernels for relation extraction from natural language text. In: Journal Machine Learning and Knowledge Discovery in Databases, pp. 270-285, Springer, 2009.

3.2.10 Similarity Learning

The key idea of similarity learning is to learn a function that produces a non-negative real number for any pair of examples. The intended semantic is that the higher this number the more similar the two examples are. The training data that the function learns from consist of example pairs labelled as similar or dissimilar.

Similarity learning is a promising technique for the broad categories of problems involving comparison, search or matching, especially in cases where the notion of similarity is problem- and application-dependent or is hard to define. Similarity learning is important in the following applications:

- An appropriate similarity function is the important prerequisite for clustering analysis or for any kind of similarity-based classification methods, such as e.g. k-Nearest Neighbour.
- There are certain kinds of problems that are similarity learning tasks by definition, e.g. duplicate detection. Although we can adapt several classification methods to solve them, these problems are originally similarity learning tasks.
- In some instance spaces, it is easier to say whether two items are similar (to define the set of basic similarities) than to adequately represent them in a feature space. The similarity learning does not assume that instances are well represented in some feature space. As long as the similarity function is well defined for any pair of instances, we can apply similarity learning.

The important requirement for similarity learning is a set of meaningful local similarities that can be defined by the domain expert. This is one of the most crucial steps that directly impact the quality of the future model. It requires specific knowledge about the addressed application domain. This step becomes even more important when comparison should include user requirements. In this case, the local similarities are used not only to estimate the differences between the objects, but also to reflect the specific user needs.

Basic Similarity Services

As long as the proper basic similarities are defined, we can easily compute the similarity of the complex objects. The following types of services for computing local similarities will be provided:

- String similarity provides float based similarity measures between string Data e.g. Levenstein similarity, Hamming similarity, Soundex similarity etc.
- Similarity of structured data, e.g data that are described by graphs, a taxonomy, or an ontology.
- Similarity of numerical data: Euclidean distance, Cosine similarity, Inner Product, Pearson similarity.
- Text similarity provides the similarity between two texts (Cosine similarities based on k-grams).

Distance Metric Learning

The performance of many learning and data mining algorithms depend critically on being given a good metric over the input space that reflect reasonably well the important relationships between the data. Distance Metric Learning intends to generate suitable metrics for

data as a basis for, e.g., Similarity Learning. Different services will be provided that implement different similarity learning algorithms.[48]

Application Services

Similarity learning can be applied in a number of application areas, for instance in recommendation systems. Typical tasks in this context will be supported by services:

- *Ranking*: On basis of known examples a list of ranked recommendations is provided;
- *Deduplication*: Similarity of objects are evaluated against a threshold;
- *Classification*: Predicting group membership for data instances based on similarity metrics.

3.2.11 Relation to the Use Cases

The data mining services in Sections 3.2.1 to 3.2.8 mainly relate to use case 1 though some of the text mining algorithms may be also used within use case 2. Subgroup discovery and similarity learning will be applied in use case 1 and use case 2. Similarity learning will in particular be used for recommendation systems. The relation of opinion mining services to use case 3 is obvious.

4 Summary

All data mining services specified in Section 3.2 will be provided as part of the Dicode Data Mining Framework. Provision of further services will be envisaged but depend on demand. The frameworks cited in the previous sections will play a dual role. On one hand, they will be used as a code basis from which to take particular algorithms to be encapsulated as services. This applies, for instance, for Weka and R. On the other hand, they will be used as execution environments for distributed data mining services. This applies to R/Rserve/GridR, Mahout, and Twister. The Map/Reduce frameworks Mahout and Twister will be used alternatively according to their different characteristics as exposed by the experiments in Appendix A.

[48] Such as those of (i) E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems 15, 2002 and (ii) J.V. Davis and I.S. Dhillon. Structured metric learning for high dimensional problems. In KDD, pp. 195--203, 2008.

Appendix A Experimental Results

This section describes experiments that have been carried out to evaluate prototypical algorithms in some of the eligible processing frameworks.

For experiments A.1, A.2, A.3 and A.4 we chose four algorithms: (i) a typical pre-processing task that creates a numerical representation of documents called "bag-of-words vectors"; the algorithm was simple, in that it only counted occurrences (frequencies) of terms, but did not apply any weighting scheme like TFIDF[49]; (ii) a clustering algorithm called LDA[50], (iii) the well-known k-means algorithm[51], and (iv) a supervised classification algorithm, the Support Vector Machine[52]. These algorithms are most closely related to services in use case 3, but in part will also be applied in use case 1.

The processing frameworks that we compared were: (i) sequential processing versus Map/Reduce in A.1, A.2 and A.3, and (ii) sequential processing versus data parallel processing in experiment A.4.

Experiment A.5 relates to use case 1. The intention is to prove that a workflow system such as Taverna provides sufficient functionality to represent typical workflows of this use case.

A.1 Text mining pre-processing workflow

The workflow processes newswire documents of the news agency "Reuters". Various collection sizes were tested, the largest one containing 272,778 documents. The documents are transformed into a numerical representation called "bag-of-words vectors". The sequential version was implemented using the text mining toolbox "Mallet"[53], the Map/Reduce version was used as provided by Mahout.

In Figure 5, we observe that with increased number of documents Mahout rapidly outperforms the sequential pre-processing procedure. The relation is even more clearly seen in Figure 6: the processing time per document increases linearly with the total number of documents in the sequential case, whereas it decreases in the Map/Reduce case.

[49] http://en.wikipedia.org/wiki/Tf%E2%80%93idf

[50] Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation. Journa Machine Learning Research, 3, pp. 993–1022.

[51] http://en.wikipedia.org/wiki/K-means

[52] Cristianini, N. and Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines. Cambridge University Press.

[53] http://mallet.cs.umass.edu/

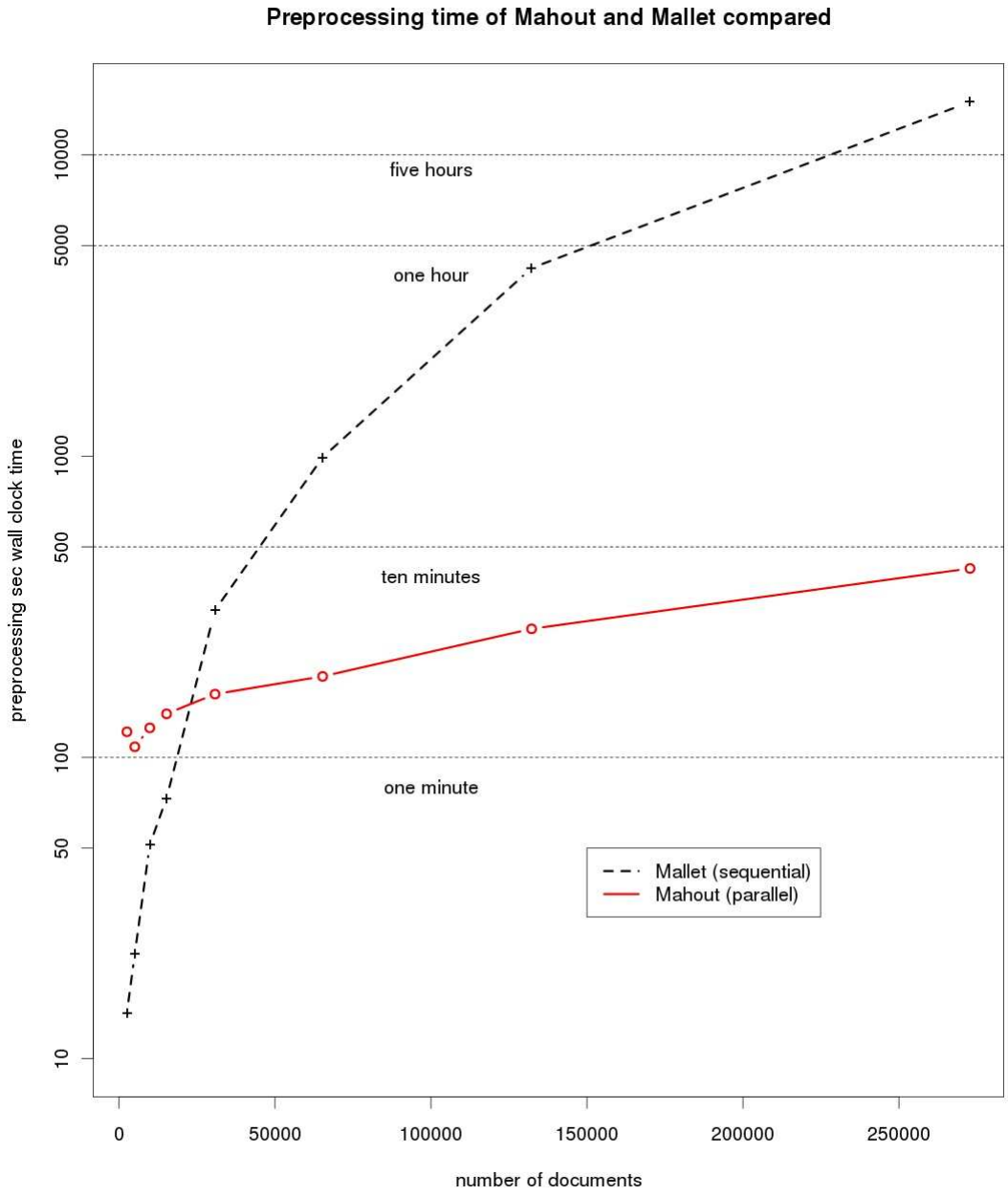


Figure 5: Log plot of pre-processing wall clock times

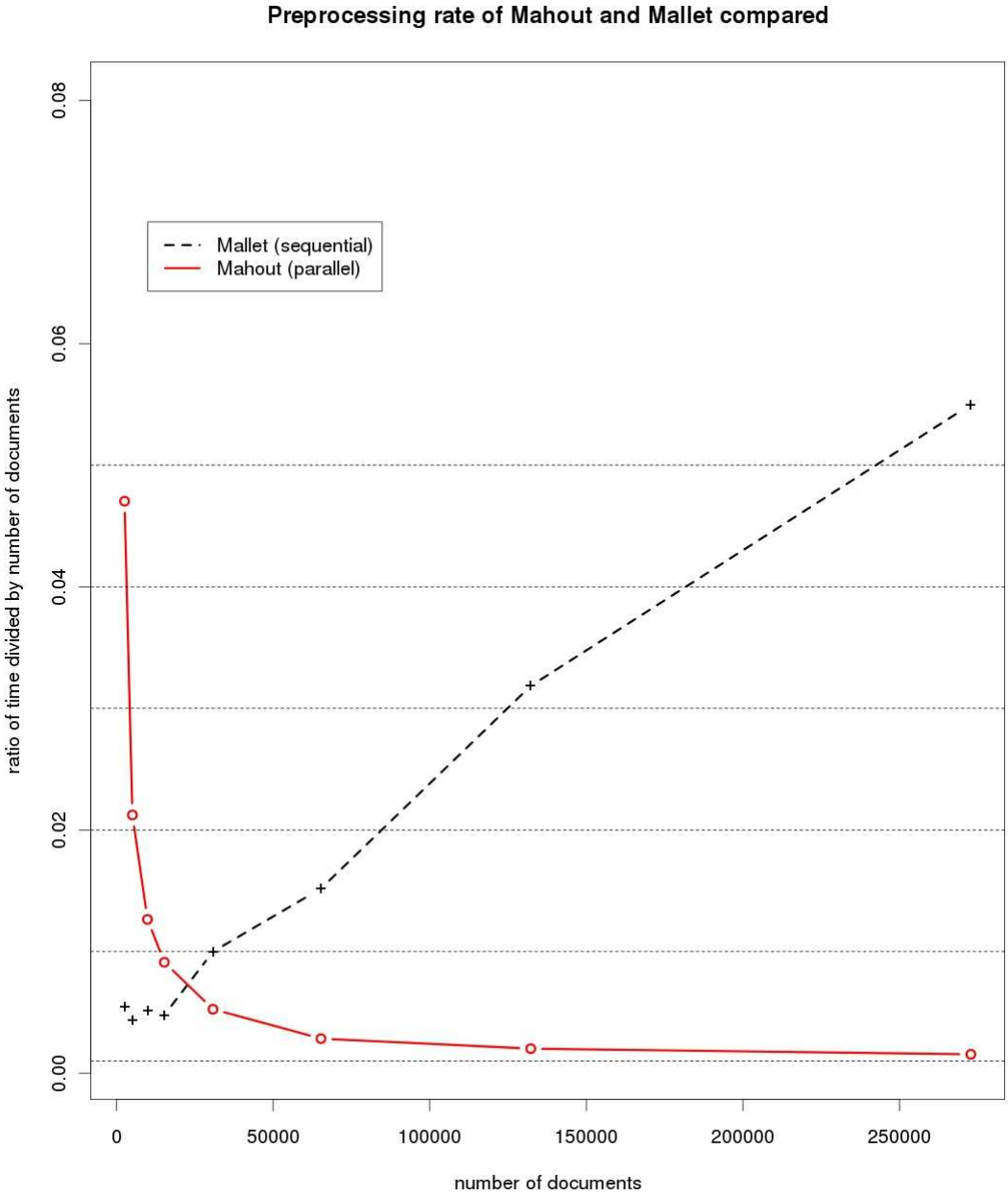


Figure 6: Pre-processing time per document

A.2 LDA clustering of documents

The LDA[54] algorithm was tested with the Map/Reduce implementation in Mahout, and with a sequential implementation in the Mallet toolbox.

Figure 7 illustrates the wall clock processing times for the LDA algorithm. As shown, the sequential implementation is generally an order of magnitude faster than Mahout. If we plot the fraction of required processing time to process one document, we observe in Figure 8 that the sequential case shows an increase. But the Map/Reduce case requires less processing

[54] Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3, 993–1022.

time per document with increasing collection size. The efficiency of Mahout however does not reach the one of the sequential case within realistic collection sizes.

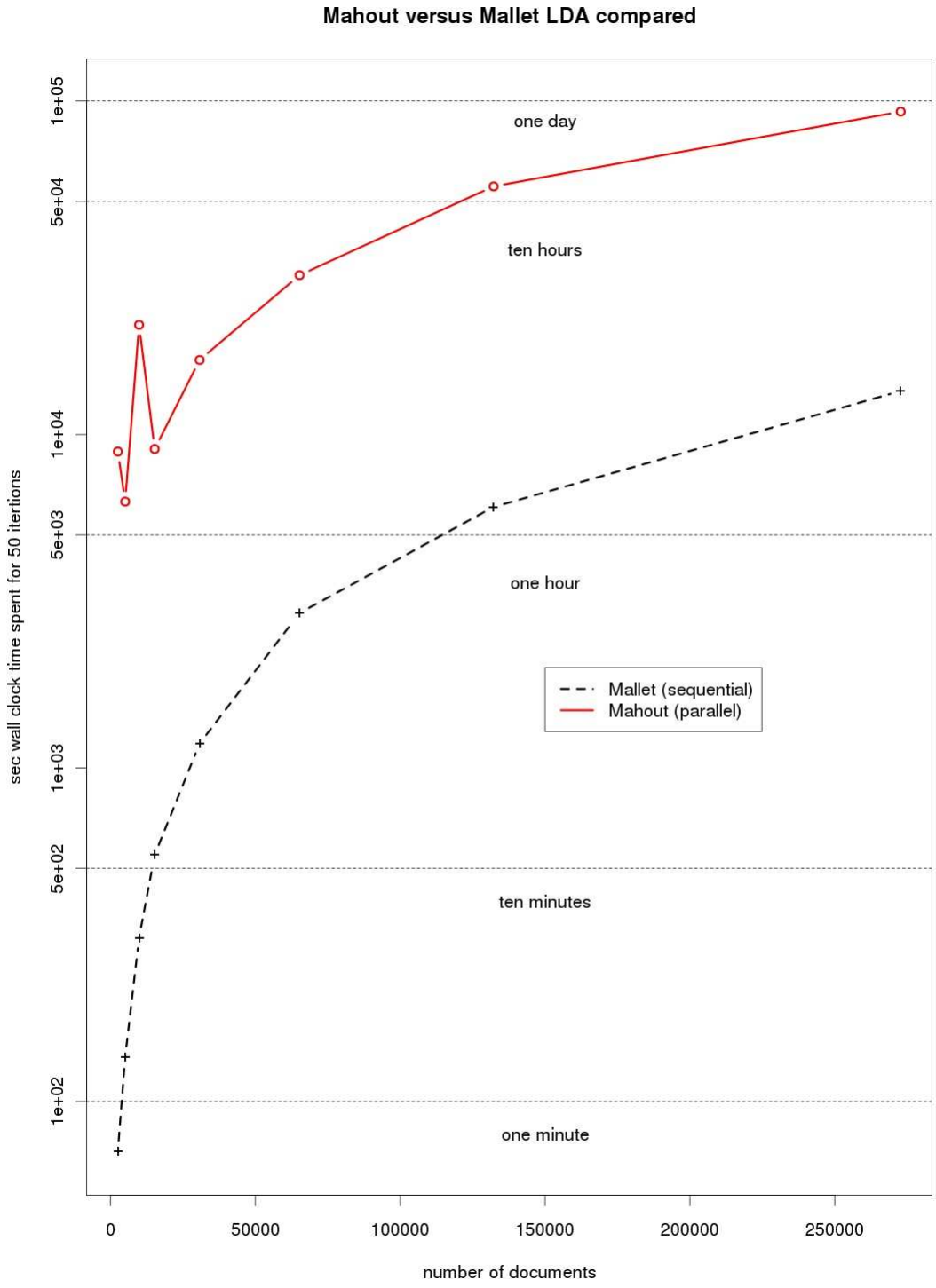


Figure 7: Log plot of LDA wall clock times

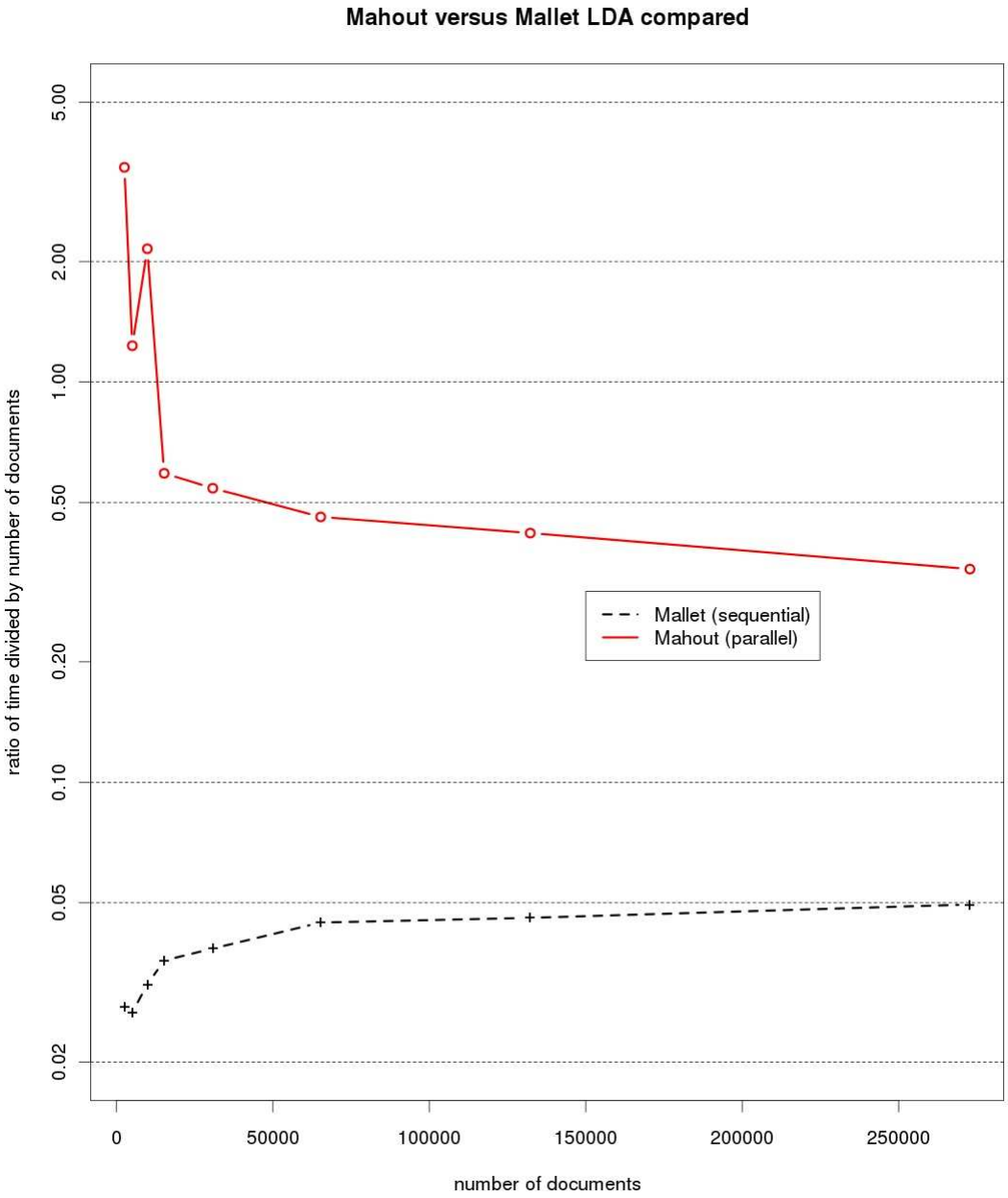


Figure 8: LDA processing time per document

The difference in performance of Mahout between pre-processing and the LDA algorithm can be explained in part by the differing computational complexity of the algorithms: pre-processing requires significantly less computations per document. Mahout seems to re-read the document from the HDFS distributed file system any time a computation is performed, and this feature is responsible for the large overhead in the LDA case. In the sequential case, the data are read only once from file, and are kept in memory during the whole computation. This of course imposes restrictions on the number of datasets that are related to the available memory. But the difference in processing speed between Mahout and Mallet would be advantageous for Mahout only if the size of the dataset would result in processing times in the order of days of wall-clock time. This, however, seems to be unrealistic for the Dicode use cases.

A.3 K-means clustering of documents.

The well known k-means algorithm was tested on a large collection of bag-of-words vectors computed from documents of the Apache mailing lists. The pre-processing that created the bag-of-words vectors was not a subject of the experiments. We tested the Mahout implementation of k-means against the Twister implementation.

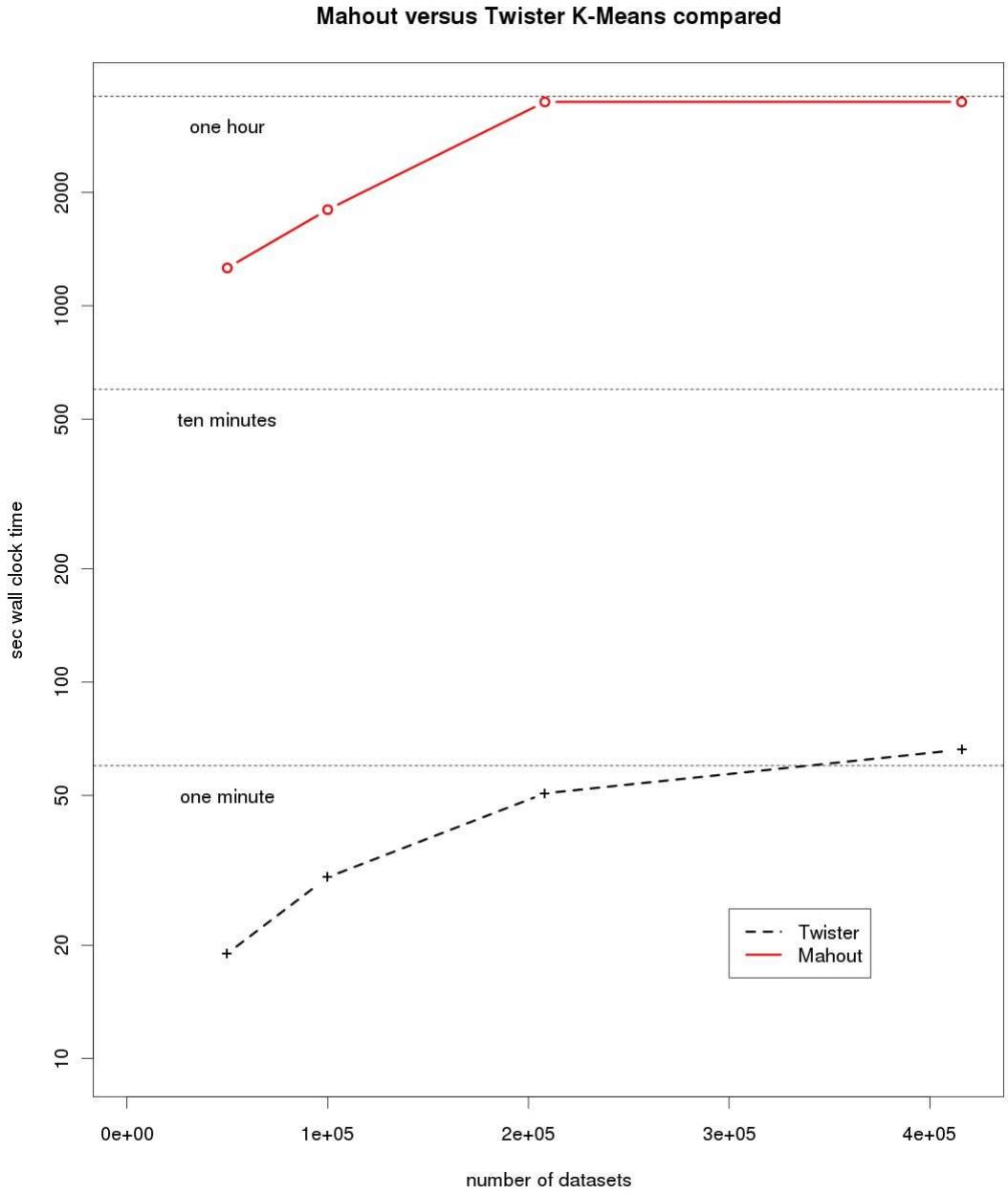


Figure 9: Log plot of k-means wall clock times

Both implementations are based on the Map/Reduce concept. They therefore show quite properties with respect to computing complexity, as can be seen in Figure 9, Figure 10, and Figure 11. Mahout is two orders of magnitude slower, however. Again the most reasonable explanation of the processing speed of the Twister variant is that it does not re-read datasets from a distributed file system during processing, as Mahout does.

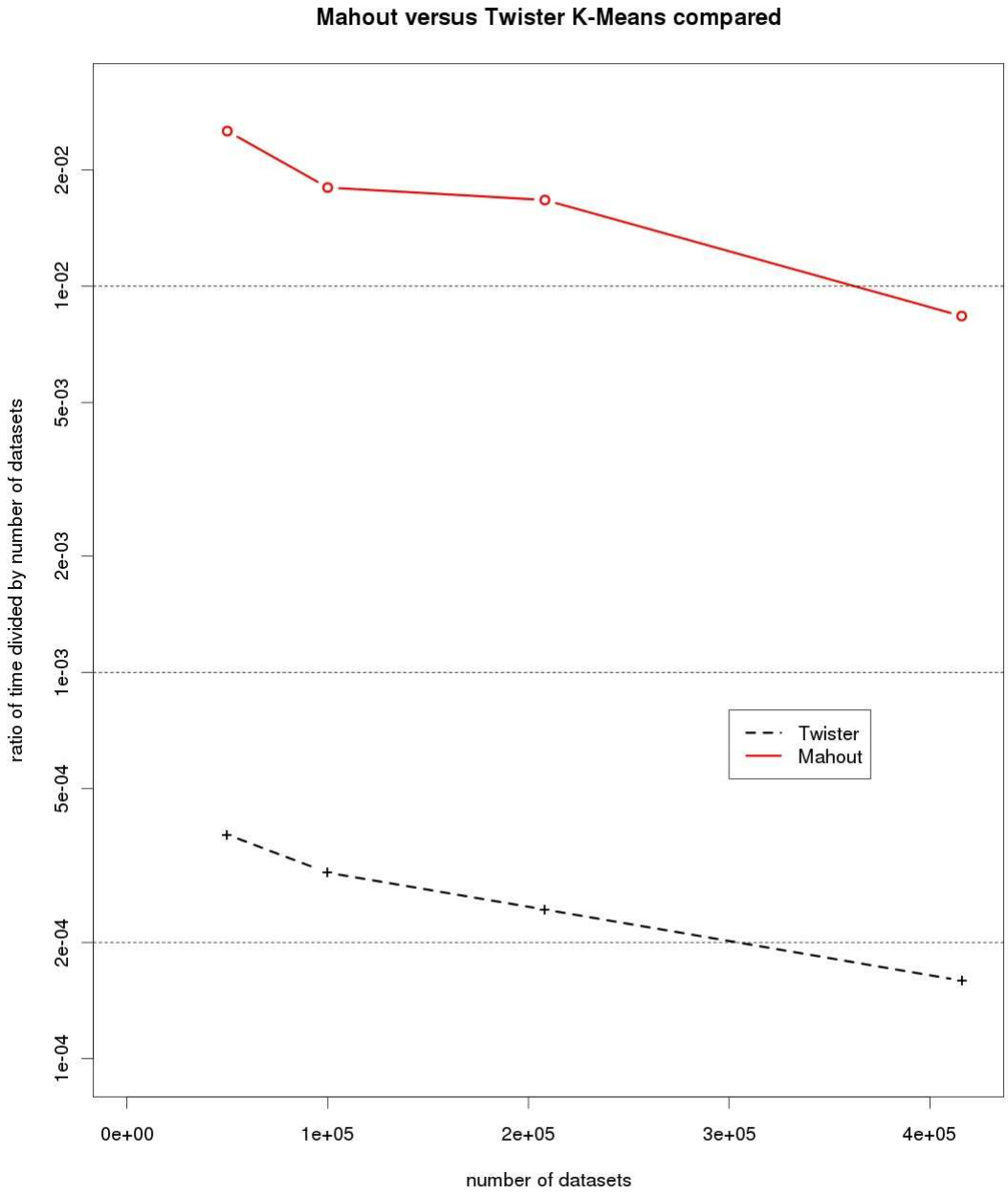


Figure 10: K-means processing time per dataset

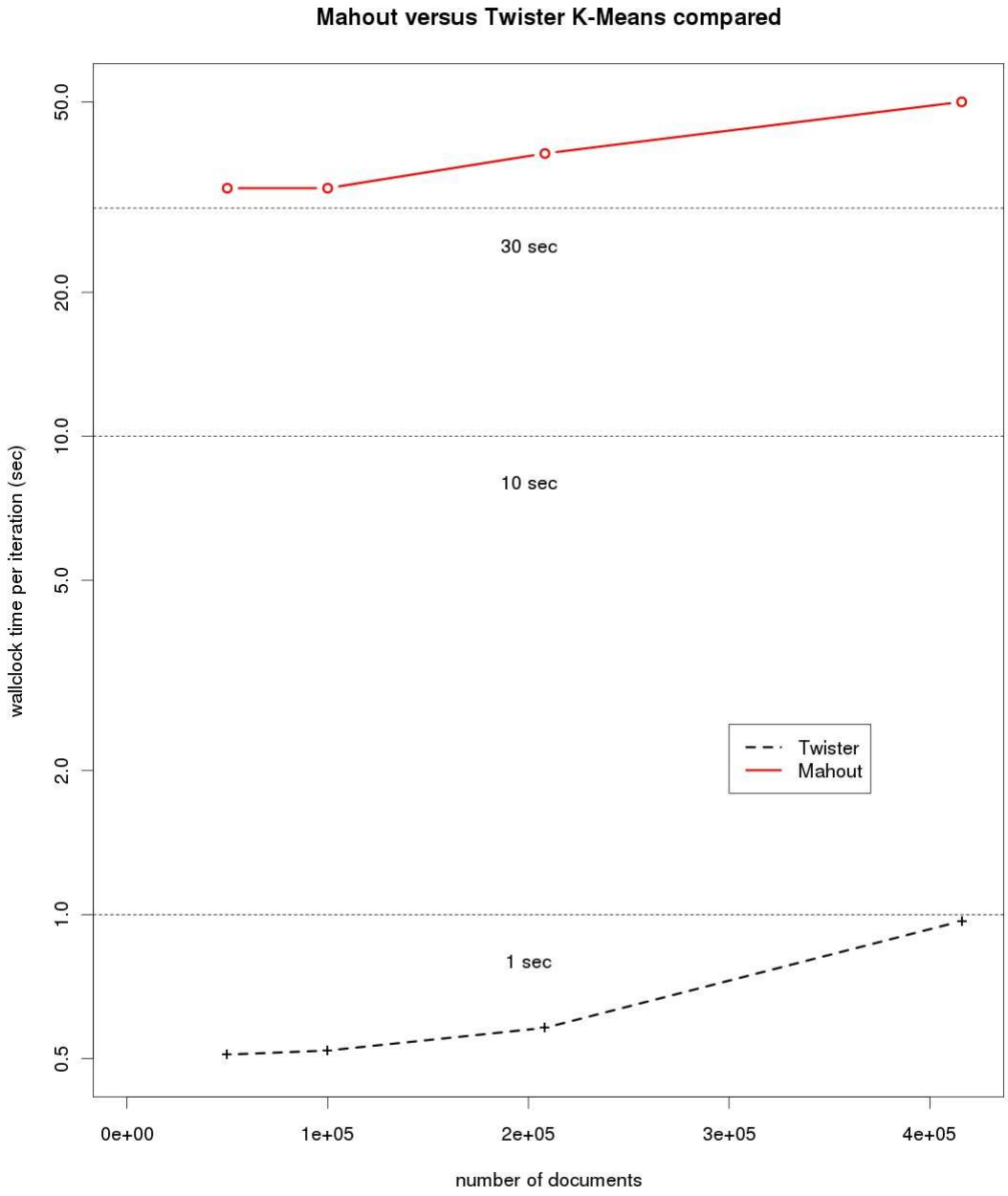


Figure 11: K-means processing time per iteration

A.4 GPU implementation of a Support Vector Machine Classifier

We tested the GPU implementation of a SVM classifier[55] that comes with the R package "gputools"[56] on moderately sized datasets. The datasets contained 40,000 to 100,000 vectors of dimension 1,000. The runtime of the GPU SVM was compared to that of SvmLight[57]. The speedup was approximately one order of magnitude. We did not run a large benchmark test, because the GPU version only has a radial basis kernel, which does not yield the best results for text mining purposes.

[55]Cristianini, N. and Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines.Cambridge University Press.
[56]http://brainarray.mbni.med.umich.edu/Brainarray/Rgpgpu/
[57]http://svmlight.joachims.org/

A.5 Using R in Taverna

An example R script for gene analysis has been split into components (see Figure 12). The dark green blocks are so-called Rshell that encapsulate R scripts. The RShell plug-in directly uses the R interpreter. It consists of a Taverna processor for R scripts and an RShell Session Manager that communicates with the R server. The RShell processor is configurable, allowing the user to define multiple inputs and outputs. Various data types are supported, such as strings, numeric data and images. To limit data transport between multiple RShell processors, the RShell plug-in supports persistent sessions.

Preliminary experience shows that splitting of a large script into components is feasible and works well and yields the same results as the original R script. However, the restricted number of types for input and output ports is sometimes a bottleneck. A notable drawback is that an Rshell cannot be reused by adding it to the list of available components. Ways to enable reuse need to be explored.

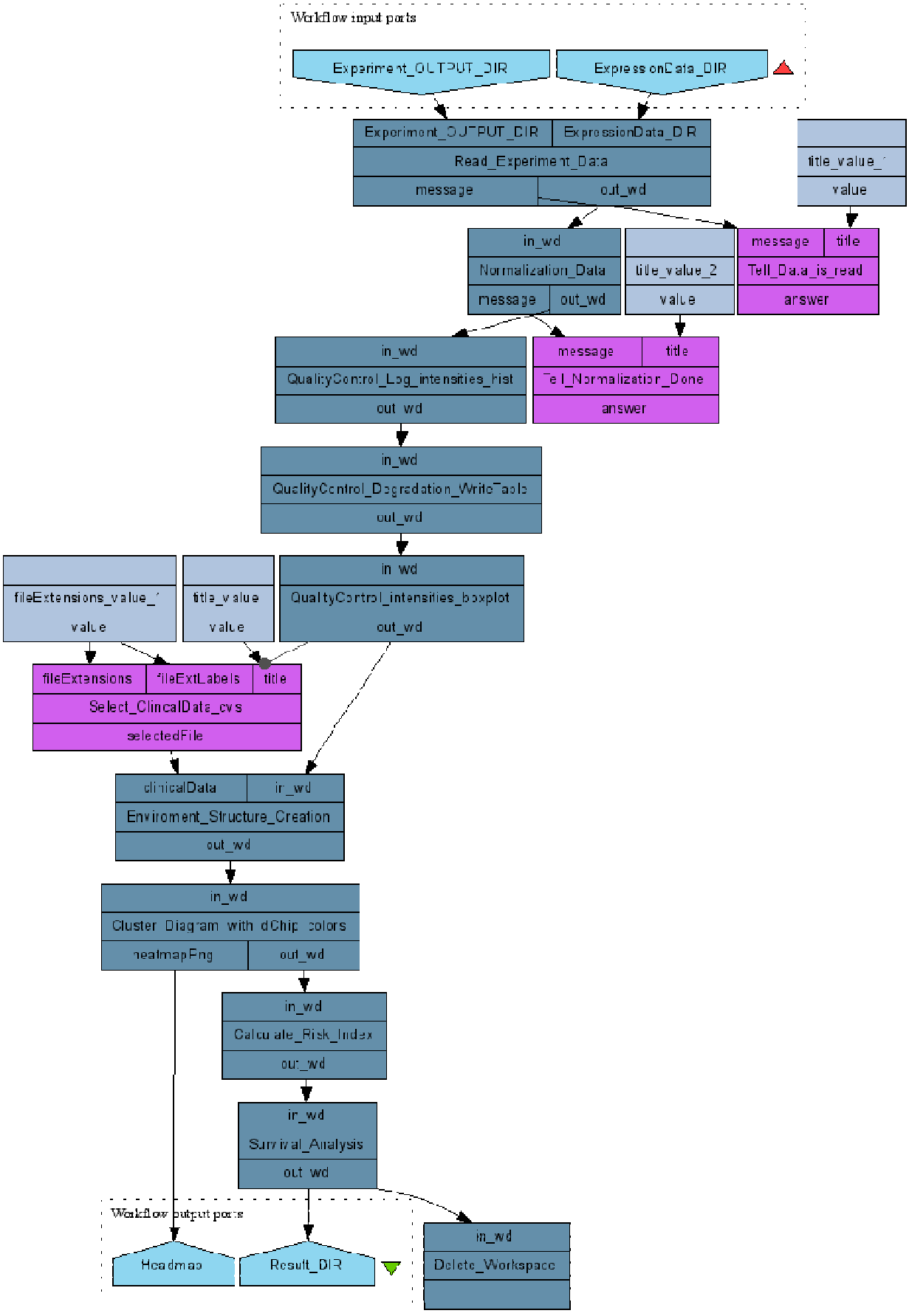


Figure 12: R script as Taverna workflow

Appendix B Service Specification

B.1Abstract Service Description

An abstract description consists of the description of the service interfaces and operations in textual form. The description may adhere to the following pattern:

Name	Name of the Service or Interface Type *Convention*: All individual words in the service type name are capitalised.
Standards	Reference to an abstract or a platform-specific service specification according to a standardisation organisation (e.g. ISO, CEN, W3C, OGC,...) or to reference material that has been taken into account when describing the service, its interfaces or operations.
Description	Description of the functionality provided by the service or interface. The service description shall end with: The <name> Service provides its functionality through the following interfaces: • *Interface*$_1$: Description of the purpose of interface 1 • … • *InterfaceN*: Description of the purpose of interface N *Note:* If an interface is re-used from another Service Type description, the name of this service type shall be indicated in brackets in the interface definition below.
Interface	*Interface*$_1$
oper_i	Description of the operation i of the interface. Only major input and output information shall be described, no individual request and result parameters. Optional operations are to be marked by suffix (optional) after the operation name.
...	...
Interface	*InterfaceN*
...	...
Example usage	Description of an example usage scenario of the service.
Comments	Description of current restrictions or possible extensions and enhancements in future versions.
Conformance classes	if defined
Implementation rules	Reference to the corresponding implementation rules if defined.

B.2Examples for Abstract Service Descriptions

Some typical examples of abstract service type descriptions relevant for Dicode are listed below. The list of service descriptions will be continually updated. UML specifications of these service types (and many others) are available (on demand) but not included. A repository of abstract service descriptions will be made available.

B.2.1 Base Notification Interface

Name	Base Notification Interface
Standards	Related standards are • *OASIS Web Services Base Notification 1.3* (http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf) • *OASIS Web Services Topics 1.3* (http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf) • *OASIS Web Services Brokered Notification 1.3* (http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf) • *OASIS Web Services Resource Properties 1.2*http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf) • *OMG Notification Service Specification* • (http://www.omg.org/cgi-bin/doc?formal/04-10-11.pdf)
Description	The Base Notification Interfaces are derived from the OASIS WS-BaseNotification Specification. According to OASIS standard "It defines the normative Web services interfaces for two of the important roles in the notification pattern, namely the NotificationProducer and NotificationConsumer roles. This specification includes standard message exchanges to be implemented by service providers that wish to act in these roles, along with operational requirements expected of them.
Interface	*NotificationConsumer*
notify	Implemented by the consumer and invoked by the producer when publishing a notification.
Interface	*NotificationProducer*
GetCurrentMessage	Upon invocation the last published Notification on a given topic is returned.
Interface	*PullPoint*
getMessages	The PullPoint interface allows accumulation of notifications by enabling the NotificationProducer to send notifications to the PullPoint. The getMessages operation enables the requestors to retrieve (or pull) notification messages from the pullPoint.
createPullPoint	Invoked by a requestor on an endpoint supporting the PullPoint interface in order to create a new PullPoint resource.
destroyPullPoint	Invoked by a requestor in order to destroy an existing PullPoint.
Interface	*PullPoint*
getMessages	The PullPoint interface allows accumulation of notifications by enabling the NotificationProducer to send notifications to the PullPoint. The getMessages operation enables the requestors to retrieve (or pull) notification messages from the pullPoint.
createPullPoint	Invoked by a requestor on an endpoint supporting the PullPoint interface in order to create a new PullPoint resource.
destroyPullPoint	Invoked by a requestor in order to destroy an existing PullPoint.
Interface	*PullPoint*
getMessages	The PullPoint interface allows accumulation of notifications by enabling the NotificationProducer to send notifications to the PullPoint. The getMessages operation enables the requestors to retrieve (or pull) notification messages from the pullPoint.
Example usage	A notification typically indicates a change within the "real world", e.g. computation of a service has terminated, sensors have been added or eliminated , or services are up or down in a computer network.
Comments	Notification, as specified above, is a direct exchange between two services.

B.2.2 Web Processing Service

Name	**Processing Service**
Standards	The functionality of the Processing Service is based on the WPS OGC draft implementation specifications: • *OGC 05-007r7 Web Processing Service (WPS), version 1.* The interface of the WPS is that of a general purpose Web Processing Service that provides client access to pre-programmed calculations and/or computation models operating on spatially referenced data. Access his provided by one generic 'execute' operation that initiates a process based on input parameters and an output definition.
Description	The Processing Service describes a common interface for services offering processing operations on spatial (vector as well as raster) and non-spatial data. It provides mechanisms to identify the data required by the calculation, initiate the calculation, and manage the output so that the client can access it. The Processing Service provides its functionality through the following interface: • *ServiceCapabilities*: Informs about common and specific capabilities. • *ProcessingService*: provides the means to get information on and to invoke a specific processing operation.
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about common and specific capabilities. Examples of specific capabilities are the supported processing operations (name and abstract).
Interface	*ProcessingService*
getProcessDescription	Allows to request and receive detailed information about one or more processing operation(s) that can be executed by the execute operation, including the input parameters and formats and the outputs.
execute	Allows to execute a specified processing operation implemented by the Processing Service, using provided values for input parameter values and returning the produced outputs.
Example usage	A client wants to create a niche model. The client queries the Processing Service for a description of available niche modelling operations (including its input and output types) using the *getProcessDescription* operation and then calls a particular niche modelling operation using the execute operation. The Processing Service returns the computed niche model either directly or as a reference (that can be used by the client to access the result).
Comments	A service instance that implements both the Processing Service and the Service Chain Access Service interface can be used to accomplish more complex task combining several operations where the workflow language is used for the combination and the Processing service for execution. An alternative architectural approach could be taken such that no Processing Service interface is described on the abstract level. Instead, the Service Meta Model would contain detailed rules about how processing service interfaces may be described by service providers. These descriptions should then include a process description, the input and output of the service and binding information, i.e. all information that is currently described in the Processing Services *getProcessDescription* operation. In both cases and for a common understanding of processing operations, (basic) operations should be grouped and described by an "operation taxonomy" to be referenced in the service specific capabilities.