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

D3.2.2 – The Dicode Data Mining Services (enhanced version)

Due date of deliverable:	31 August 2012
Actual submission date:	31 August 2012
Responsible Partner:	FHG, NEO
Contributing Partners:	FHG, NEO

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: Data Mining Services, Abstract Description

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

Computer Technology Institute & Press "Diophantus" (CTI) (coordinator), Greece

University of Leeds (UOL), UK

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

Universidad Politécnica 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	01-08-2012	First draft	Natalja Friesen, FHG
2	17-08-2012	Second draft (sent to internal reviewers)	Jörg Kindermann, FHG Doris Maassen, NEO
3	25-08-2012	Third draft (reviewers comments incorporated, sent to SC)	Natalja Friesen, FHG Jörg Kindermann, FHG Doris Maassen, NEO
4	29-08-2012	SC's comments incorporated	Natalja Friesen, FHG Jörg Kindermann, FHG Doris Maassen, NEO
5	31-08-2012	Final version (approved by SC, sent to the Project Officer)	Natalja Friesen, FHG

Deliverable manager

- Natalja Friesen, FHG

List of Contributors

- Jörg Kindermann, FHG
- Daniel Trabold, FHG
- Doris Maassen, NEO

List of Evaluators

- Fan Yang-Turner, UOL
- Georgia Tsiliki, BRF

Summary

This deliverable reports progress on the enhanced version of the Dicode data mining services that are being designed and developed in the context of WP3 and WP4. The document includes an updated description of the service functionalities, as well as a detailed report on modifications and extensions of each service.

Table of Contents

1 Introduction 5
1.1 Context 5
1.2 Objectives and structure 5

2 Architecture 5
2.1 Architecture of Text Mining Services 6
2.2 Architecture of Data Mining Services 6

3 Service Descriptions 7
3.1 Text Mining Services 8
3.1.1 Twitter Harvester 8
3.1.2 Twitter pre-processing service 8
3.1.3 Blog pre-processing service 9
3.1.4 Named Entity service 9
3.1.5 Entity Prominence service 11
3.1.6 Topic Detection Service 13
3.1.7 Phrase extraction (Sentiment Analysis, Opinion mining) 16
3.1.8 Phrase Extraction Training Service 18
3.2 Planned Text Mining Services 20
3.2.1 Emotion Detection 20
3.2.2 Relation Extraction 21
3.3 Data Mining Services 22
3.3.1 Subgroup Discovery Service 22
3.3.2 Subgroup Discovery for Genomic Data Analysis (Use case 1) 24
3.3.3 Recommender Service 27
3.3.4 Similarity Learning Service 28
3.3.5 Recommender for GEO Datasets 30
3.3.6 RapidMiner Service 32
3.3.7 Embedded R Executor 34

4 References 35

1 Introduction

1.1 Context

This deliverable describes the enhanced version of data mining services that have been designed and developed in the context of WP3 and WP4 of the Dicode project (development of data mining services in WP4 concerns the recommendation mechanism of Task 4.2). The services have been initially introduced in the deliverable D3.1.1 "Data Mining Framework". The initial abstract service description and their basic functionalities were demonstrated in deliverable D3.2.1 "The Dicode Data Mining Services (initial version)". More specifically, the document reports in detail on the modifications that were performed on each service and the current implementation status.

In particular, the initial versions of data mining services were extended as follows:

- Integration of data mining Toolkits;
- More precise adjustment of the services to the use cases;
- Integration of the services into the Dicode Workbench.

This is the second document in a series of three deliverables reporting on the progress in the development of data mining services. While the previous deliverable (D3.2.1) presented the abstract service description of the initial version, the focus of this deliverable is on their enhanced version. The final version of the data mining services will be documented in deliverable D3.2.3, due in month 33.

1.2 Objectives and structure

The purpose of this deliverable is to demonstrate the enhanced version of the data mining services that have been developed in the context of the WP3 and WP4 of the Dicode project.

The deliverable is structured as follows: In Section 2, we describe the architecture of the Dicode services. Section 3 presents an updated description of the services including text mining services and data mining services. It also gives an overview of new services to be developed in the third year of the project.

2 Architecture

The objectives of WP3 are concerned with the development of the Dicode's data mining infrastructure, which consists of services for pre-processing, clustering and classification, data mining and text mining. Task 3.1 of WP3 focuses on the design of a Data Mining Architecture according to the functional specifications outlined in D2.2 ("The Dicode approach: User requirements, conceptual integrative architecture, agile methodology and functional specifications") as follows:

- The services should cover a wide range of data mining tasks;
- The integrated services should be able to handle large data.

The architecture of the Dicode services consists of the two following parts:

- An architecture for Data Mining Services
- An architecture for Text Mining Services

2.1 Architecture of Text Mining Services

The text mining services are generally implemented as REST Web services. They run at servers hosted by Neofonie and Fraunhofer. Some of the services (the pre-processing services) do not need a user interface, because they are only triggered by other services and only provide output to other services. The user interfaces of the top-level services are integrated into the Dicode Workbench.

Text Mining Services without user interface currently are:

- Twitter Harvester
- Twitter pre-processing service
- Blog pre-processing service

Text Mining Services with user interface currently are:

- Named Entity service
- Entity Prominence service
- Topic detection service
- Phrase extraction service
- Phrase extraction training service
- Emotion detection service
- Relation extraction service

2.2 Architecture of Data Mining Services

Data Mining Services provide a variety of tools for knowledge creation or/and extraction. To select an appropriate platform for the Dicode data mining services we considered several frameworks that satisfied the Dicode requirements. Many well-known data mining algorithms for a broad set of application scenarios are freely available as open source implementation. Several frameworks combine many of them, such as Weka[1], RapidMiner[2], KNIME[3] and the R-project[4]. Amongst those frameworks RapidMiner is the most popular one (even more popular than any commercial product) according to a poll at KDNuggets.com[5] - a well-known and broadly trusted website amongst data miners.

The ability to analyze very large datasets with RapidMiner comes through a third party extension offered by Radoop[6], which is not available as open source. The RapidMiner platform in its current version is therefore limited to datasets that fit into the memory of a single Dicode server. Datasets of use case 1 fit well into memory. Datasets which do not fit into memory may need some pre-processing. The interesting subset for analysis is often much smaller than original data and may fit into memory even if the original data would be too large. Due to the fact that the RapidMiner satisfies all the requirements listed above, it was chosen as the Dicode data mining platform.

Data Mining services currently are:

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://rapid-i.com/
[3] http://www.knime.org/
[4] http://cran.r-project.org/
[5] http://www.kdnuggets.com/polls/2010/data-mining-analytics-tools.html
[6] http://rapid-i.com/component/option,com_myblog/show,Big-data-analytics-made-easy-Radoop.html/Itemid,172/

- Subgroup Discovery (SD)
- Recommender Service
- Similarity Learning Service
- RapidMiner Service
- Embedded R Executor

The instances of the data mining services with user interface currently are:

- Subgroup Discovery for Genomic Data Analysis (instance of SD service)
- Recommender of GEO Datasets (instance of Recommender and Similarity Learning services)

While the SimilarityLearning and Recommender services presented in D3.2.1 were always integrated into RapidMiner, the SubgroupDiscovery service has been ported to RapidMiner. The field of biomedical research (use case 1) prescribes some specific requirements to the analysis tools. Clinico-genomic research makes extensive use of R. The experts analyzing genomic data have built a wide range of custom libraries for R. Bioconductor[7] - one of the most popular open source and open development software for the analysis and comprehension of high-throughput genomic data - also uses the R statistical programming language. We therefore offer the R utility service as a second general data mining platform, which targets those Dicode users with a background in bioinformatics analysis. R offers an extra advantage. It can be used with Hadoop for the analysis of big data sets, which cannot be analyzed on a single machine. Hadoop Streaming[8], Hadoop Interactive[9], Rhipe[10] and Segue[11] are all packages which make R executable on Hadoop.

In distributed multi-user data analysis the collaboration aspect becomes prominent. The most important part in the analysis of genomic or medical data is the understanding of the results, since the outputs delivered by an algorithm are often not easy to interpret. The bottleneck is not the algorithms anymore, but rather the ability of a researcher to understand and to interpret the analysis results. The Dicode data mining services consist of algorithms whose results are suitable to be discussed in a collaboration environment. Instances of the data mining services have been implemented and integrated into the Dicode workbench to facilitate the analysis of data (e.g. genomic data in use case 1) and enable a group discussion of results. This is far more useful for the collaboration than faster algorithms. Through the integration of data mining services into the collaborative workspace, Dicode supports the common understanding of complex data.

All data mining services are offered as REST services hosted by Fraunhofer.

3 Service Descriptions

The great majority of details described in Deliverable D3.2.1 are still valid. To keep the document concise we will not present them again. Dicode's Named Entity Recognition Service which had been specified by FHG in D3.2.1, has been developed by NEO. The new features will be described subsequently. There are also some changes concerning the Twitter

[7] http://www.bioconductor.org/
[8] http://hadoop.apache.org/common/docs/r0.15.2/streaming.html
[9] http://hive.apache.org/
[10] http://www.rhipe.org
[11] http://code.google.com/p/segue/

Harvester, the Twitter pre-processing and the Blog pre-processing services which are presented in the following subsections.

3.1 Text Mining Services

3.1.1 Twitter Harvester

Twitter Harvester is Dicode's tweet acquisition component. Since the beginning of the Dicode project, Twitter's business model has been constantly changing towards a monetization of data. In D3.2.1, two services for Twitter data acquisition were described: Twitter Harvester and the Twitter pre-processing service. Dicode still provides those services, but the tweets are restricted to those available via Twitter's Streaming API, which returns about 1% of the tweets published worldwide. The configuration component for Twitters Search API which was suggested in D3.2.1 will not be developed, because there is no demand for tweets on special topics from Dicode's partners which cannot be met by a direct access to Twitter's Search API.

A project in need of a significant amount of tweets will have to use one of Twitter's commercial data providers. Twitter partners with data providers like Gnip and Datasift[12], which offer the complete Twitter stream and provide custom filters and Twitter pre-processing. We expect that in the future Social Media Analysts in need of Twitter analytics will receive their analysis either directly from Twitter or from specialised platforms which provide analysis of the complete Twitter stream. For Dicode's use cases, data sources like blogs, forums and business news are generally much more attractive than Twitter data.

As there is no need for a configuration component anymore, the Twitter Harvester does not provide a service interface but serves as the data storage layer which is accessed via the HBase API directly.

3.1.2 Twitter pre-processing service

The Twitter pre-processing service serves as an interface to Dicode's Twitter corpus. Due to Twitter's API restrictions, the corpus contains only recent Twitter data. Applications using this service should therefore fetch data regularly. The Twitter pre-processing service returns the corresponding Tweet ids and a condensed representation of Tweets containing only significant nouns. The user can fetch the original Tweet via Twitter's Search API. The query can be either a string or a regular expression. Additionally, the user can specify a language, a time period and a result limit.

The service can be used directly as a content service, which displays tweets for a given topic fetched from Twitter's search API. It can also be chained with another service, which takes the result as input for arbitrary text mining operations.

The interfaces of the Twitter pre-processing service will be provided as described in D3.2.1. Some technical changes have been made, due to changed requirements of the consuming services. For convenience, the output format has been changed from a compressed file to REST. The Parameter "minSupport" which defines the minimal occurrence of a term in the selected set of Tweets has been removed.

The API documentation appears at:

[12] https://dev.twitter.com/docs/twitter-data-providers

https://wiki.dicode-project.eu/display/DIC/Twitter+pre-processing+API

At the moment, Twitter pre-processing service is only used by the Keytrend Service which returns aggregated Twitter data. No other services based on the Twitter pre-processing service are currently planned.

3.1.3 Blog pre-processing service

The Blog pre-processing service acts as an interface to Dicode's weblog corpus, which contains weblog entries from English and German weblog of various domains. The API differs from the Twitter pre-processing API in just one point: besides significant nouns, the full text of a blog post is also returned. Currently, the Blog pre-processing service is only used for development, e.g. for the evaluation of Dicode's Named Entity service.

The API documentation can be found at:
https://wiki.dicode-project.eu/display/DIC/Blog+pre-processing+API

3.1.4 Named Entity service

Named Entity Recognition (NER) is the task of the identification and classification of proper names in natural-language text. Dicode's Named Entity service identifies named entities of the following types: PERSON, PLACE, ORGANISATION and WORK. In D3.2.1, the basic features of the Named Entity service have been described. It was mentioned that the context dependency is a big challenge in Named Entity Recognition: the entity "Germany" can be a location in a geographical context or an organization in political context. Named entity disambiguation is therefore performed based on the context of the analysed surface form. The quality of disambiguation usually increases with text size.

D3.2.1 proposed the use of conditional random fields (CRF) [5] based on extracted syntactical structures. In the enhanced version, we decided to use a different approach. Fraunhofer currently uses the CRF-based approach for the Phrase Extraction service. The Named Entity service is now developed by Neofonie and uses a combination of different techniques which have proven to provide good results. CRF are still used for the improvement of precision. Disambiguation is now performed based on external knowledge. The use of Wikipedia data for this purpose has been widely discussed in the literature [8]. The Named Entity Service was developed based on link statistics extracted from Wikipedia and NEO's Alexandria Ontology which uses Freebase[13]. We use a Hadoop-based Open Source component for Wikipedia extraction (called *pignlproc*) which was extended to meet Dicode's requirements.[14] Wikipedia statistics used for disambiguation include the probability of a link from a certain surface form to a Wikipedia article (e.g. how often "George Bush" either links to the father or to the son), how often a certain term in Wikipedia is showing up with a link or without a link, and how many links are referring to a Wikipedia article.[15]

[13] http://www.freebase.com/

[14] Dicode's contributions to *pignlproc* are available at the project's Github account : https://github.com/dicode-project/pignlproc. Wikipedia statistics is computed by the following pig script : https://github.com/dicode-project/pignlproc/blob/master/examples/nerd-stats/nerd-stats.pig

[15] Developers interested in using the statistics might have a look at Max Jacob's (NEO) talk at the Berlin Buzzwords conference 2012 which explains the extraction of Wikipedia statistics in detail: http://vimeo.com/45123391

The output of the Named Entity service can be used by the front-end developer directly - e.g. to visualize the document content by showing a list of named entities as a tag cloud. The service can also be used as input for higher level services, e.g. for topic detection.

Figure 3.1 shows an example visualization of a Named Entity list, which contains all Named Entities found in a news article. The Named Entity service is available for both English and German. The visualization is currently only available in German. It lists Persons ("Personen"), Places ("Orte") and Organisations ("Organisationen") separately. In the example, several persons and one organisation have been recognized.

Figure 3.1 : Named Entitiy List (Draft)

The abstract service description of the enhanced version of the "Named Entity Service" is presented in the table below.[16]

Name	Named Entity service
Standards	Not applicable.
Description	The Named Entity service returns disambiguated Named Entities for Dicode's document corpora. The Named Entity service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *NER*: provides the Named entity extraction functionality
Interface	*ServiceCapabilities*

[16] The API documentation appears at:
https://wiki.dicode-project.eu/display/DIC/Named+Entity+Service

getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*NER*
annotate	Provides annotated texts. Parameters: a collection (corpus) of full texts and the name of annotation category system. Output consists of texts with named entities annotated.
Example usage	Will be applied in use case 3 to identify persons, locations, brand names, etc. • Marketing analyst searches for a condensed representation of documents like presented in Figure 3.1 • Service developer develops text mining service which uses named entity annotations as input
Comments	Named Entity recognition is performed in batch mode. Typically, all incoming documents (e.g. from a Twitter or a news stream) will be processed in a scheduled interval. The results will then be made available via the Named Entity service. An interactive Named Entity service which takes an unknown document and instantly returns the Named Entities will not be developed. The service is currently used directly by the Entity Prominence service. Depending on the partner's needs, other document corpora will be annotated and made available via the Named Entity API. Recognition of brands has to be improved. Today NEO's ontology only contains brands which have been classified as organizations by Freebase. More brands will have to be added to the ontology.
Conformance classes	Not available.
Implementation rules	Not available.
Implementation status	Implemented.
UML model	Not available.

3.1.5 Entity Prominence service

The Entity prominence service returns statistics about the occurrence of Named Entities in news and blog documents within a certain time period (hour, day, week, month, year). The user can either query for a certain entity or retrieve the top entities for a time period. The user can filter by language, domain and entity type. The service builds on Dicode's Named Entity service.

Based on the service, prominence charts for certain entities can be produced. A chart of widgets might allow for brand comparison, source filtering and language filtering. Figure 3.2 shows an example of such visualization.

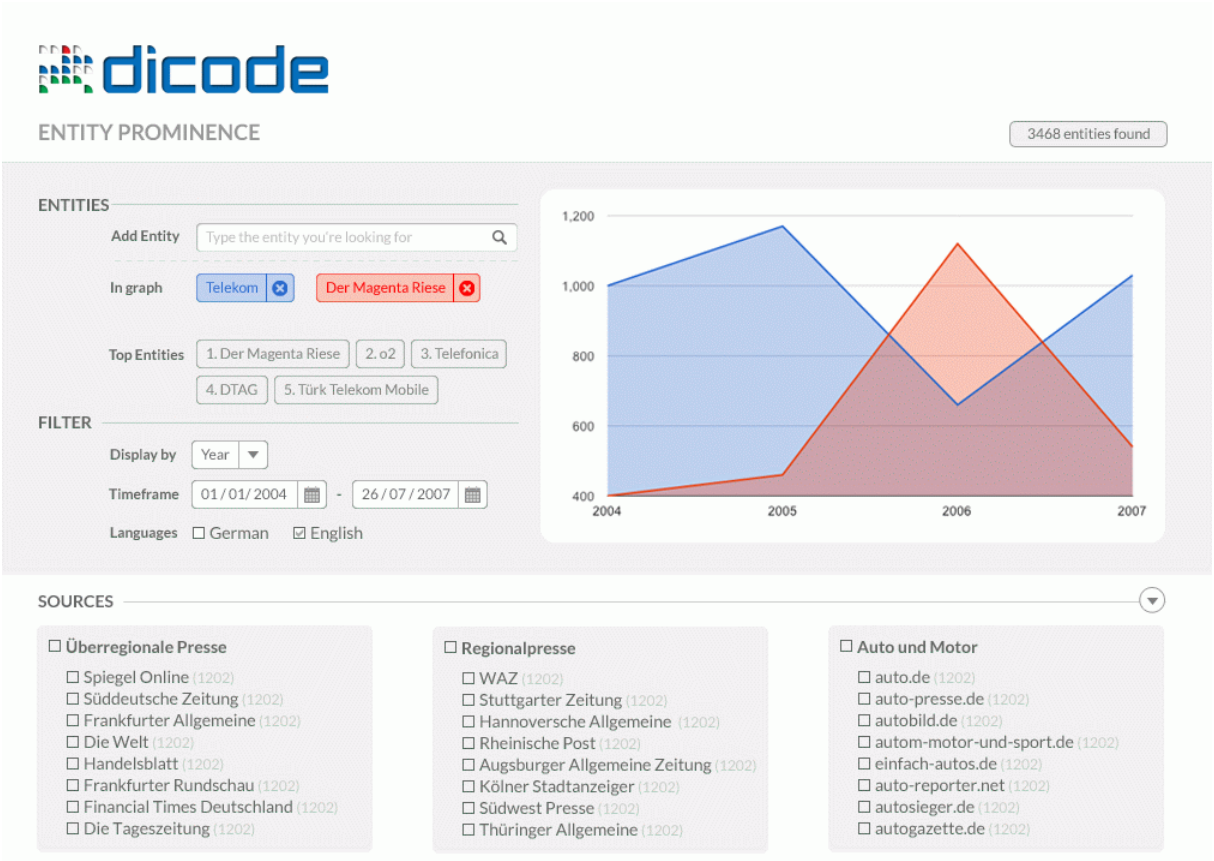

Figure 3.2 Visualization for Entity Prominence service (Draft)

Lists of top entities for a time unit (like "entity of the day") are another option. A widget might for example compare the prominence of political parties in different news sources. The top entities can also be used for the development of higher level services like event detection. For event detection, "unusual" top entities have to be detected. This could be done by monitoring the baseline of Named Entities which occur in news or social media documents. If an entity becomes popular, which did not show up before or which showed up only rarely in the past, this could indicate an event.

The number of brands available is still limited because the ontology used contains only those brands categorized as organizations in Freebase.

The abstract service description of the enhanced version of the "Entity Prominence Service" is presented in the table below.[17]

Name	Entity Prominence service
Standards	Not applicable.
Description	The Entity Prominence service can be used to query the "prominence" of Entities in documents and blogs over time. The Entity Prominence service

[17] The API documentation can be found at:
https://wiki.dicode-project.eu/display/DIC/Entity+Prominence+Service+API

	provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *Prominence:* Provides the prominence counts
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*Prominence*
Prominence	Provides the prominence counts for a given Named entity and time unit (hour, day, week, month, year). Filters for Language (de, en) and domain (like bbc.co.uk) can be applied. The number of data points counting back from current time has to be specified. Optionally, a *startTime* and/or an *endTime* can be defined.)
TopEntities	Returns top entities for a time unit. Additional parameters are domain and language.
Example usage	The Entity Prominence service will be applied in use case 3 for brand monitoring: • Marketing analyst monitors prominence of a brand over time • Analyst requests alerts about the entity of the day
Comments	
Conformance classes	Not available.
Implementation rules	Not available.
Implementation status	API available for evaluation. Entity counts for ca. 60 million news documents for German and English news sources. Visualizaton and Workspace integration are in preparation.
UML model	Not available.

3.1.6 Topic Detection Service

An integration of the Topic Detection Service into the Dicode workbench, including a visualization of the results, is planned for year 3. Here we give a preview on the new features. The results that are displayed here have been obtained from a collection of tweets[18]. Figure 3.3 shows a topic map that was created from the tweets collection. The numbered blue nodes in the graph represent the different topics that have been extracted. The nodes that are labelled by words represent those terms that are most important for a given topic. The network structure of the graph emerges, because some of the defining words are related to several topic nodes (words that are related to one topic only are coloured in yellow, those that are related to several topics are coloured in orange or red, depending on the number of topic relations.). The visualization provides a quick overview of the topics that are present in a text collection as well as their interrelations. Users will also be able to zoom in on a graph

[18] http://www.sananalytics.com/lab/twitter-sentiment/

detail as it is shown in Figure 3.4. Here the neighbourhood of the term “ballmer” (i.e. the Microsoft CEO Steve Ballmer) is shown. It is related to the three topics 13, 31, and 46.

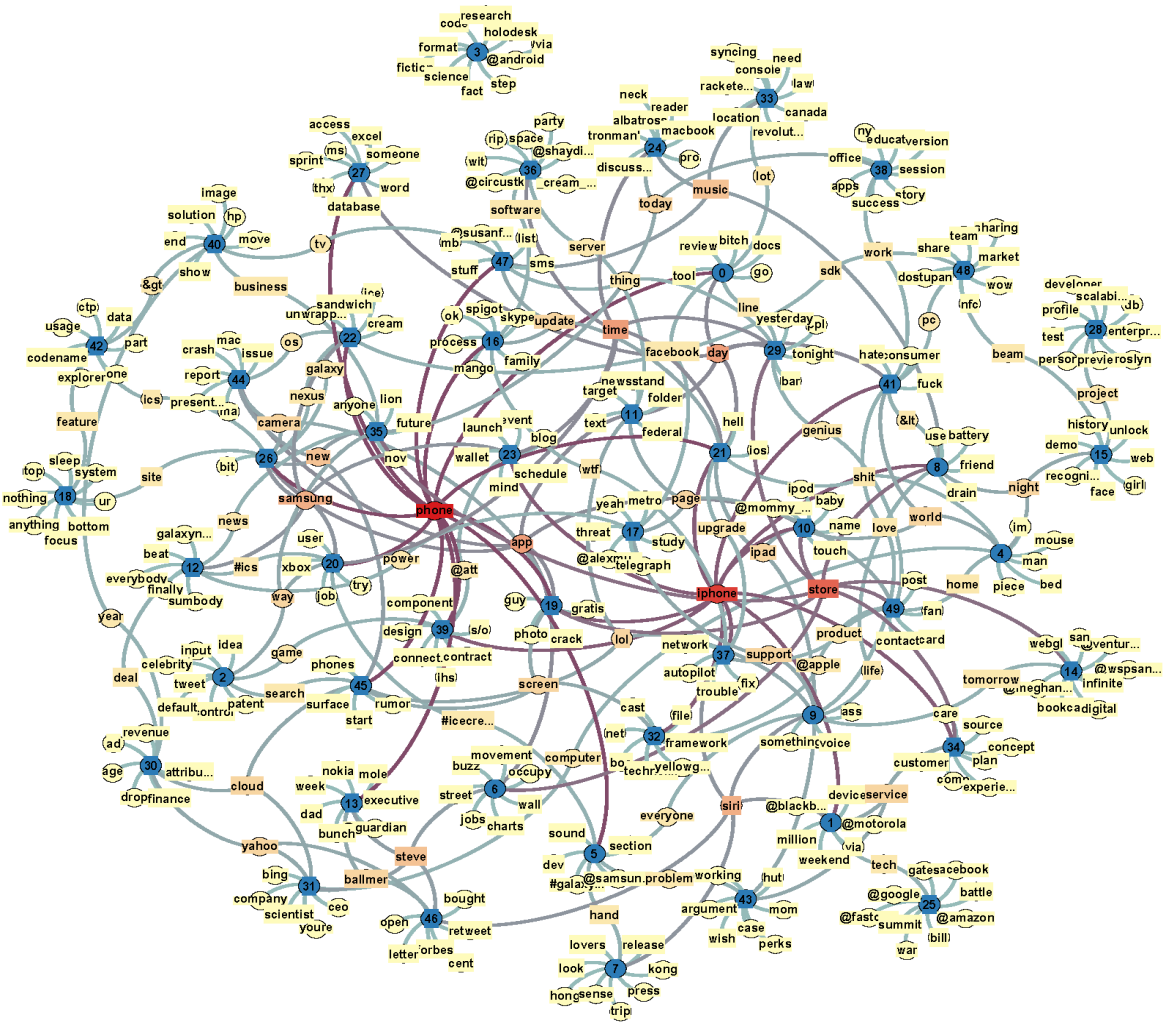

Figure 3.3 Topic map of a tweet collection

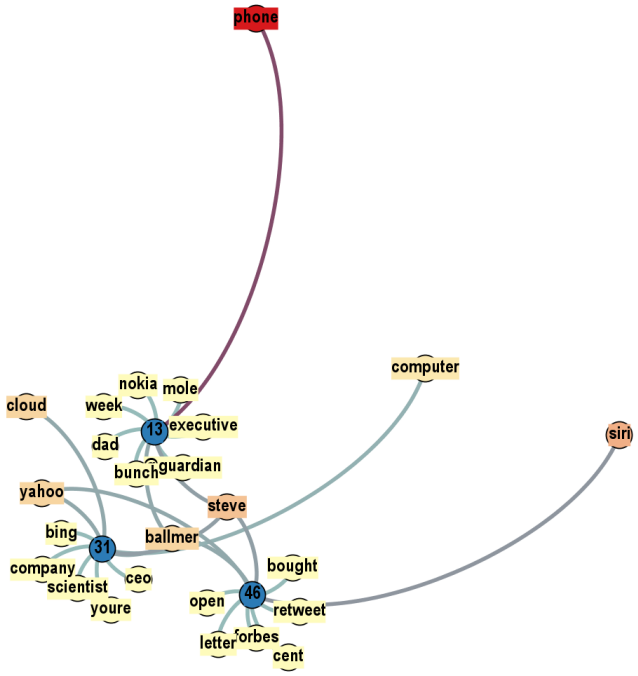

Figure 3.4 Topic map detail

The Topic Detection service has been slightly modified to allow for more intuitive use. The abstract service description of the enhanced version of the "Topic Detection Service" is presented in the table below.

Name	Topic Detection Service
Standards	Term-frequency vectors (as input)
Description	The Dicode Topic Detection Service gives the user a quick albeit superficial overview of the thematic content of a document collection. The Topic Detection Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *Detection:* Provides the topics
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*Detection*
provideTopics	Provides *n* topics as described by a list of *m* keywords per topic. Parameters: number of topics, number of keywords per topic, term-frequency vectors that have been computed from the original input texts, list of stop-words that will be removed from the term frequency vectors (optional) The workbench interface will be simplified: the user can choose between "granularity" values "fine" and "coarse". Depending on the choice, the

	parameters *m* and *n* will be computed. Output is a comma separated value table of topics. Each topic corresponds to a row in the table. Each row consists of a list of keywords sorted by keyword importance. The workbench interface will also output a "topic map" visualization.
Example usage	Topic detection will be applied in Use Case 3 to provide an overview of the thematic content of a text or tweet collection. A second application is possibly topic detection in bio-medical texts with regard to Use Cases 1 and 2
Comments	• The input of term frequency vectors is obtained from the Blog or Twitter pre-processing services • Weights (probabilities) may be added to the keywords.
Conformance classes	Not available.
Implementation rules	Not available.
Implementation status	Prototypical version of the algorithm is implemented. Not yet provided as workbench integration. Visualization not yet finished.
UML model	Not available.

3.1.7 Phrase extraction (Sentiment Analysis, Opinion mining)

The services of Sentiment Analysis and Opinion mining that were described in deliverable 3.2.1 have been subsumed under the new service Phrase Extraction. This service is a more generic one that allows extracting different types of phrases from a text collection. It is based on the Conditional Random Field algorithm [5], which has already been described in section 4.4.2 of D2.1 ("Dealing with data intensiveness and cognitive complexity in contemporary collaboration and decision making settings: A state of the art survey"). The service uses an active learning strategy that is an extension of the one developed by FHG in 2008 [6].

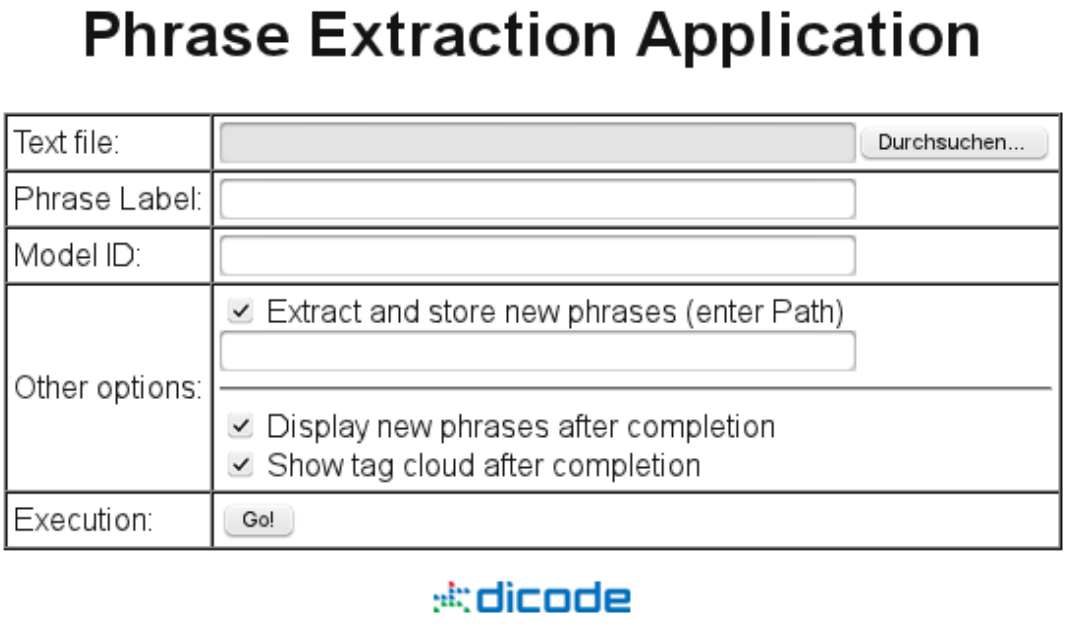

Figure 3.5 Dicode Workbench interface of the phrase extraction application service

In the first version (which is currently ready), the service will rely on pre-trained models the can extract positive and negative phrases from domain-specific text collections. The second version will be implemented in year 3. It will support an interactive workflow, where the end-user can train phrase extraction models interactively and apply them to a text collection.

I purchased this Toshiba Satellite laptop approximately one month ago and I love it. It does everything I want it to do and more. The display on the monitor is gorgeous. The keyboard is the perfect size and I absolutely love the number pad. I always said that if they would just put a number pad on a laptop that I would buy it. So I Did!

Figure 3.6 Phrase highlighting in a text segment.

The Dicode workbench interface (Figure 3.5) provides two visualizations of the results: highlighted phrases in the texts (Figure 3.6) and a tag cloud that is generated from the full set of phrases that have been extracted from the whole text collection (Figure 3.7). These examples have been generated using a collection of Amazon customer reviews on laptops.

Figure 3.7 A tag cloud generated from a set of extracted phrases

The abstract service description of the current version of the "Phrase Extraction Service" is presented in the table below.

Name	Phrase Extraction Service
Standards	No relevant standards available
Description	The Dicode Phrase Extraction Service identifies text passages that are of a

	specific interest. This includes sentiments, and also indicates the nature (positive versus negative or further categories) of the sentiment. Other phrases, that express opinions may also be extracted, depending on the extraction model that is used. The Phrase Extraction Application Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *Phrase Extraction*: provides the analysis functionality
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*Phrase Extraction Application*
annotate	Provides annotated texts. Inputs are text corpora and extraction models. A selection of models is provided. It is also possible to train new models with the Dicode Phrase Extraction Training Service. Output is the same as input but with phrases annotated. A second output is a tag cloud that is generated from the extracted phrases.
Example usage	Will be applied in use case 3 to identify sentiments and opinions expressed in the texts
Comments	• Phrase Extraction service will be applied to Twitter tweets as text corpora. • The Phrase Extraction Service will eventually use the output of the Named Entity Recognition Service.
Conformance classes	Not available.
Implementation rules	Not available.
Implementation status	First version ready
UML model	Not available.

3.1.8 Phrase Extraction Training Service

The Phrase Extraction Training Service will be integrated into the Dicode Workbench in year 3, using the interface template that is displayed in Figure 3.8. It enables the user to interactively develop and test new phrase extraction models. These models can then be used in the Phrase Extraction service described in section 3.1.7.

The abstract service description of the current version of the "Phrase Extraction Training Service" is presented in the table below.

Name	Phrase Extraction Training Service
Standards	No relevant standards available
Description	The Dicode Phrase Extraction Training Service supports an interactive workflow that enables the user to build his own phrase extraction models. It is based on a text collection and a list of specific phrase examples that the user provides. These phrases must occur anywhere in the text collection literally. The Phrase Extraction Training Service generalizes the phrases and extracts other, similar phrases from the text collection. It provides a list of newly extracted phrases which are then manually corrected by the user – i.e. the user may delete unwanted phrases. After that a second training run with the corrected phrase list is done on the text collection. The Phrase Extraction Training Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *Phrase Extraction*: provides the analysis functionality
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*Phrase Extraction Training*
train	Provides extracted phrases and an extraction model (binary file). Input is a text collection and a phrase list. A list of extracted phrases and an extraction model is provided by the service.
Example usage	Will be applied in use case 3 to identify sentiments and opinions expressed in the texts
Comments	• The Phrase Extraction Training service will be applied to Twitter tweets as text corpora. • The Phrase Extraction Service will eventually use the output of the Named Entity Service.
Conformance classes	Not available.
Implementation rules	Not available.
Implementation status	First version ready
UML model	Not available.

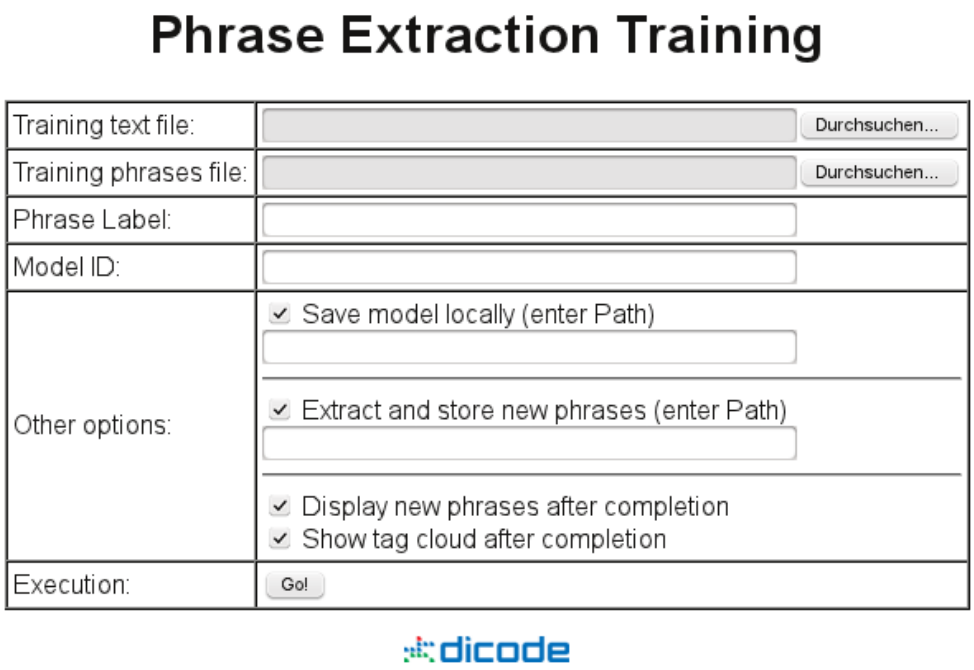

Figure 3.8 Dicode Workbench interface of the phrase extraction training service

3.2 Planned Text Mining Services

Planned text mining services include the Emotion Detection Service and the Relation Extraction Service, which will be implemented in year 3. The Emotion Detection Service will help the analyst to generate a survey of consumer needs and wishes, based on their personal comments on a product. The Relation Extraction service can be used to extract certain aspects of consumer's opinions towards a product, i.e. analyse the "relations" a consumer has towards a product.

The abstract service description of the "Emotion Detection Service" is presented in the table below.

3.2.1 Emotion Detection

Name	Emotion Detection Service
Standards	No relevant standards available
Description	The Dicode Emotion Detection Service The Emotion Detection Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *EmotionDetection*: provides the emotion detection functionality
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*EmotionDetection*

annotate	Provides annotated texts Inputs are text corpora. Output is the same as input but with emotions annotated
Example usage	Will be applied in use case 3 to detect emotive texts.
Comments	• Emotion detection will be applied to Twitter tweets as text corpora. • The format of text annotation needs to be determined. • The service may use ontologies such as sentiWordNet. • The Emotion Detection Service will eventually use the output of the Named Entity Recognition Service and the Sentiment Analysis Service.
Conformance classes	Not yet available.
Implementation rules	Not yet available.
Implementation status	Not yet implemented
UML model	Not yet available.

3.2.2 Relation Extraction

The abstract service description of the Relation Extraction Service" is presented in the table below.

Name	Relation Extraction Service
Standards	No relevant standards available
Description	The Dicode Relation Extraction Service detects certain prototypical relations between individuals, products, and brands that are related to emotions. The Relation Extraction Service provides its functionality through the following interfaces: • *ServiceCapabilities*: Informs about the common and specific capabilities. • *RelationExtraction*: provides the relation extraction functionality
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Interface	*RelationExtraction*
annotate	Provides annotated texts

	Inputs are text corpora and relation type(s) to be detected.
Example usage	Will be applied in use case 3 to detect certain standard relations
Comments	• Relation extraction will be applied to Twitter tweets as text corpora. • The format of text annotation needs to be determined. • The service may use ontologies such as sentiWordNet. The Relation Extraction Service will eventually use the output of the Named Entity Recognition, Sentiment Analysis, and Emotion Detection Services.
Conformance classes	Not yet available.
Implementation rules	Not yet available.
Implementation status	Not yet implemented
UML model	Not yet available.

3.3 Data Mining Services

The data mining services include algorithms addressing a wide range of analysis tasks, from standard data mining problem, such as classification and regression, up to more difficult tasks like local pattern mining or similarity learning. The next subsections present the two generic services: the Subgroup Discovery Service and the Recommender Service. These services can be easily adapted for more specific tasks. We demonstrate this through examples showing the application of the Subgroup Discovery Service for Genomic Data Analysis and the application of the Recommender Service for GEO Datasets.

3.3.1 Subgroup Discovery Service

Deliverable D3.2.1 presented a Subgroup Discovery (SD) service as a standalone service for finding new, previously unknown local patterns (subgroups) in the given dataset.

Subgroup discovery is a Knowledge Discovery task that aims at finding subgroups of a population with high generality and distributional unusualness with respect to the target attribute. A subgroup is a set of terms $\{t_1, \ldots, t_k\}$ where every term t_i is a constraint on an attribute, i.e. t_i has the form $(a_i = v_i)$, v_i in $D(a_i)$, where $D(a_i)$ is a domain of the attribute a_i. The length of the subgroup description is the number of terms it is built of.

SD is a method which is often used to generate a human understandable representation of the most interesting dependencies in the data. Hence, the more crisp and concise the output is, the better. Unfortunately, standard algorithms often produce very large and redundant outputs. It is hard for a researcher to make use of the results, particularly for large and complex data. In order to reduce the output space, we extended the SD service by a component for finding relevant subgroups. The newly developed approach is described in Grosskreuz at al. [7]. This modification leads to a considerable reduction in the amount of returned patterns without loss of their statistical descriptiveness, and as a consequence, to a better understanding of the data. Another important issue concerning interpretability of the results is to enable a user to influence the output by including/excluding certain attributes

from the search. Receiving a feedback on their results, the user can set up a new iteration of the algorithm by specifying undesired (e.g. biological or medical) attributes. We extended the Subgroup Discovery Service by a component that attends this task.

An updated description of the service is presented in the table below.

Name	Subgroup Discovery Service
Standards	see D3.2.1
Description	The Dicode Subgroup Discovery Service considers a given target variable and aims to detect novel knowledge given by subgroup patterns with regard to this property of interest. The algorithm returns the most relevant subgroups. The Subgroup Discovery Service provides its functionality through the following interfaces: • ServiceCapabilities: Informs about the common and specific capabilities. • Discovery: Generates the subgroup patterns. • NotificationProducer: Allows for subscription of notification consumers.
Interface	*ServiceCapabilities*
getCapabilities	see D3.2.1
Interface	see D3.2.1
Interface	*NotificationProducer*
subscribe	see D3.2.1
getCurrent-Message	see D3.2.1
Comments	• The present algorithm is a variant of the well-known FP-Growth algorithms; • The algorithm has been extended by a method for finding relevant subgroups, which significantly reduces the output space.
Implementation rules	Not available.
Implementation	The algorithm is provided as a REST service

To sum up, we offer a SD service as a standard method for data analysis. We developed a new mechanism for finding relevant subgroups, which is important to get non-redundant results. We extended the service in order to be able to include/exclude particular terms from the search.

The presented SD service is generic enough to be applied in a wide range of research fields. Despite its generality, the service can be easily targeted to more specific tasks, such as analysis of genomic data. Through the integration into the collaboration workbench (i.e. collaboration workspace, see D4.1.2), the service enables the user to benefit from decision support mechanisms, provided by the Dicode platform. We demonstrate this in the example

of a Subgroup Discovery Service for multi-platform Genomic Data Analysis, which is an instance of the SD service.

3.3.2 Subgroup Discovery for Genomic Data Analysis (Use case 1)

Applied to gene expression data, the standard SD service would deliver a set of gene names that share similar properties relative to the research question of interest, i.e. SD uses the filtered dataset as produced by the statistical methodology used. The translation of these results into useful biological knowledge still remains a necessary validation procedure, which is often time-consuming. For instance, one might wonder how the set of genes can be described in terms of molecular or cellular function. Knowledge databases such as Gene Ontology[19] (GO) or Kyoto Encyclopedia of Genes and Genomes[20] (KEGG) serve as an excellent basis for the interpretation of genes. The integration of these additional knowledge databases into the Subgroup Discovery Service would provide the researcher with more meaningful results.

Another important issue is to enable the user to share their results with other researchers that are working on similar problems and to discuss weaknesses of the identified patterns. Taking these issues into account, the enhanced version of SD service includes the following extensions:

- Integration of external knowledge databases, such as Gene Ontology
- Adaptation of the SD Service to the task of functional interpretation of gene sets
- Development of a user friendly interface
- Integration into the collaborative workbench

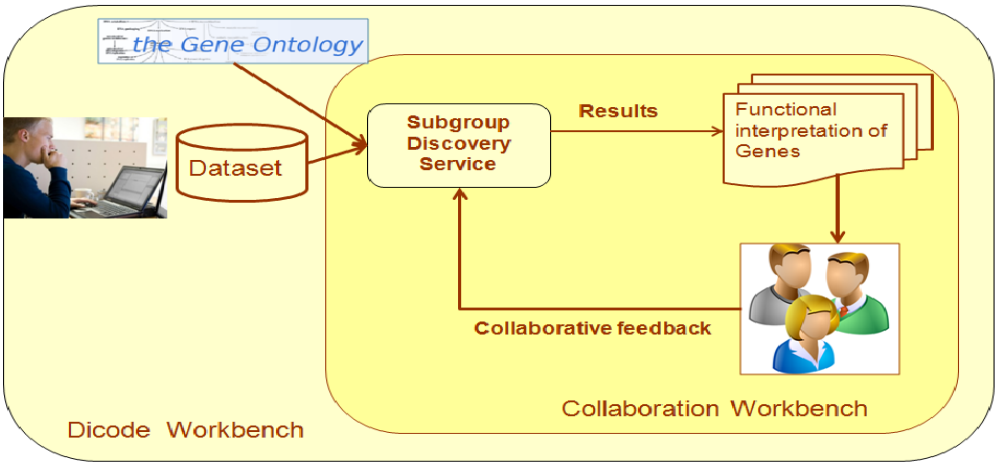

Figure 3.9 Interactive SD service for functional interpretation of genomic data

Figure 3.9 sketches a scenario of using the extended SD service in the Dicode Workbench as an interactive process. First, the gene sets identified by the user are automatically enriched with Gene Ontology (GO) terms. Next, the algorithm finds the most interesting and relevant subgroups that describe differentially expressed genes. Each discovered subgroup is presented as one piece of knowledge in the collaboration workbench and can be discussed by experts in the overall discussion process. According to the feedback, the user can set up a new iteration of the algorithm by specifying undesired attributes. Below, we describe the

[19] http://www.geneontology.org/
[20] http://www.genome.jp/kegg/

performed changes on the Subgroup Discovery Service in more detail. The integration of KEGG is planned to take place in year 3.

Next, we describe briefly the abovementioned extensions

- **Integration of external knowledge databases**
  Gene Ontology (GO) serves as a controlled vocabulary of terms for describing genes according to several aspects. GO includes three ontologies containing the description of molecular functions, biological processes and cellular locations of any gene product, respectively. Within each of these ontologies, the terms are organised in a hierarchical way, according to parent-child relationships in a directed acyclic graph (DAG). This allows a progressive functional description, matching the current level of experimental characterization of the corresponding gene product.

  Currently, the three components of the GO are integrated in the service. Moreover, we aim to include further knowledge databases in the future, such as the KEGG or Reactom[21].

- **Adaptation of the SD Service for the functional interpretation of gene sets**
  The main purpose of a typical microarray experiment is to find a molecular explanation for a given macroscopic observation. The most common methods are based on a 'functional enrichment'. First, genes of interest (e.g. genes that are significantly over- or under expressed when two classes of experiments are compared) are selected. Then, external sources of information, such as gene ontologies and pathways databases, are included to translate the set of genes into interpretable biological knowledge. We extend the SD service by a component that transforms the dataset submitted by the user into a large list of genes enriched by GO terms. We adopted an approach first presented by Trajkovsky [2].

- **Development of a user friendly interface.** The user interface to the SD service is presented in Figure 3.10. The parameters serve in both controlling the complexity of the output, such as the number of retrieved subgroups, as well as in defining a particular knowledge database used for interpretation of genes.

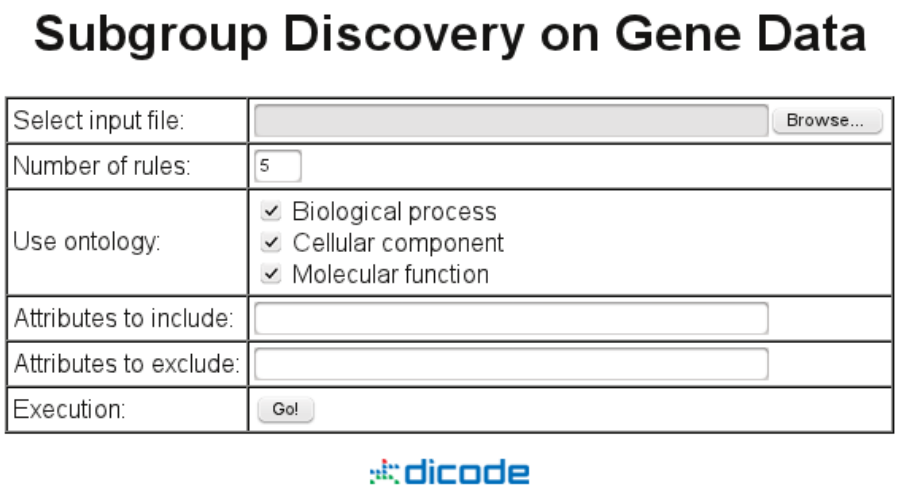

Figure 3.10 Interface of SD service in Dicode Workbench

[21] http://www.reactome.org/ReactomeGWT/entrypoint.html

- **Integration into the collaborative workbench.** Figure 3.9 illustrates the interactive character of the SD service. First, the user specifies parameters such as a category of GO and the number of retrieved subgroups (Figure 3.10). Then, the service identifies the most interesting subgroups and displays them in the collaboration workbench (Figure 3.11). Each retrieved subgroup is presented as a single element (XML document) and can be discussed separately.

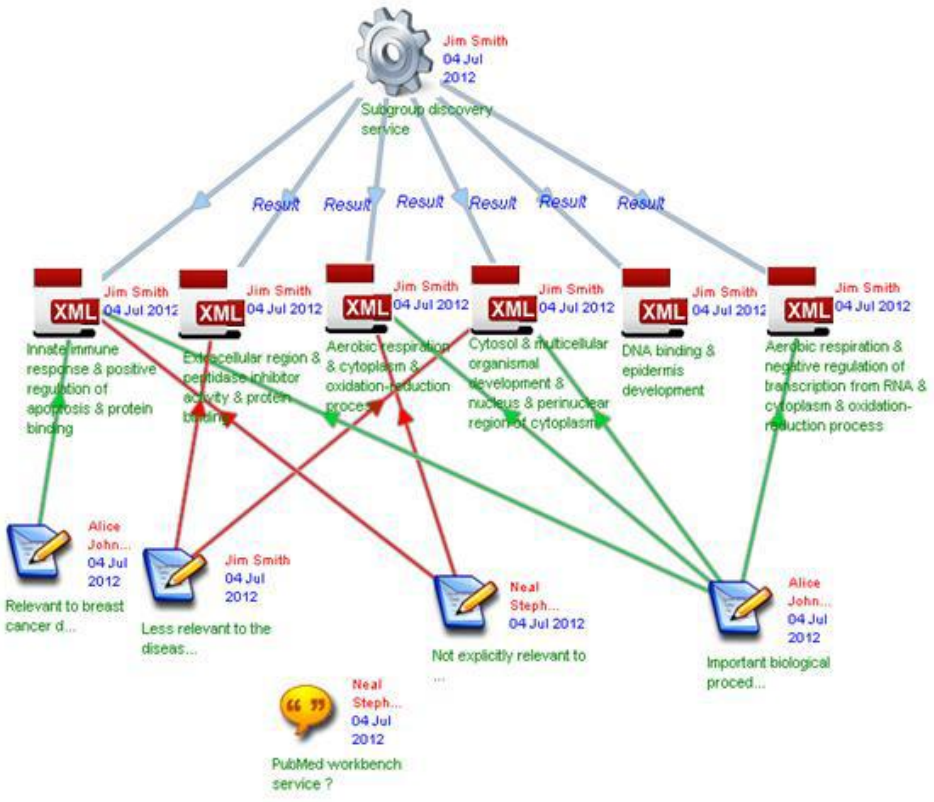

Figure 3.11 Discussion of SD results in the collaboration workbench

In summary, we provide a service for functional interpretation of genomic data. The service is based on a subgroup discovery algorithm and includes external knowledge databases (GO). The service has an interface that enables the user to customize the output.

Name	SD for functional interpretation of gene data
Standards	.csv (http://www.ietf.org/rfc/rfc4180.txt)
Description	The Dicode Subgroup Discovery Service for functional interpretation of genomic data returns a number of subgroups that describe the property of interest, such as a specific disease in use case 1, in terms of entries from the Gene Ontology. The service automatically includes the knowledge stored in GO database. The Subgroup Discovery Service provides its functionality through the

	following interfaces: •ServiceCapabilities: Informs about the common and specific capabilities. •Discovery: generates the subgroup patterns
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported information item types, similarity models related to the type, and output options
Comments	• The service is an instance of the subgroup discovery service which was adapted for the tasks of functional interpretation of genomic data (Use Case 1).
Implementation	Prototypical version of the service including GUI interfaces is implemented.

3.3.3 Recommender Service

Task 4.2 of WP4 deals with recommendation mechanisms, aiming to navigate a user through a large space of possible choices. In the fields of medical (use case 2) and biomedical research (use case 1), tools are needed to reduce the information overload, provide advice in finding an interesting dataset or medical report, and facilitate decision-making. Recommendation exploits user feedback to predict the "preference" that a user would give to an item not seen before. The Recommender Service and Similarity Learning Service address this issue. They have been initially presented in deliverables D3.1.1 and D3.2.1. The services form two necessary components of a recommendation mechanism (Task 4.2). While the Recommender Service provides the user with relevant and interesting information, Similarity Learning Service aims to create a similarity model from the user preferences, which can be used for user specific recommendations.

Both services support a general framework that can be easily adapted to recommendation of a wide range of information objects. We illustrated this in the example of Recommender System for Gene Expression Omnibus (GEO) datasets, which was created using both services.

A prototype of the Recommender Service has been implemented and described in deliverable D3.2.1. The service ranks information items with respect to their importance to the user. In some cases, it is hard to formally model the user's interests. It may be easier for the user to define an object of interest instead. Given a particular information item the user is interested in (the "reference object"), the most similar items will be then recommended to the user. The service first computes a similarity between the item the user is interested in and each item among the candidate items. In the second step, the algorithm orders all items according to their similarity to the reference object and returns the most similar ones. This method is computationally intensive since its computation time is increasing linearly to the number of comparisons. This computational burden can be reduced by pre-structuring the data, e.g. using Antipole tree indexing as proposed by Cantote et al. [3]. To reduce the computation time, we adopted this approach and integrated it into the service.

Name	Recommender Service
Standards	see D3.2.1
Description	All details about basic functionalities described in the Deliverable D3.2.1 are still valid. Additionally, the service has been extended by a component for creation of an efficient data structure that intends to significantly reduce time of search for similar items.
Interface	*ServiceCapabilities*
getCapabilities	see D3.2.1
Interface	*Recommendation*
query	see D3.2.1
feedback	see D3.2.1
Example usage	see D3.2.1
Comments	Computation time is reduced due to the integrated algorithm for building an efficient data structure.
Conformance classes	see D3.2.1
Implementation rules	see D3.2.1
Implementation status	The algorithm is provided as a service and can be used as a general framework for recommender tasks. The feasibility of the service is demonstrated on the example of Recommender Service for GEO datasets.
UML model	Not available.

3.3.4 Similarity Learning Service

Deliverable D3.2.1 presented the Similarity Learning Service as a generic learning framework that can operate on a wide range of information items from different research fields. Given an information object, the service delivers a similarity model that is learned from user feedback – a set of object pairs labelled as "similar" or "dissimilar". Therefore, the most important prerequisites for model creation are the availability of training data and a set of basic similarities that are defined by a domain expert. The service is specific enough to learn an accurate similarity model that delivers good results.

Due to the variety of information objects, the question of how to define a similarity for them is a challenging task. It is even more complicated in case the definition of similarity depends on user needs. It makes no sense to generalize a recommendation function among multiple users. There is a requirement for an easy-to-use service that each user can create a similarity model according to his/her particular preferences.

At the same time, a typical user will be unwilling to spend a lot of time to set up a recommendation system. The process of obtaining labelled data is costly in terms of time and manual effort. In order to start learning with n examples, the user needs to give his feedback for $n * (n-1)$ object pairs. Hence, he should be only asked for input that he can give quickly and correctly. In particular, it is very favourable to ask the user only questions regarding specific instances, for which domain experts can usually give very concrete feedback. As an example, when recommending papers to read, it is better to ask the user "is this paper relevant to you?" instead of "do you like to see more papers of the same author?"

In order to reduce the user's efforts in labelling, we investigated how to select a small set of pairs that is informative enough to create an accurate model. We developed an intelligent sampling strategy that selects the most 'interesting' pairs from a pool of unlabelled data to show them to the user [1].

We exploit the idea of describing massive data sets by using latent components proposed by Thurau et al. [4]. We adopted the algorithm presented in their article for sampling the most 'informative' instance pairs. The main idea is to describe data by representing it as a linear combination of dominant latent components. Let $V = [v_1,\ldots, v_n]$ be a data matrix. It can be decomposed into a product of two lower rank matrices W and H: $V \approx WH$. The matrix W determines a basis of extreme points $W = [w_1, \ldots, w_k]$, $k \ll n$. $H = [h_1,\ldots, h_n]$ contains the mixing coefficients that result from solving constrained quadratic minimizing programs. This yields basic vectors that usually correspond to the most extreme data points. Moreover, these vectors span over a simplex that encloses most of the remaining data. More details about the algorithm can be found in [4]. To find the most 'informative' instance pairs, we set a data matrix $V = P$ a matrix of instance pairs represented by numerical vectors.

The described approach was integrated in the Similarity Learning service. This extension contributed significantly to the quality and usability of the service, since the new sampling strategy enables to create an accurate personalized similarity model on the cost of minimal manual efforts from the user.

Name	Similarity Learning Service
Standards	see D3.2.1
Description	All the details about basic functionalities described in the Deliverable D3.2.1 are still valid. Additionally, the service has been extended by a component for finding the most 'informative' instance pair that is presented to the user for labelling. This enables to create an accurate similarity model with low manual efforts.
Interface	*ServiceCapabilities*
getCapabilities	see D3.2.1
Interface	*Recommendation*
query	see D3.2.1
feedback	see D3.2.1

Example usage	see D3.2.1
Comments	
Conformance classes	see D3.2.1
Implementation rules	see D3.2.1
Implementation status	The algorithm is provided as a service and can be used as a generic framework for similarity learning tasks. The feasibility of the service is demonstrated on the example of Similarity Learning Service for GEO datasets.
UML model	Not available.

3.3.5 Recommender for GEO Datasets

The analysis of in-house data is often restricted by the small size of datasets. However, an enrichment of data by additional datasets can significantly improve the results. As a high number of datasets can be generated and stored relatively easily, it becomes increasingly hard to keep an overview of them. Dataset repositories such as Gene Expression Omnibus (GEO) tackle this problem by offering the possibility to publish, explore and query archived datasets. An even higher amount of support is given by different tools integrated into a repository. However, for a single scientist it is difficult to stay aware of all the work being done that may be relevant to him. In order to find an interesting dataset, the user has to define the search criteria. The search for an appropriate dataset is a time-consuming manual process. Due to the complexity of research issues, the user would benefit from recommendations that were generated automatically according to his interests. There is clearly a need for a recommender system that facilitates the reuse and retrieval of datasets.

Taking into account this issue, we developed the Recommender and Similarity Learning Services to provide the user with relevant and interesting datasets from GEO repository. The Gene Expression Omnibus (GEO) is the largest public repository for high-throughput gene expression data. GEO was initially set up to store gene expression data generated by microarrays and serial analysis of gene expression. To facilitate the usage of the services, we developed a user friendly interface.

A dataset in the context of the GEO repository is an item that defines a set of related *Samples* considered to be part of a study and describes the overall study aim and design. A *Sample* record is composed of a description of the biological material, the experimental protocols to which it was subjected and a data table containing abundance measurements for each feature. Each stored dataset is created by a specific user, is associated with a set of samples, and contains text information such as title, description, and summary. The similarity model was set according to an expert feedback about the importance of single attributes. We have set in the service the following list of basic attribute-similarity measure pairs, which compose the similarity model used for recommendation of GEO datasets:

- Q-Gram distance on title text
- Q-Gram distance on summary text
- Cosine distance on overall study design text
- Cosine distance on experiment type

- Dice similarity on organism
- Euclidean similarity on the number of samples

The importance of each single attribute-measure pair is obtained from user feedback in the similarity learning process. The user is shown pairs of datasets and is asked to mark them as "similar"/"dissimilar"/"don't know". The presented pairs are selected according to the sampling algorithm, described in Section 3.3.4. Figure 3.12 shows an example of a dataset pair presented to the user in the learning process.

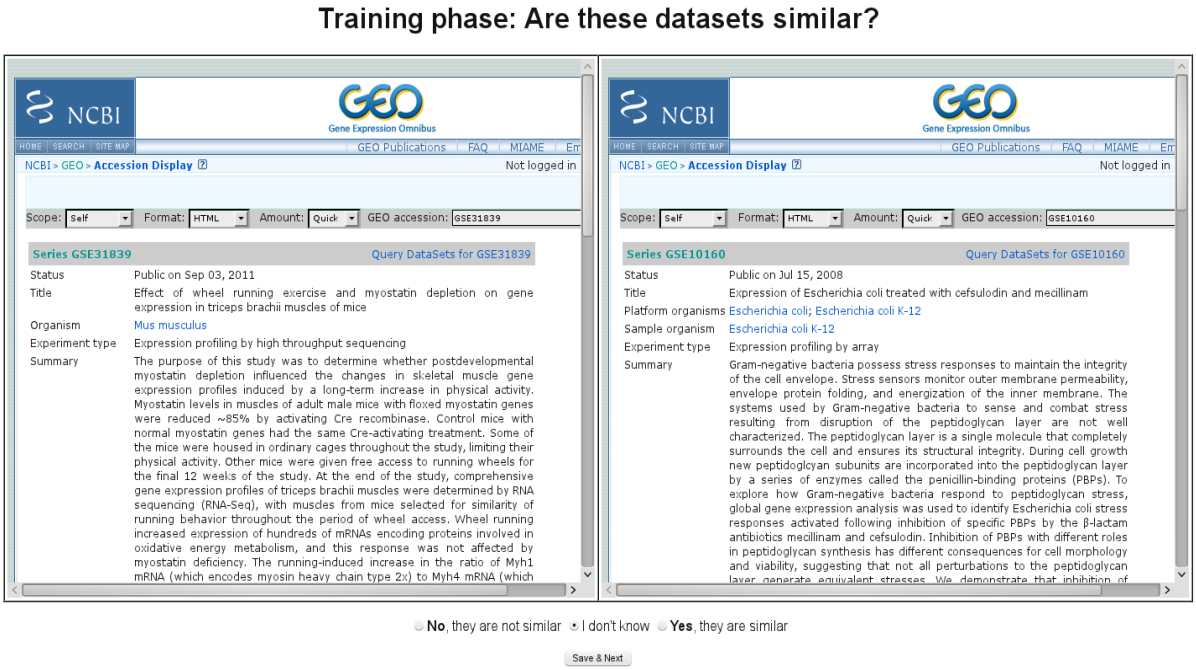

Figure 3.12 Similarity Learning Service - Interface to obtain a user feedback about the similarity of a dataset pair

Once a similarity model is learned, the user can start a recommendation process. Figure 3.13 presents the GUI interface for the recommendation service. The user can formulate his needs in two different ways: by defining a dataset of interest from GEO repository or by filling in some text field. The service returns a list of the *k* datasets that best satisfy the criteria defined by the user.

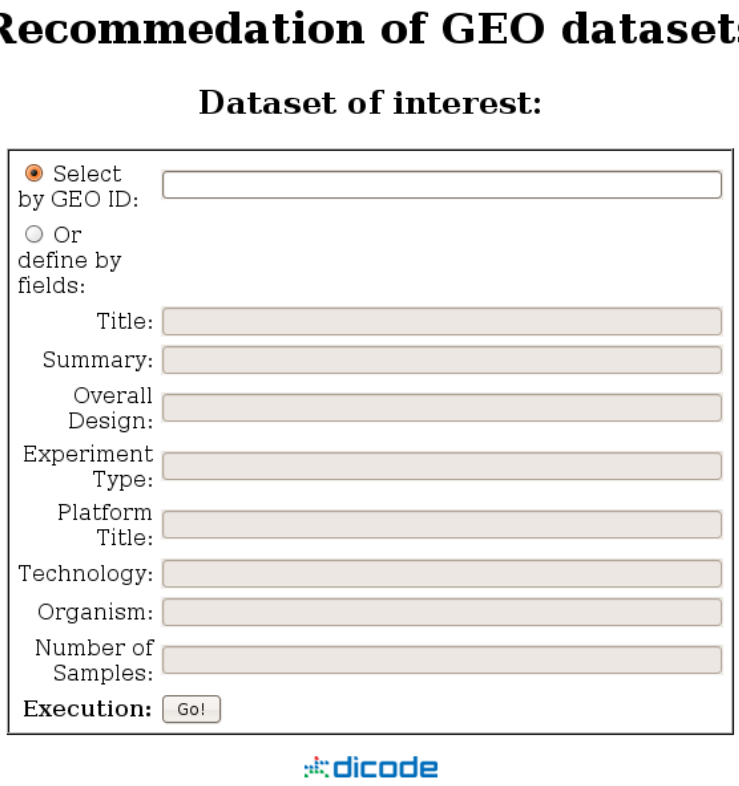

Figure 3.13 Interface of Recommender Service

3.3.6 RapidMiner Service

RapidMiner is a widely used open source data mining workbench suitable for many data mining tasks. It features a graphical interface to design custom analysis workflows, which enables non data mining experts to easily design their first data mining workflow. RapidMiner comes with a wide set of functionalities for:

- Process control
- Import
- Export
- Data generation
- Data Transformation
- Modeling
- Evaluation

A wide set of well-known machine learning techniques from the following areas are available and tested:

- Classification and Regression
- Clustering
- Association Rule and Itemset mining
- Correlation and Dependency computation

These include:

- Support Vector Machines
- Artificial Neural Networks
- Decision Trees
- Naive Bayes Classifiers

- Logisitic Regression
- FP-Growth
- several discretization strategies
- and many other

The wide set of algorithms makes it easy to try different algorithms on a data set to find a suitable model. These models may be as simple as decision trees, which are easy to understand or as complex as artificial neural networks.

Most algorithms have a number of parameters which allow further tuning. Automatic parameter adjustment and cross validation may be used to tune a robust model. The RapidMiner service is very generic in nature and applicable to a large range of analysis tasks. It may be used for single tasks, such as data transformation, data inspection, data analysis or validation as well as for entire workflows with many analysis, validation and transformation steps.

The Dicode RapidMiner Service offers the functionality of RapidMiner within Dicode as a REST service. In particular, it features a workflow and result repository to share workflows and results with other Dicode users. This contributes towards the goal of collaborative data analysis and decision making. The benefit of this service grows with the number of shared workflows.

A graphical interface to design new workflows is not provided within Dicode. A desktop installation of Rapid Miner is required to edit workflows.

The abstract service description of the current version of the "RapidMiner Service" is presented in the table below.

Name	RapidMiner
Standards	.xml (http://www.w3.org/TR/2006/REC-xml11-20060816/) .csv (http://www.ietf.org/rfc/rfc4180.txt) .arff (http://www.cs.waikato.ac.nz/~ml/weka/arff.html)
Description	The Dicode RapidMiner Service supports the execution of RapidMiner processes within the Dicode framework. The service provides its functionality through the following interfaces: •ServiceCapabilities: Informs about the common and specific capabilities. •DeployProcess: deploys a RapidMiner process within Dicode. •ExecuteProcess: Executes a RapidMiner process on some data within Dicode. •DownloadProcess: Downloads a RapidMiner process from Dicode. •NotificationProducer: allows subscription of notification consumers.
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported operators, data formats, and output options
Interface	*DeployProcess*

	Stores a new RapidMiner process within Dicode. Deployed processes can be used by all users.
Interface	*ExecuteProcess* Executes a RapidMiner process within Dicode. The Output is as defined in the process.
Interface	*DownloadProcess* Downloads a RapidMiner process from Dicode. Downloaded processes may be modified and Redeployed within Dicode.
Interface	*NotificationProducer*
subscribe	A subscriber can register given his/her interest as notification consumer to receive notifications. Parameter is callback of the notify operation of the respective notification consumer.
getCurrent-Message	The subscriber pulls the current notification message. Example usage: The discovery interface will be mainly applied in Use Case 1 – clinicogenomic research for detecting genomic patterns with regard to target taken from the clinical data.
Comments	• The RapidMiner service supports the execution of all processes build upon standard rapid miner capabilities and the additional specific capabilities implemented by the Dicode services, Subgroup Discovery and Similarity Learning. • Additional plugins for further custom functionalities may be easily integrated.
Conformance classes	Conforms to RapidMiner 5.1
Implementation Rules	Not yet available.
Implementation	First version of the service available.

3.3.7 Embedded R Executor

Clinico-genomic research prescribes some specific requirements to the analysis tools. Many good software modules for statistical analysis of genomic data are offered as open source. One of the most important platforms for these is R (free, open source). We offer the Embedded R Executor Service as a second data mining platform.

The abstract service description of the current version of the "Embedded R Executor Service" is presented in the table below.

Name	Embedded R Executor
Standards	http://cran.r-project.org/doc/manuals/R-lang.pdf
Description	The Dicode Embedded R Executor Service supports the execution of R processes within the Dicode framework. The service provides its functionality through the following interfaces: • ServiceCapabilities: Informs about the common and specific capabilities.

	• DeployProcess: Deploys a R script within Dicode. • ExecuteProcess: Executes a R script on some data within Dicode. • DownloadProcess: Downloads a R script from Dicode. • NotificationProducer: Allows subscription of notification consumers.
Interface	*ServiceCapabilities*
getCapabilities	Informs the requestor about the common and specific capabilities. Examples of specific capabilities are the supported operators, data formats, and output options
Interface	*DeployProcess* Stores a new R process within Dicode. Deployed processes can be used by all users.
Interface	*ExecuteProcess* Executes an R process within Dicode. The Output is as defined in the process.
Interface	*DownloadProcess* Downloads a R process from Dicode.
Interface	*NotificationProducer*
subscribe	A subscriber can register given his/her interest as notification consumer to receive notifications. Parameter is callback of the notify operation of the respective notification consumer.
getCurrent-Message	The subscriber pulls the current notification message. Example usage: The discovery interface will be mainly applied in Use Case 1 – clinicogenomic research for detecting genomic patterns with regard to target taken from the clinical data.
Comments	The service supports the execution of all processes build upon standard R packages. Additional packages may be added by an administrator.
Conformance classes	Not yet applicable
Implementation rules	Not yet available.
Implementation	Not yet provided

4 References

[1] Natalja Friesen and Stefan Rüping. Distance Metric Learning for Recommender Systems in Complex Domains Mastering Data-Intensive Collaboration through the Synergy of Human and Machine Reasoning (dicoSyn 2012). A workshop at CSCW 2012, February 12, 2012, Seattle, WA

[2] C. Thurau, K. Kersting, M. Wahabzada, C. Bauckhage. Convex Non-negative Matrix Factorization for Massive Datasets. Knowledge and Information Systems (KAIS), 2010.

[3] Domenico Cantone, Alfredo Ferro, Alfredo Pulvirenti, Diego Reforgiato Recupero, and Dennis Shasha. Antipole Tree Indexing to Support Range Search and K-Nearest Neighbor Search in Metric Spaces, IEEE/TKDE, 2005

[4] Igor Trajkovsky. Functional Interpretation of Gene Expression Data: Translating high-throughput DNA microarray data into useful biological knowledge. LAP LAMBERT Academic Publishing (December 2011)
[5] Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Machine Learning conference*, 2001 pp. 282–289.
[6] Paass, G., Kindermann, J. (2008). Entity and Relation Extraction in Texts with Semi-Supervised Extensions. In: Tresp, V., Bundschus, M., Rettinger, A., & Huang, Y. (Eds.), Security informatics and terrorism: social and technical problems of detecting and controlling terrorists' use of the World Wide Web; proceedings of the NATO Advanced Research Workshop on Security Informatics and Terrorism – Patrolling the Web, Vol. 15, p. 132 Beer-Sheva, Israel. IOS Press.
[7] Henrik Grosskreutz, Daniel Paurat, Stefan Rüping. An Enhanced Relevance Criterion for More Concise Supervised Pattern Discovery. The 18th annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012.
[8] Cucerzan, S. (2007). Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL,* pp. 708-716