D3.2.3 – The Dicode Data Mining Services (final version)

Due date of deliverable: 31 May 2013
Actual submission date: 31 May 2013
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
Summary
This deliverable describes the final version of the data mining infrastructure in the Dicode system, which consists of data mining and text mining services, as well as of services for data preprocessing. The Dicode data mining infrastructure has been designed and developed in the context of WP3 (Voluminous and Complex Data Mining Services) and WP4 (Data-Intensive Collaboration and Decision Support Services). The service descriptions and the majority of details presented in a previous related deliverable (D3.2.2) remain unchanged. This deliverable aims to inform about updates and extensions of the existing services. Moreover, the deliverable introduces two new services that were recently integrated in the Dicode workbench.
Table of Contents

1 Introduction ........................................................................................................................................... 5

2 An Overview of the Dicode Data Mining System ............................................................................... 5

3 Description of Services ......................................................................................................................... 6
   3.1 Text Mining Services ....................................................................................................................... 6
       3.1.1 Entity Prominence service ....................................................................................................... 7
       3.1.2 DBpedia Spotlight Named Entity service (Interactive Named Entity annotation) .................. 10
       3.1.3 Log Aggregation .................................................................................................................... 14
       3.1.4 Emotion Detection Training service ....................................................................................... 17
       3.1.5 Opinion Mining service .......................................................................................................... 19
       3.1.6 Phrase Extraction Application service .................................................................................... 20
       3.1.7 Phrase Extraction Training service ........................................................................................ 20
       3.1.8 Topic Detection service .......................................................................................................... 20
   3.2 Data Mining Services ....................................................................................................................... 21
       3.2.1 Subgroup Discovery Service .................................................................................................. 21
       3.2.2 Recommender Service and Similarity Learning Service ......................................................... 21

4 Conclusions .......................................................................................................................................... 22
1 Introduction

The main objective of WP3 is the development of the Dicode Data Mining infrastructure. This includes: (i) design of an appropriate Data Mining architecture enabling an integration of a wide range of services, and (ii) development and integration of services for data acquisition, data mining and data preprocessing. The current document summarizes the work carried out in the Tasks 3.1-3.7 of WP3 and Task 4.2 of WP4 and presents the final version of the Dicode Data Mining Services.

The great majority of these services have been already described in deliverable D3.2.2 (“Data Mining Services (enhanced version)”), which contains all details from both a conceptual and a technical point of view, and describes all the functionalities provided to the users. The majority of details presented in D3.2.2 remain unchanged. This deliverable aims to inform exclusively about updates of the previous content and extensions of the existing services. It focuses on the final changes and modifications. This is the last document in a series of three deliverables reporting on the progress in the development of Data Mining services. The related lessons learned from the Dicode project will be documented in deliverable D3.3 (due in month 36).

In particular, the previous version of Data Mining services, as reported in D3.2.2, was extended as follows:

- Minor changes in internal implementation of some services;
- More precise adjustment of the services to the use cases;
- Integration of new services (the DBpedia Spotlight Named Entity service and the Log Aggregation service).

The deliverable is structured as follows: Section 2 gives an overview of the Dicode Data Mining system. Section 3 presents updates of the services including text mining services and data mining services. It also introduces new services and gives detailed information about them.

2 An Overview of the Dicode Data Mining System

The Dicode Data Mining framework consists of: (i) its architecture, and (ii) a set of services. The architecture of the Dicode Data Mining framework is described in detail in D3.2.2 and is still valid. Minor changes in the internal architecture of some services are presented in the corresponding subsections. Some important decisions about the architecture will be discussed in deliverable D3.3.

The Dicode Data Mining services aim to support different types of users and to facilitate data analysis from the 3 use cases of the project. All services can be easily targeted to very specific tasks, such as analysis of genomic data. Through their integration into the collaboration workbench (i.e. collaboration workspace, see D4.1.2), the services enable the user to benefit from various decision support mechanisms, provided by the Dicode platform. We have demonstrated this in the example of a Subgroup Discovery (SD) Service for multi-platform Genomic Data Analysis, which is an instance of the SD service.

All services integrated into the Dicode system can be divided in the following groups:
Group 1: General Data Mining services – these provide a declared functionality and can be easily targeted to a specific task. They include:

- Subgroup Discovery
- Recommender Service
- Similarity Learning Service
- RapidMiner Service
- Embedded R Executor

Group 2: Instances of particular data mining services – these are services originally belonging to Group 1, which were adapted to a specific task. They can be directly applied by the user without any particular data mining knowledge. This group consists of the following services:

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

Group 3: Text mining services – these services enable the processing and analysis of unstructured data, in particular text. The full list of these services is as follows:

- Twitter Harvester
- Twitter pre-processing service
- Blog pre-processing service
- Named Entity service
- Entity Prominence service
- Topic detection service
- Phrase extraction service
- Phrase extraction training service
- Emotion detection service
- Relation extraction service
- DBpedia Spotlight Named Entity service (Interactive Named Entity annotation)
- Log Aggregation Service

3 Description of Services

The great majority of details already described in deliverable D3.2.2 remain unchanged. To keep the document short, we will not present them again.

3.1 Text Mining Services

In deliverable D3.2.2, we described a series of text mining services that are updated here. These are:

- Entity Prominence service (section 3.1.1)
- DBpedia Spotlight Named Entity service (section 3.1.2)
- Emotion Detection Training service (section 3.1.4)
- Opinion Mining Service (section 3.1.5)
- Topic Detection service (section 3.1.8)

In this deliverable, we describe the changes and extensions that have been made in the third year of the project. The Relation Extraction service has been renamed as Opinion Mining service. Furthermore, its functionality has been changed, because it did not seem practical to
offer the user a generic relation extraction service. Instead, we decided to implement a more specialised service that can provide insights into the opinions of customers that are related to certain topics.

3.1.1 Entity Prominence service

Some additions were made to Dicode’s Entity Prominence service. This service returns statistics about the occurrence of Named Entities in news and blog documents within a certain time period. Additions to the service were still in progress during the production of deliverable D3.2.2. Therefore, we now use the opportunity to present the changes and give some background information about the architecture of our software component “Dicube”, a distributed OLAP cube for event aggregation on which the service is based.

---

**Figure 1 – Dicube architecture**
The abstract service description of the Entity Prominence service remains unchanged, but the implementation has been improved in many ways. At the back end, the service uses our “Dicube” software, a distributed OLAP-cube for event aggregation. As described in D3.2.2, the current implementation is based on the “datacube” project developed by the company Urban Airship1. Our implementation adds missing features like slicing to the data cube.2 The first version (“EntityCube”) was developed exclusively for the Entity Prominence service. In the current version, the implementation was generalized and can be reused for a variety of use cases dealing with highly scalable event aggregation. The first prototype was further improved to become a generic component consisting of all layers from persistence to visualization and offering a REST based interface for interoperability to other components. The general architecture of Dicube is described in Figure 1, which shows the read/write path of the application.

The current implementation permits both batch- and stream import. Depending on the size of the document collection and the number of dimensions and rollups defined, the number of events to be stored can be quite high. We therefore pre-aggregate the events before storing them in HBase3. Our implementation currently comes with an HBase sink, which can be easily replaced by an implementation using any other KeyValue store with column/dictionary support such as Apache Cassandra4 or Redis5, if slicing is needed. The complete configuration of a Dicube backend is performed in a declarative way via Spring6. The mapping defined in the Spring configuration defines both the dimensions and rollups of the OLAP cube and the REST API which is queried by the Dicube frontend.

The UI is defined in a declarative style via JSON. Without any coding, the user can specify a set of charts and filters to be displayed. The following example shows the definition of two form elements: a date range filter which is applied to a selection of fields, and an autocomplete filter where the user enters the top level domain for which he wants to look up the prominence counts. The list of autocomplete values is retrieved from the file system.

```json
{
    "xAxisFilters": {
        "time": {
            "type": "DateRangeFilter",
            "filters": [
                "language",
                "entity_uri",
                "domain"
            ]
        }
    },
    "dynamicFilters": {
        "domain": {
            "type": "AutocompleteFilter",
            "config": {
                "displayName": "Domain",
                "defaultValue": "",
                "prompt": "Domain (without dots)",
                "autocomplete": true,
                "minLength": 0
            }
        }
    }
}
```

1 https://github.com/urbanairship/datacube
2 We made those features available in our fork of the Datacube project on Dicode’s GitHub account at https://github.com/dicode-project/datacube
3 http://hbase.apache.org/
4 http://cassandra.apache.org
5 http://redis.io/
6 http://www.springsource.org/spring-framework
The service has been integrated into the Dicode workbench. An improved version of the UI based on D3.js\(^7\), which will remove the dependency on Google’s APIs and a mandatory internet connection, is in preparation and will be presented in deliverable D5.4.3 (“Integrated Dicode Services (final version)”, due in month 34).

An extensive documentation of all modules of Dicube was created, which is part of the Maven project “dicube” and built using the Maven site plugin. The documentation includes a step-by-step tutorial (Figure 2) for the implementation of an OLAP cube and a visualization web application in a retail scenario. The tutorial helps any software developer at NEO to implement a cube project without too much support from the developers.\(^8\)

The following table contains the abstract service description of the Entity Prominence service.

\(^7\) [http://d3js.org/](http://d3js.org/)

\(^8\) Details about the use of the component will be given in the final Dissemination and Exploitation Activities Report (D7.2.2).
**Name** | Entity Prominence service  
---|---
**Description** | The Entity Prominence service can be used to query the "prominence" of Entities in documents and blogs over time. The Entity Prominence service provides its functionality through the following interfaces:
- ServiceCapabilities: Informs about the common and specific capabilities.
- Prominence: Provides the prominence counts

**Implementation status** | API available for evaluation. Entity counts for ca. 60 million news documents for German and English news sources. New: The visualization component has been implemented and integrated into the Dicode workbench. An improved version of the interface is in preparation.

3.1.2 **DBpedia Spotlight Named Entity service (Interactive Named Entity annotation)**

An interactive Named Entity annotation service is required for the following use case which is part of UC3: A marketing professional wants to retrieve additional meta data about all brands found in a single document. For this task, we implemented a new service: DBpedia Spotlight Named Entity service.

Dicode’s original Named Entity service does not have the required capabilities, because it annotates named entities in batch mode. As described in D4.1.3 (“The Dicode Collaboration Support Services (final version)”)⁹, we enhanced interactivity for the software developer by adding an operator to the processing language MiaQL which is available on NEO’s development cluster¹⁰. A developer who wants to annotate a newly added document collection can easily annotate named entities in the collection by executing the operator. Despite the improved interactivity, the available Named Entity service still lacks the opportunity to analyze single documents instantly.

The newly implemented service is based on DBpedia Spotlight¹¹, an open source project for automatic annotation of DBpedia¹² entities in natural language text. The project is closely connected to Dicode, because NEO’s Max Jakob is one of the core developers of DBpedia Spotlight.¹³ It provides programmatic interfaces for phrase spotting (recognition of phrases to be annotated) and disambiguation (entity linking), as well as various output formats

---

⁹ See section “Interactive Analysis” in D4.1.3 for details about interactive analysis of document corpora.  
¹⁰ Documentation of MiaQL will be available soon.  
¹² http://dbpedia.org/  
¹³ NEO’s contribution to this year’s IESD challenge at the Second International Workshop on Intelligent Exploration of Semantic Data (IESD2013) at Hypertext was based on DBpedia Spotlight. In the project “Semantic Exploration of Open Source Software project Descriptions”, the descriptions of the projects participating in Google’s summer of code were analyzed. The available information was enriched with background knowledge retrieved from the DBpedia knowledge base. For a demo, see http://dbpedia-spotlight.github.io/gsoc-searcher/
(XML, JSON, RDF, etc.) in a REST-based web service. The standard disambiguation algorithm is based upon cosine similarities and a modification of TF-IDF weights (using Apache Lucene\textsuperscript{14}). The main phrase spotting algorithm is exact string matching, which uses LingPipe’s Aho-Corasick\textsuperscript{15} implementation.

The project initially focused on the English language, but models for more and more languages have been recently provided by the community.\textsuperscript{16} However, since DBpedia Spotlight’s models are learned from Wikipedia, it should be possible to adapt the system to any other language that has a Wikipedia edition. Figure 3 shows the user interface of the DBpedia Spotlight Named Entity service in the Dicode workbench.

![Figure 3 - DBpedia Spotlight Named Entity service](image)

The user can restrict the entities spotted in the document by selecting types via the “SELECT TYPES” menu, as shown in Figure 4.

\textsuperscript{14} http://lucene.apache.org/
\textsuperscript{15} http://en.wikipedia.org/wiki/Aho-Corasick_algorithm
\textsuperscript{16} Demonstrations of internationalized versions are provided in English and Dutch and models for 9 additional languages are made available via the supporting material for a submission to this year’s ISemantics conference. See: https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/ISemantics-2013-Supporting-Data
The following table contains the abstract service description of the DBPedia Spotlight Named Entity service.

<table>
<thead>
<tr>
<th>Name</th>
<th>DBPedia Spotlight Named Entity service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standards</td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>
| Description              | The DBpedia Spotlight Named Entity service is used to annotate named entities in single documents. The service integrated into the Dicode workbench processes English documents only. The DBpedia Spotlight Named Entity service provides its functionality through the following interfaces:  
  - *ServiceCapabilities*: Informs about the common and specific capabilities.  
  - *NER*: Annotates named entities |

<table>
<thead>
<tr>
<th>Interface</th>
<th>getCapabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>ServiceCapabilities</em></td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Interface</td>
<td>NER</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>annotate</td>
<td>Annotates text with named entities. Input: English text. Parameters:</td>
</tr>
<tr>
<td></td>
<td>- Confidence (double): combined parameter that applies two checks:</td>
</tr>
<tr>
<td></td>
<td>o The similarity score of the first ranked entity must be bigger than a threshold.</td>
</tr>
<tr>
<td></td>
<td>o The gap between the similarity score of the first and second ranked entity must be bigger than a relative threshold</td>
</tr>
<tr>
<td></td>
<td>- Contextual score: It ranges from 0 to 1. When large values of contextual score are selected, the application does not annotate terms with little topical pertinence.</td>
</tr>
<tr>
<td></td>
<td>- Support (int): number of Wikipedia inlinks an annotated resource must have</td>
</tr>
<tr>
<td></td>
<td>- Restrict spotting: “No common words” (default) / Annotate all spots / No verbs and adjectives / Only persons, organizations, locations</td>
</tr>
<tr>
<td></td>
<td>- Disambiguation algorithm: Default disambiguation (default) / Occurrence-centric / Document-centric</td>
</tr>
<tr>
<td></td>
<td>- Disambiguation result: show best candidate (default) / show N best candidates</td>
</tr>
<tr>
<td></td>
<td>Output consists of texts with named entities annotated.</td>
</tr>
<tr>
<td>Example usage</td>
<td>The Entity Prominence service will be applied in use case 3 for text analysis:</td>
</tr>
<tr>
<td></td>
<td>- A marketing analyst wants additional information form DBpedia for all entities occurring in a single text</td>
</tr>
<tr>
<td>Comments</td>
<td>For detailed documentation see <a href="http://dbpedia.org/spotlight/technicaldocumentation">http://dbpedia.org/spotlight/technicaldocumentation</a></td>
</tr>
<tr>
<td>Conformance classes</td>
<td>Not available.</td>
</tr>
<tr>
<td>Implementation rules</td>
<td>Not available.</td>
</tr>
<tr>
<td>Implementation status</td>
<td>Integrated into the Dicode workbench.</td>
</tr>
<tr>
<td>UML model</td>
<td>Not available.</td>
</tr>
</tbody>
</table>
3.1.3 **Log Aggregation**

During the course of the project, NEO’s Dicode team constantly strived to improve manageability and stability of our distributed computing infrastructure. In the third year of Dicode, a new service for Log Aggregation of NEO’s development cluster was added. Debugging distributed systems, like Hadoop and especially Hadoop’s distributed database HBase, can be rather tedious. Tracking down the root cause of an error involves searching multiple gigabytes of logfiles on various machines. Using standard Unix tools like *grep* for searching is inefficient, because a search on a single machine might already take several minutes. Analysing log files on a dozen of nodes requires plenty of time. Log aggregation has therefore become standard for the operation of large software systems like Hadoop clusters. Besides log aggregation, commercial products like Splunk or open source software like *logstash* offer a convenient search interface. Dicode’s Log Aggregation service, which is deployed on NEO’s development cluster, uses logstash.

Logstash permits the usage of a variety of technologies for processing and storage of logfiles and thus offers both the performance and flexibility required to process such potentially high volume data. Our setup is based on Redis which is used as event queue and *elasticsearch* as persistence layer. Logstash has an active community and develops quite fast. Thanks to its modular concept, logstash is highly extensible. Numerous input modules, filters and output modules for most of the common open source technologies in this area are already available. As front-end application, we use *Kibana*, an open source interface to logstash and elasticsearch, which is available under the MIT licence.

Figure 5 shows the logstash interface.

![Figure 5 - Kibana interface (initial view)]

---

19 [http://logstash.net/](http://logstash.net/)
21 [Elasticsearch](http://www.elasticsearch.org) is an increasingly popular open source search engine based on Apache Lucene.
Via the timeline, the user can zoom into the log messages. The menu on the left of Figure 6 offers filtering and/or highlighting capabilities.

![Columns menu](image)

**Figure 6** – Filtering by host in logstash Kibana interface

Additionally, logstash offers a streaming view which serves the incoming messages instantly as shown in Figure 7.

![Streaming view](image)

**Figure 7** – Logstash streaming view
Due to security restrictions, the Log Aggregation service is only available within NEO’s internal network. Access from outside the network is not necessary because the log analysis interface is only used by developers responsible for the operation of the development cluster. If external parties (like FHG) develop a module which is deployed via the MIA web application23, debugging of Hadoop/HBase lies in the hands of NEO’s developers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Log Aggregation service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standards</td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>
| Description           | The ServiceCapabilities Interface: Informs about the common and specific capabilities.  
  - Aggregation: log file aggregation |
| Interface             | ServiceCapabilities     |
| getCapabilities       | Informs the requestor about the common and specific capabilities. |
| Interface             | Aggregation Interface   |
| generalView           | Shows aggregated log entries.  
  A set of facets like tags used in the log file, host name and other common elements are automatically retrieved and indexed as facets for faceted search. They can be used as filters.  
  Date filtering via the timeline is also possible. |
| realtimeView          | Shows incoming messages in near real-time |
| Example usage         | The Log Aggregation service is used for log file analysis on the development cluster. Example usage might be:  
  - Analysis of HBase distributed database after system inconsistencies are detected |
| Comments               |                         |
| Conformance classes   | Not available.          |
| Implementation rules  | Not available.          |
| Implementation status | Set up on the development cluster at NEO. Currently, the aggregation server is deployed on a standard workstation, which is accessible in the network. In the future, it will be available on a machine sharing a switch with the cluster’s nodes to minimize latency and network traffic. |
| UML model             | Not available.          |

23 See D4.1.3 for details.
3.1.4 Emotion Detection Training service

This service has been developed in the third year of the project. It was described as an abstract service in deliverable D3.2.2. It is based on the Phrase Extraction Training service. This service does not need a list of phrases to learn. It only needs a list of wanted seed terms (that may include regular expressions) and a list of terms (or regular expressions) that should not be contained in the phrases. Figure 8 and Figure 9 provide example expressions. The service will use all phrases around matching seed terms for the training. Special routines to determine phrase boundaries have been implemented. A list of training phrases is a result of the execution (together with the extraction model). The list may be edited and re-used to train an even better model with the Phrase Extraction Training service.

```
"([Ff]antastisch",
"([^]| )([Ww]under([Aa]ller|[Bb]ild])*schön(e|es|ne|em|er|ere|eres|eren|erem|ste|sten|stem)(\$| )")",
"([^] )Cool(st)*(e|en|er|em|es)*( |\$)",
"([^] )Schön(e|es|ne|em|er|ere|eres|eren|erem|ste|sten|stem)(\$| )")",
"([^] )Spa(ss|ß)\(\$| )"
"([^] )[Ff]re(e|st|t|en)( |\$)"
"([^] )[Gg]lücklich\(\$| )"
"([^] )[Ww]under(bar|voll)(er)\(e|es|er|em|es\)(\$| )"
"([^] )g(e)ffallen\(\$| )"
"([^] )klasse\(\$| )"
"([^] )prima\(\$| )"
"([^] )super\(\$| )"
"([^] )toll\(e|es|er|em)(\$| )"
"(ober|super|mords|^| )cool(st)*(e|en|er|em|es\)\( |\$)"
"[Bb]egeistert"
"[Ee]mpfehenswert"
"[Gg]eil"
"[Hh]errlich",
"[Hh]ervorragend",
"[Ss]ensationell"
"[Zz]ufrieden",
"hervorragend"
```

Figure 8 – Regular expressions of wanted seed terms for the emotion “delight/pleasure” (extract).
Figure 9 – Regular expressions of unwanted terms for the emotion “delight/pleasure” (extract).

The collection of training phrases with the regular expression heuristics simplifies this task considerably and can be performed more effectively than collecting examples manually.
3.1.5 Opinion Mining service

This service is a combination of the Phrase Extraction Application service and the Topic Detection service. It first detects phrases in a text collection. Then, it builds a topic model on those texts that contain at least one detected phrase. In a last step, the topic model outputs the most significant sentences assigned with each topic. This service needs a trained phrase extraction model to operate.

![Graphical output of the Topic Detection Service on a text collection of notebook reviews with the query term “performance”.

The graph shown in Figure 10 has been constructed from a text collection on notebook reviews. We show a part of the whole graph that only contains topics that are related to the query term “performance”. Figure 11 shows a digest of sentences related to “topic 88”, which contain positive emotions. This digest expresses (positive) opinions of consumers that are related to “performance” and to “gaming”, which is one of the main terms of this topic (topics can be defined by words of the text collection that are related to the topic with large probabilities).
3.1.6 Phrase Extraction Application service

The Phrase Extraction Application service has not been changed since its description in deliverable D3.2.2, section 3.1.7.

3.1.7 Phrase Extraction Training service

The Phrase Extraction Training service has not been changed since its description in deliverable D3.2.2, section 3.1.8.

3.1.8 Topic Detection service

This service has been extended to allow certain modification of the topic model graph by restricting the number of topics to display, or by including meta-data. The details have been described in deliverable D5.4.2, section 4.3.23.
3.2 Data Mining Services

3.2.1 Subgroup Discovery Service

The Subgroup Discovery (SD) service uses a specialized in-memory database using specialized data structures like \textit{fp-trees}\textsuperscript{24}. The majority of subgroup discovery algorithms typically rely on top-down search combined with considerable pruning, which exploits anti-monotonicity of the quality measure. When dealing with high-dimensional data, the hypothesis space becomes extremely large, and the whole discovery process becomes overly time-consuming. The SD algorithm used in SD service uses weighted covering strategy\textsuperscript{25}, which was proven to perform well in comparison with the other similar algorithms\textsuperscript{26}.

The SD service and its instantiation – the Subgroup Discovery for Genomic Data Analysis - have not been changed since their description in deliverable D3.2.2 (sections 3.3.1 and 3.3.2).

3.2.2 Recommender Service and Similarity Learning Service

The Dicode recommender system was especially developed for recommendation of scientific items. The recommender systems in a scientific context are significantly different from the standard case of product recommendations. The biggest issues are the representation of complex objects, a smaller - more heterogeneous - set of users and a lack of information about the user’s preferences. One of the most important requirements for such systems is their ability to provide personalized recommendations. As the users working on a collaborative problem come from various disciplines – the typical usage scenario being small teams of people working on different projects - a single recommendation scheme might not cover the needs of all users. Thus, recommendations need to be more personalized.

For ease-of-use purposes, the user should not be burdened with the additional trouble of customizing the recommendation. Instead, the system should use machine learning techniques to adapt itself to the user’s preferences. This issue is addressed by the Similarity Learning service, which aims to create a similarity model from the user’s feedback. The very important advantage of this approach is that it avoids the so called “cold start problem”, common for systems based on collaborative filtering approach. Collaborating filtering is based on collecting users’ profiles. A typical profile consists of aggregated information about a user’s preferences that are represented by a set of rated items. The user is recommended items that liked people with similar taste. A more detailed comparison of recommender systems in e-science with common approaches can be found in a recent Dicode paper\textsuperscript{27}.

A Dicode-specific recommender system that consists of the Recommender service and the Similarity Learning service is designed for the field of e-Science. An appropriately-designed

\textsuperscript{24} J. Han, J. Pei, Y. Yiwen and R. Mao (2004): Mining Frequent Patterns without Candidate Generation: A Frequent-\textit{Pattern} Tree Approach, Data Min. Knowl. Discov.


sampling strategy enables to avoid the “cold start problem” and helps to provide the user by personalized recommendations.

The Recommender service and Similarity Learning service, as well as their instantiation Recommender for GEO Datasets, have not been changed since their description in deliverable D3.2.2 (sections 3.3.3, 3.3.4 and 3.3.5).

The Dicode recommender system is going to be evaluated in a scenario with real users. The evaluation results will be described in deliverable D3.3 (“Data Mining in Data-Intensive and Cognitively-Complex Settings: Lessons Learnt from the Dicode Project”, due in month 36).

4 Conclusions

The Data Mining architecture introduced in D3.1.1 and described in detail in D3.2.2 has been proven as successful. No fundamental changes on the architecture were necessary during the development and integration phases. The next important issues are to estimate how effective the services were deployed and to evaluate in a study with real users the usability of them. The evaluation process is still running and will be described in deliverable D3.3, due in month 36.