Mastering Data-Intensive Collaboration and Decision Making

FP7 - Information and Communication Technologies

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

D3.1.2 – The Dicode Data Mining Framework (enhanced version)

Due date of deliverable:	28.February 2013
Actual submission date:	28 February 2013
Responsible Partner:	FHG
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 framework, data mining services, Apache Pig, RapidMiner, Weka, R, Hadoop, text mining.

The Dicode project (dicode-project.eu) is funded by the European Commission, Information Society and Media Directorate General, under the FP7 Cooperation programme (ICT/SO 4.3: 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	10-02-2013	First draft, sent to internal reviewers	Natalja Friesen, FHG Jörg Kindermann, FHG Doris Maassen, NEO
2	20-02-2013	Reviewers' comments incorporated, sent to SC	Natalja Friesen, FHG Jörg Kindermann, FHG Doris Maassen, NEO Guillermo de la Calle Velasco, UPM Nikos Karacapilidis, CTI
3	27-02-2013	SC's comments incorporated	Natalja Friesen, FHG Jörg Kindermann, FHG Doris Maassen, NEO
4	28-02-2013	Final (approved by SC, sent to the Project Officer)	Natalja Friesen, FHG

Deliverable manager

- Natalja Friesen, FHG

List of Contributors

- Jörg Kindermann, FHG
- Doris Maassen, NEO
- Stefan Rüping, FHG

List of Evaluators

- Nikos Karacapilidis, CTI
- Guillermo de la Calle, UPM

Summary

This deliverable reports progress on the development of the Dicode Data Mining Framework. It includes a detailed report on modifications of the framework components that were first presented in deliverable D3.1.1 ("The Dicode Data Mining Framework (initial version)"). Furthermore, the deliverable informs about the associated modifications of the Dicode data mining services. Finally, it discusses issues and lessons learnt from the usage of the selected framework components and their integration in the Dicode services.

Table of Contents

1 Introduction

This deliverable describes the current status (enhanced version) of the Dicode Data Mining Framework that has been designed and developed in the context of WP3.The components of this framework have been initially presented in deliverable D3.1.1: "The Dicode Data Mining Framework (initial version)". This is the final document in a series of deliverables reporting on the design of Dicode's Data Mining Architecture (see also deliverable D3.2.2 "The Dicode Data Mining Services (enhanced version)"). Its main purpose is to report on the modifications performed.

In particular, the initial version of the Dicode Data Mining Framework (DDMF) was modified due to:

- changes in the selection of data mining frameworks adopted, in order to better handle integration issues, and
- the associated adaptation of the Dicode data mining services.

Modifications on the DDMF components are presented in Section 2. A series of issues concerning the usage of the selected components and their integration in Dicode data mining services are discussed in Section 3.

2 Dicode Data Mining Framework Components

To avoid repetition, we only describe the components of the DDMF which have not been discussed in other deliverables in detail. For information about all other components, refer to D3.1.1 ("The Dicode Data Mining Framework (initial version)") and D3.2.2 ("The Dicode Data Mining Services (enhanced version)").

2.1 Frameworks

This section reports on all important modifications regarding the selection of data mining tools and frameworks that were initially described in D3.1.1.

2.1.1 Apache Pig[1]

In the initial phase of Dicode, NEO developed applications in plain MapReduce[2]. Shortly after the beginning of the project, we evaluated several high-level libraries for MapReduce, because the implementation of plain MapReduce workflows is tedious even for trivial tasks and requires the implementation of in-house libraries for recurring tasks. Available operators, operations, extensibility and a large number of supported data sources and formats, such as HBase[3] and XML were some of the requirements which led us to Apache

[1] http://pig.apache.org/
[2] http://hadoop.apache.org/
[3] http://hbase.apache.org/

Pig[4], a declarative data flow language which compiles into MapReduce and is easy to understand for developers and analysts due to its similarity to SQL. Pig proved as suitable both for smaller data crunching tasks and for the development of complex MapReduce workflows. Shortcomings like insufficient error reporting and limited modularization were significantly diminished during the project as new versions were released. The introduction of macros in version 0.9.1 allowed for a modularization of the project and better extensibility.

2.1.2 RapidMiner[5] extended by Weka[6]

The deliverable D3.1.1 introduced the two data mining frameworks that we intended to integrate in Dicode: RapidMiner and Weka. Both are very popular free data mining tools. Since version 5.0, RapidMiner offers the latest version of Weka 3.7.3 as an extension[7] which can be easily downloaded and installed. Therefore, all modeling methods and attribute evaluation methods from Weka are available now within RapidMiner. Moreover, about 400 operators with different preprocessing methods, validation and visualization techniques offered by RapidMiner are not available within Weka.

Due to the fact that RapidMiner includes all learning functionalities of Weka and outperforms it in terms of additional operators, we decided to use only RapidMiner in DDMF.

2.1.3 R[8]

Due to numerous packages and libraries, R can be seen as one of the best frameworks for analysis of genomic and microarray data and statistical computations. However, R is a programming language that is not very user friendly, particularly for users without specific programming skills. Taking the above into account for the needs of the project, the Dicode data mining services (Subgroup Discovery, Text Mining Services) use R internally in order to facilitate their usage.

2.2 Particular Dicode Data Mining Services

FHG's envisioned Sentiment Analysis service and Relation Extraction service (see deliverable D3.1.1) have not been implemented for the following reasons:

- Sentiment Analysis is more abstract than the implemented Emotion Detection Training service. From our point of view, the latter service can derive information that cover those expected from the former. It is possible to train the Emotion Detection Training service to detect positive and negative sentiments, if an appropriate text collection and a list of seed phrases are provided. The details are given in Section 2.2.4.
- The Relation Extraction service was not implemented because a state of the art survey and research done at FHG[9] revealed that the current capabilities of relation extraction

[4] Christopher Olston, Benjamin Reed, Utkarsh, Srivastava, Ravi Kumar, Andrew Tomkins: Pig Latin: a not-so-foreign language for data processing. SIGMOD Conference 2008: 1099-1110
[5] http://rapid-i.com
[6] http://www.cs.waikato.ac.nz/ml/weka/
[7] http://rapid-i.com/content/view/202/206/lang,en/#weka
[8] http://cran.r-project.org/

algorithms are limited to very special cases. The semantic type of relation to be extracted has to be defined in quite detail in advance. This renders ad-hoc investigations by analysts impossible.

2.2.1 Named Entity Recognition

The named entity recognition and disambiguation task deals with finding all names of people, places, brands, etc. in text, disambiguating these names and assigning unique identifiers to them which might represent ontology resources. In the first version of this deliverable, the implementation of a workflow for named entity annotation was assigned to FHG. In addition to the annotation of named entities, the identification of key phrases was suggested. During the course of the project, these two tasks were split and assigned to different partners: FHG now is responsible for phrase extraction and NEO took over the named entity recognition and disambiguation task. Both solutions have something in common: they make use of the machine learning method of Conditional Random Fields (CRF)[10]. FHG uses a CRF model for the detection of phrases whereas NEO uses CRF for the recognition of persons and places during named entity recognition.

In Dicode, the service for named entity recognition and disambiguation was designed for Big Data. The main achievement of Dicode's Named Entity Recognition Service therefore is high scalability. In the past, NEO could only perform named entity recognition and disambiguation (NERD) on relatively small document corpora. A sequential implementation processed two documents per second, which resulted in processing times of several weeks for typical corpora like archive exports which today often contain several millions of documents. Annotating larger corpora like Blog crawls was impossible and the only option was using less complex Gazetteer-based algorithms for named entity recognition and to refrain from named entity disambiguation in this case.

The named entity recognition and disambiguation task can be split into multiple steps which can be easily parallelized and other steps which rely on global resources. NEO's NERD algorithm relies on several lexical sources and ontologies which are used during different phases. The databases used in the past to hold these data were RDBMs and RDF stores. To exploit Hadoop's linear scaling capabilities, the newly developed distributed NERD (NERDist) could not use external services like these because they would sooner or later act as a bottleneck. This problem is solved by using Hadoop's distributed database HBase for both lexicon and ontology.

In the recognition phase, all "surface forms" stored in a lexicon are searched in the given text. For all surface forms found, entity candidates are retrieved that constitute possible disambiguations. Typically, there are many entity candidates for a given surface form (e.g. there are more than 20 Berlins in the world). For all candidates, there are prior probability estimations for them being the correct disambiguation. The lexicons used for Dicode are extracted from Freebase and Wikipedia. For the Dicode service, we use Wikipedia page links for estimating the prior probability of each candidate. Several probabilities describing the relation of surface form (anchor text) and linked page can be extracted from the Wikipedia

[9] Frank Reichartz, Hannes Korte, Gerhard Paass: "Semantic relation extraction with kernels over typed dependency trees", Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, p. 773--782

[10] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". (2001).

dump that help to produce associated statistic results (e.g. that the surface form "Berlin" refers to 94% of the cases to the capital of Germany).

To make sure that we even have probability estimates for newly created concepts in Wikipedia, we always use statistics generated from the most recent Wikipedia dump. This puts some performance requirements in place that were met after re-implementing the statistics computation in a distributed way on Hadoop via Apache Pig (see Section 2.1.1). Positive results from various experiences during the project resulted in choosing Apache Pig for other Hadoop tasks as well. The implementation of the distributed statistics generation (internally called "NERD stats") was published on Dicode's GitHub account[11] and has already been adopted by the open source Named Entity Disambiguation framework DBpedia Spotlight. Computing the statistics on Hadoop resulted in a huge speedup compared to the former sequential statistics generation method used at NEO[12]. Now, after a new Wikipedia dump is published, an alert is issued to the developer team which starts the computation of the statistics[13]. The computation finishes in the order of minutes, while in the past the runtime was several days.

2.2.2 Entity Prominence

Named entity annotations can be used for many different tasks in information retrieval and text mining. In Dicode, we decided to aggregate named entity counts and to visualize them via the newly developed Dicode Entity Prominence Service. Dicode's Entity Prominence Service provides several widgets which visualize the prominence counts for all Named Entities annotated in NEO's news corpus which contains more than 1100 news sources crawled on a daily basis. The Entity Prominence widgets provide great insights for social media analysts: they can, for example, monitor the performance of their brand over time and compare the number of occurrences in different sources. From a technical perspective, these widgets offer an opportunity to demonstrate the scalability of Dicode's Named Entity Disambiguation Service. They show at a glance that Dicode processed a huge amount of documents (approximately 50 million) and make use of the results. Currently, the aggregation of prominence counts is performed in batch mode. In a future version of the service, the Named Entity Disambiguation Service and the aggregations for the Entity Prominence Service will have to deal effectively with near real-time requirements and annotate all incoming documents immediately.

The starting point for the Entity Prominence Service is the output of the Named Entity Disambiguation Service which has been stored in HBase together with the document corpus in question. The Entity Prominence Service implements the functionality of a legacy component at NEO which was used for named entity aggregation. Formerly, there had been a MySQL database holding prominence statistics like named entity counts for different time frames (e.g. per month/day/year). The system was not scalable and could not deal with the number of documents processed in the Dicode project. Additionally, there was the requirement to aggregate counts for different dimensions like the language of a document and the domain of the document. Extensibility was another requirement: it should be possible to filter the aggregated counts for additional metadata.

[11] https://github.com/dicode-project/pignlproc/tree/master/examples/nerd-stats
[12] http://spotlight.dbpedia.org/
[13] We decided to use this semi-automatic approach because, depending on the load of the cluster and planned jobs, we can then opt for direct or nightly execution.

These requirements are typically met by an OLAP cube. The open source project "datacube"[14] offers the functionality in question, i.e. a multidimensional storage with rollups for numerical data. Datacube is implemented in Java and has a pluggable database backend. The default implementation is an HBase sink which perfectly fits into Dicode's infrastructure. Datacube was developed by the US based company Urban Airship for its analytics stack for mobile apps. NEO forked the project on GitHub and added slicing functionality which is able to display the top entities for instance. In Urban Airship's implementation, dimensions and rollups have to be implemented in Java which can be quite tedious for multidimensional datasets. For the Entity Prominence Service, an additional layer was implemented. Now, the configuration and use of the cube is more convenient: dimensions and rollups can be declared in the Spring[15] Application Context. In addition, the import of events has been strongly improved. The layer also contains a REST API which grants easy access to the data inside the OLAP cube. The visualization component was developed in JavaScript. RequireJS[16] helped modularizing the code and Google's Chart Tools[17] are used for the graph visualizations.

2.2.3 Service based on Latent Dirichlet Allocation[18] (LDA)

The LDA training algorithm is one of the basic FHG text mining algorithms that are used in the project.

- The **Topic Detection service** is based on LDA. It is configured to work on either uploaded texts or texts that are received from one of NEO's services. The LDA training algorithm provides two tables: one that relates topics and words, and one that relates documents and topics, both via probabilities. These tables are evaluated to provide a graphical display of topic interrelations. If meta-information on the documents (e.g. "author") is provided, this data can also be displayed in the topic graph. For details of the service interface, the reader should refer to deliverable D5.4.2 ("Integrated Dicode services (enhanced version)", page 86).

2.2.4 Services based on Conditional Random Fields[19] (CRF)

The CRF algorithm is the second basic algorithm used to empower the FHG's text mining services in Dicode. The CRF algorithm learns to identify phrases in a sentence from examples that have been pre-annotated manually. Historically, CRF was first used to identify names of persons, organizations and geographical locations. Research in the Dicode project however revealed that also the detection of more complex phrases can successfully be trained. Several of the FHG text mining services are based on the CRF algorithm:

- The **Phrase Extraction Training service** allows training a CRF on example phrases and example texts that have been supplied by the user. The resulting CRF models have to be downloaded and stored by the user for further use with the Phrase Extraction Application service.
- The **Phrase Extraction Application service** extracts phrases from a text collection, using a pre-trained CRF model. The model can either be supplied by the user (via

[14] https://github.com/dicode-project/datacube
[15] http://www.springsource.org/
[16] http://requirejs.org/
[17] https://developers.google.com/chart/
[18] Blei, DM, Ng, AY, Jordan, MI, Latent Dirichlet Allocation, The Journal of Machine Learning Research, 2003 vol. 3 pp. 993-1022 – see also deliverable D3.2.1, page 12
[19] Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proc. of Machine Learning Conference, 2001, pp. 282–289.

uploading), or the user can choose one of the models that have been integrated into the service for demonstration purposes.

- The **Emotion Detection Training service** can be used to train a CRF to extract phrases that convey emotions. In contrast to the Phrase Extraction Training service, it works on a text collection and a list of so called "emotional seed terms" provided by the user. The service then extracts the actual emotional phrases from the text collection and uses them for training. The resulting CRF model can then be transferred to the Phrase Extraction Application service for use on a new text collection.

2.2.5 Opinion Mining

The Opinion Mining service is based on a combination of the CRF and LDA algorithms. It first detects phrases in a text collection using a pre-trained CRF model. 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. In case that the phrase detection model was trained on an emotion, these sentences will contain opinions of the text authors that are related to the topic in question.

2.2.6 Subgroup Discovery

The Subgroup Discovery Service (introduced in D3.1.1) aims to find new previously unknown local patterns (subgroups) in the given dataset.

At the beginning of the Dicode project we have shown that subgroup discovery cannot be efficiently parallelized using MapReduce because the algorithm cannot be broken down to independent trivial tasks. Therefore, this service is an important example of the in-memory processing. In-memory processing enables an efficient parallelization on the thread level. For better performance, the implemented Subgroup Discovery algorithm exploits the complex in-memory database based on the special data structure called FP-Tree. An FP-tree is a compact data structure that represents the data set in tree form. Such data representation enables one to reduce both running time and memory size requirements of an algorithm.

2.2.7 Similarity Learning

Task 4.2 of WP4 deals with recommendation mechanisms that navigate a user through a large space of possible choices. In the context of this task, deliverable D3.1.1 introduced the following services:

- Basic similarity services;
- Distance Metric Learning, and
- Application service.

During the implementation phase, the proposed architecture of services was slightly restructured in order to better meet the task requirements and satisfy user needs. The distance metric learning is a special case of similarity learning. The distance value can be easily recalculated into the similarity values. Due to this fact, and in order to keep it easy-to-use, we decided to develop one similarity learning service. The aforementioned services were incorporated into the Dicode recommender system which consists of the two following components:

- Similarity Learning, which creates a similarity model from the user preferences. The model is then used by the Recommender Service to create user specific recommendations.
- Recommender Service, which applies a learned model to a large number of items in order to find the most similar ones. The service provides the user with relevant and interesting information according to his preferences.

Since the ability to compute basic similarities is an important prerequisite for similarity learning service and recommender service, we included such functionality into both of them. The following basic distance measures were implemented[20]:

- Levenstein Distance;
- Block Distance;
- Dice Similarity;
- JaroWinkler Similarity;
- Matching Coefficient;
- Jaccard Similarity;
- Cosine Similarity;
- Chapman Length Deviation;
- Euclidean Distance;
- Jaro;
- MongeElkan Distance;
- NeedlemanWunch Distance;
- Overlap Coefficient;
- QGrams Distance;
- SmithWaterman Distance;
- SmithWatermanGotoh Distance;
- SmithWatermanGotohWindowedAffine Distance, and
- Soundex Distance for English.

While none of the measures by itself is applicable to all conceivable scenarios, an appropriate combination of them enables to compare complex items described by heterogeneous information: text, structural information or numerical data. The aforementioned basic distances/similarity measures cover the wide range of different data types.

3 Lessons Learned

3.1 Hadoop

During the last years, "Big Data" has become a mainstream topic. Apache Hadoop has been integrated into major software solutions like IBM's InfoSphere BigInsight[21] and Oracle's software stack[22]. In the open source segment, close-to-machine implementations like "pure MapReduce" approaches were partly replaced by higher-level abstractions like Apache Pig, Hive[23] or Cascading[24]. High scalability comes at a huge price: development and operations of

[20] Felix Naumann and Melanie Herschel. An Introduction to Duplicate Detection. Synthesis Lectures on Data Management
[21] http://www-01.ibm.com/software/data/infosphere/biginsights
[22] http://www.oracle.com/us/technologies/big-data/index.html
[23] http://hive.apache.org/
[24] http://www.cascading.org/

Hadoop based applications is inherently hard and requires a lot of training. As the ecosystem became more and more complex, pre-packaged distributions of all Hadoop related components have become increasingly popular. Cloudera's CDH, Hortonworks' Data Platform and MapR are three major Hadoop distributions available.[25] Although the core of the distributions is open source, proprietary tools, e.g. for managing software distribution and configuration on large clusters, are becoming more and more important. Especially for SMEs, this situation is quite challenging. Additional to huge investments in development skills, they also have to invest in system engineering know-how – otherwise they risk a vendor lock-in. Considering the huge investment, the initial euphoria for Big Data technologies seems to have vanished. Companies think twice before opting for a Hadoop-based solution: does their project really deal with big data and will better hardware do the same job? Today, most people working in the area of big data seem to agree that technologies like Hadoop solved a number of problems but created lots of new challenges.[26] Especially for SMEs, entry obstacles are still quite high and return on investment is achieved rather late.

Building up Hadoop expertise

Dicode is NEO's first Hadoop related project and there were lots of lessons learned. Two project members were certified as "Cloudera Certified Developer for Apache Hadoop" and helped to speed-up the skill acquisition in the first part of the project. During the project, NEO had furthered their significant skills in this area. In addition, the setup of a distributed system like Hadoop is far more costly than the setup of loosely coupled services on single nodes. When developing for Hadoop, operations and development are closely linked – especially in debugging and optimization of MapReduce jobs. Development for Hadoop is going to be an area which relies on strong integration of software development and operations as proposed by the DevOp[27] movement. For the operation of the Dicode cluster, NEO had to build up DevOp skills in this area, which were not previously available in the company.

Lessons learned in Hadoop configuration

During the second year of the project, NEO joined forces with another publicly funded project, MIA[28], which develops a cloud-based Marketplace for Information and Analytics. Together, the two projects are now operating a development cluster which consists of three Hadoop nodes provided by Dicode and nine additional nodes provided by NEO's MIA team. The new setup of the development cluster offers a more realistic setup of Hadoop and HBase. The initial three-node-cluster had been configured with a replication factor of three, which resulted in untypically high replication load for each node especially during data imports and HBase compactions. Now, the data is distributed to 12 nodes which lead to a higher (and more typical) network traffic within the cluster.

[25] Cloudera, a Palo Alto based company which offers a commercial and a free version of its distribution CDH4, seems to be the winner concerning the acquisition of venture capital. In December 2012 the company announced the raise of $65 Million of funding and the set-up of a European headquarters in 2013. http://www.marketwire.com/press-release/Cloudera-Raises-65M-to-Accelerate-Enterprise-Growth-1734798.htm

[26] See Isabel Drost's open collection on experiences on the topic "How to fail your big data project quick and rapidly" at http://titanpad.com/MVrCNj06Sn

[27] http://en.wikipedia.org/wiki/DevOps

[28] http://www.dima.tu-berlin.de/menue/research/current_projects/mia/

For the development cluster, NEO uses Cloudera's Hadoop Distribution CDH[29]. Starting from version 4 (CDH4) of the distribution, Cloudera offers the YARN[30] framework ("Yet-Another-Resource-Negotiator"), a generic platform for the development of distributed applications. For YARN, a new version of the MapReduce framework was implemented. The old framework is now called MR1, whereas the new version is called MR2. In addition, Cloudera offers the "Cloudera Manager", a proprietary installation and configuration utility which comes in a free and a commercial version. In the free edition, Cloudera Manager supports the installation of up to 50 nodes, but does not offer any versioning of configuration. Switching between MR1 and MR2 on a cluster is a one-click operation with this Manager, but it also requires the deployed applications to be MR2 compatible. NEO decided to move to YARN - although it was still a beta version - to be able to profit from the strong community development effort in this branch. After some weeks of evaluation, we decided to go back to MR1, because - especially under heavy load - many MapReduce jobs failed with errors which previously did not appear.

In the long run, NEO strives at replacing Cloudera Manager by an automated installation and configuration management based on Open source tools like Puppet[31] because the usages of a proprietary solution like Cloudera Manager is far too risky: the free version lacks essential features like versioning, whereas the commercial version would be an investment barrier especially for SME customers. Another advantage of the usage of Puppet is its seamless integration into the company infrastructure.

Towards real-time processing

As near real-time requirements become more important, future systems will have to combine batch and stream processing efficiently. Sometimes Hadoop reminds us of programming in the punch card era: we have to wait for the result (or failure) of a Hadoop batch job for hours. Besides being an obstacle for efficient software development and debugging, the latency of batch processing also interferes with the requirement for "freshness" which is at the core of many information processing applications: in social media monitoring, the user demands an instant alert if there is a new report about the brand or event in question; in a news search, each article has to be analyzed and annotated with automatically generated meta data before being indexed for search. In both cases, a delay of several minutes is not acceptable.

In the final months of Dicode, NEO wants to tackle these issues. We want to find out how we can keep the advantages of batch-style distributed systems like Apache Hadoop when dealing with near real-time requirements. In the literature, a combination of batch and stream processing frameworks is suggested, which combines shallow processing of incoming data and in-depth processing of batches of data for final results.[32] In Dicode, we want to evaluate the techniques for low latency document analysis. The envisioned architecture combines batch and stream processing. Large numbers of documents will be processed in batch style on Apache Hadoop. Additionally, a fast lane will process high priority documents immediately. This architecture adds additional complexity to the already challenging Hadoop-only solution. Light-weight event processing frameworks like Storm

[29] http://www.cloudera.com/content/cloudera/en/products/cdh.html
[30] http://blog.cloudera.com/blog/2012/10/mr2-and-yarn-briefly-explained/
[31] https://puppetlabs.com/
[32] See, for example, Nathan Marz and James Warren: Big Data - Principles and best practices of scalable realtime data systems. Manning Publications, Early Access Edition, http://www.manning.com/marz/

also might fill a gap in big data scenarios and serve as an easier solution for large-scale text mining.

Another key issue in the development of highly scalable document processing systems is search. Open source search solutions are increasingly capable of dealing with big data requirements. Recently, several frameworks for highly scalable indexing and search have been published, which allow for the indexing of multiple terabytes of data. SolrCloud[33] adds distributed capabilities to the Lucene-based search engine Solr. ElasticSearch[34] is another Lucene-based solution for highly scalable search. With the increasing scalability of search solutions, tasks formerly performed with batch processing systems like Hadoop will be re-integrated into search solutions. A state of the art document processing solution therefore has to choose its components carefully depending on the use case. Due to the advanced state of Dicode, we most likely have to postpone the implementation of a hybrid document processing system to a follow-up project.

3.2 Text mining

After the start of the Dicode project, it became clear quickly that the text mining algorithms needed for central tasks in use case 3 (CRF and LDA) would need to be trained on example datasets. This turned out to be a seriously limiting factor on data throughput, because until today no parallel implementations exist that would be easy to integrate in the Dicode framework. An evaluation of the Mahout framework had a negative result (see deliverable D3.1.1). CRF models needed for the Named Entity Recognition service could be trained off-line, but CRF as well as LDA models needed in the other services have to be re-trained ad-hoc and on-line to fit the requirements posed by the ongoing collaboration on the Dicode Workbench. We therefore changed the analysis strategy applied in Use Case 3: services that do not need an on-line training step are used to select a relatively small collection of texts (in the order of 10.000 documents) that are processed by the time-critical services. For example, the Entity Prominence service can be used to select brand names and time intervals for an in-depth analysis. This reduces the number of documents to be processed by the Topic Detection service. The topic graph can then be used to identify an even more reduced set of products to be compared. Finally, the texts dealing with only those products can be processed to detect emotions and opinions related to the products.

4 Conclusions

During the development and integration of data mining services, we compared many different technological solutions and platforms. Computational demands of each service are very different. Therefore, we experimented in Dicode with different solutions regarding the most important dimensions, such as storage, speed and human factor. While Hadoop has been proved as a suitable technology for text mining problems, in-memory processing method has been shown as a good solution for the subgroup discovery. A solution for the human bottlenecks was demonstrated in the example of similarity learning service. We have also shown that an intelligent sampling strategy enables one to reduce human efforts on labeling data.

[33] http://wiki.apache.org/solr/SolrCloud
[34] http://www.elasticsearch.org/