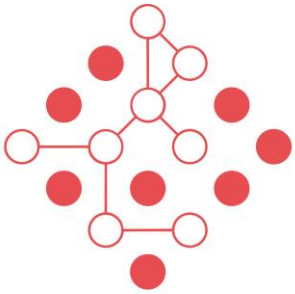

DATA MARKET

AUSTRIA

www.datamarket.at

Broker and Assessment Technology Specification and Development Road

Deliverable number	*D7.1*
Dissemination level	*Public*
Delivery date	*31 May 2017*
Status	*Ready for Review*
Authors	*Matthias Traub (KNOW)* *Heimo Gursch (KNOW)* *Mihai Lupu (RSA)*

FFG

The Data Market Austria Project has received funding from the programme “ICT of the Future” of the Austrian Research Promotion Agency (FFG) and the Austrian Ministry for Transport, Innovation and Technology (Project 855404)

bmvit

Executive Summary

This document presents the DMA Broker and Assessment Technology Specification and Development Road regarding the ongoing work in WP7. The document relates to deliverable D2.2 “Community-driven Data-Services Ecosystem Requirements (1st version)” which formulates the requirements of the DMA community as well as the two other technical deliverables D5.1 “Data Technology Specification and Development Roadmap” and D6.1 “Service Technology Specification and Development Roadmap”.

WP7 has three focus areas, namely information extraction, the scalable recommender framework (ScaR), and the service evaluation components. The information extraction receives incoming data from other DMA components, including metadata of datasets and services, interactions of users on the DMA platform, and user profiles. The information extraction translates the incoming data from all components into structured data interpretable by the recommender framework.

The scalable recommender framework will be based on the ScaR framework developed by Know-Center. The ScaR framework is ported into a Docker-based architecture to fit in the general DMA core architecture. The ScaR framework will provide suggestions to users of the DMA portal about possible combinations of datasets and services. These suggestions will be derived from the metadata of datasets and services combined with the interactions of users with the datasets and services. Additionally, the ScaR framework will then be able to provide a search service over the metadata of datasets and services available within the DMA.

In terms of evaluation of DMA Services, dedicated evaluation components will be generated based on existing technologies, but encapsulated as DMA Services themselves. From an existing technology based on Swarm and Docker connected via NFS, we will be moving to the DMA architecture plan based on Kubernetes and CEPH. Initial evaluation components will be provided for basic services (ranking, clustering, and classification) based on existing test data available to the academic and commercial world. This initial step will define the metadata necessary to specify a test collection (including a ground truth component).

The deliverable gives an introduction of the work package job description, its goals and obstacles in Chapter 1. Chapter 2 is highlighting the state of the art of related work and how these technologies and solutions are picked up for the development of our services in terms of traditional/open/ontology based information extraction, various recommender approaches and evaluation methods. The technological foundation, which is used as a starting point for developing the services, is described in Chapter 3. This chapter also highlights potential candidates for IE software. Further, the scalable recommender framework (ScaR) is described in detail. With its micro service based architecture, ScaR makes it very easy to connect to all the other central services developed in other work packages. The third part describes the Docker based modules for service evaluation. Chapter 4 is picking this up and describes the proposed system architecture with the underlying metadata schema and the respective services. It depicts the connection points to the DMA Portal, the service and dataset management services. Chapter 5 provides an outline of the development roadmap which is divided into two main phases: the first being *prototype development*, resulting in deliverable D7.2, and the final development concluding with the third deliverable D7.3. The whole development will be in close collaboration with partners of other technical work packages to ensure the compatibility and fixed release dates.

Table of Contents

List of short names and abbreviations

API	Application Programming Interface
CB	Content Based Recommender Approach
CEPH	Name of a distributed object storage platform
CF	Collaborative Filtering Recommender Approach
DMA	Data Market Austria
DML	Data Manipulation Layer
GATE	General Architecture for Text Engineering
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Information Extraction
MAE	Mean Absolute Error
MALLET	MAchine Learning for LanguagE Toolkit
MLT	MoreLikeThis Recommender Approach
MP	Most Popular Recommender Approach
MSE	Mean Squared Error
NER	Named Entity Recognition
NFS	Network File System
NLP	Natural Language Processing
OBIE	Ontology Based Information Extraction
OIE	Open Information Extraction
PDF	Portable Document Format
POS	Part-Of-Speech
RDF	Resource Description Framework
ROC	Receiver Operating Characteristic
SCAR	Scalable Recommender Framework
UIMA	Unstructured Information Management Architecture
WP	Work Package
WWW	World Wide Web

1 Introduction

The heart and soul of DMA will be the lively interactions of dataset and service providers in combination with their customers. With an increasing number of different offers available on DMA, it will be progressively difficult for customers and providers to find the partners and offers they are looking for. WP7 will offer a recommendation service as part of the DMA core components to foster the interactions between providers and customers. The recommendation service, or recommender for short, will be designed to suggest possible collaborations of different actors on the DMA platform. Hence, the recommender will facilitate the brokerage by making users aware of the content available on DMA by its suggestions.

The recommender will derive the suggestion from different sources, namely metadata describing data sets and services, past interactions of users recorded on the DMA platform, and automated assessments of service performances. From each of these sources, features describing the services and datasets on DMA will be extracted. These features are the basis to calculate the personalized recommendations presented to users of the DMA platform.

The work in WP7 is partly based on the Know-Center Scalable Recommendation Framework (ScaR)[1]. The ScaR framework provides the core recommendation functionality and will be adapted to the requirements for serving recommendations in the DMA portal. Additionally, the ScaR recommender has to be integrated into the DMA core services. The integration and the adaption are both considered engineering-oriented work. They are the foundations for the research oriented work, both being depicted in Figure 1. The research oriented tasks are focused on designing and testing novel functionality not already covered by ScaR or similar products. One aspect of these research oriented tasks is the tools for measuring the effectiveness and efficiency of services. Together with the information extraction and the metadata enrichment, these two tasks provide the necessary tools to generate the knowledge base for the recommender. The recommender combines knowledge from all sources to adaptively generate personalised recommendations for each user and fosters the brokerage on the DMA platform. The unification of the different knowledge sources will be

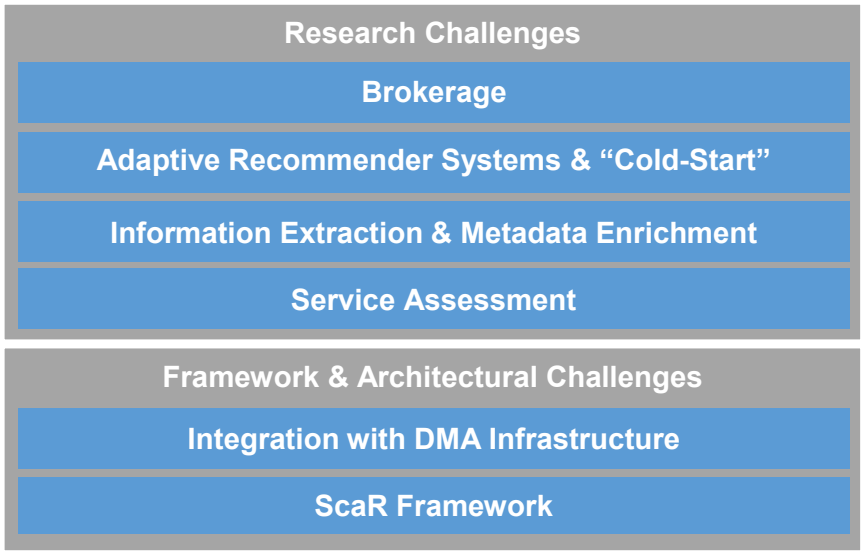

Figure 1: The work in WP7 can be divided in research-oriented work (research challenges) and engineering-oriented work (Framework and Architectural Challenges).

[1] http://scar.know-center.tugraz.at/

investigated on its potentials to overcome the cold start problem and achieve a better personalisation of the recommendations.

1.1 Modules and Interfaces

WP7 uses different data sources to calculate the recommendations, which are finally delivered to the DMA portal and seen by the users. Hence, WP7 provides a core service for the recommender communication with other core services. Figure 2 depicts the interactions of WP with the other parts for DMA:

- The **Service Providers** offer metadata about each of their individual services available via DMA. The metadata describe the requirements and functionality of each service. Hence, the metadata is used by the recommender to categorise individual services.
- Each **Data Provider** describes published datasets with metadata. This metadata is used by the recommender to categorise a dataset. Hence, not the actual data in a dataset but just the metadata is of interest to the recommender.
- The **Interactions** of platform users with datasets and services are an additional input for the recommender. These interactions come from users interacting with items on the DMA portal. Such interactions might be clicks on items to see the detailed metadata information of a service or a dataset, the purchase of a licence to use a service or a dataset, or the publishing of own services or datasets on the platform.
- **Users** on the DMA platform are described by their own user profile consisting of interactions with datasets and services.
- **Recommendation Interfaces** offer the derived suggestions to the frontend, i.e. the DMA portal. These interfaces provide personalised recommendations for the presentation on the DMA portal to the user.

In the middle of the architecture shown in Figure 2 are the three core components of WP7: the matchmaking Framework, the Tools for service assessment and the recommender.

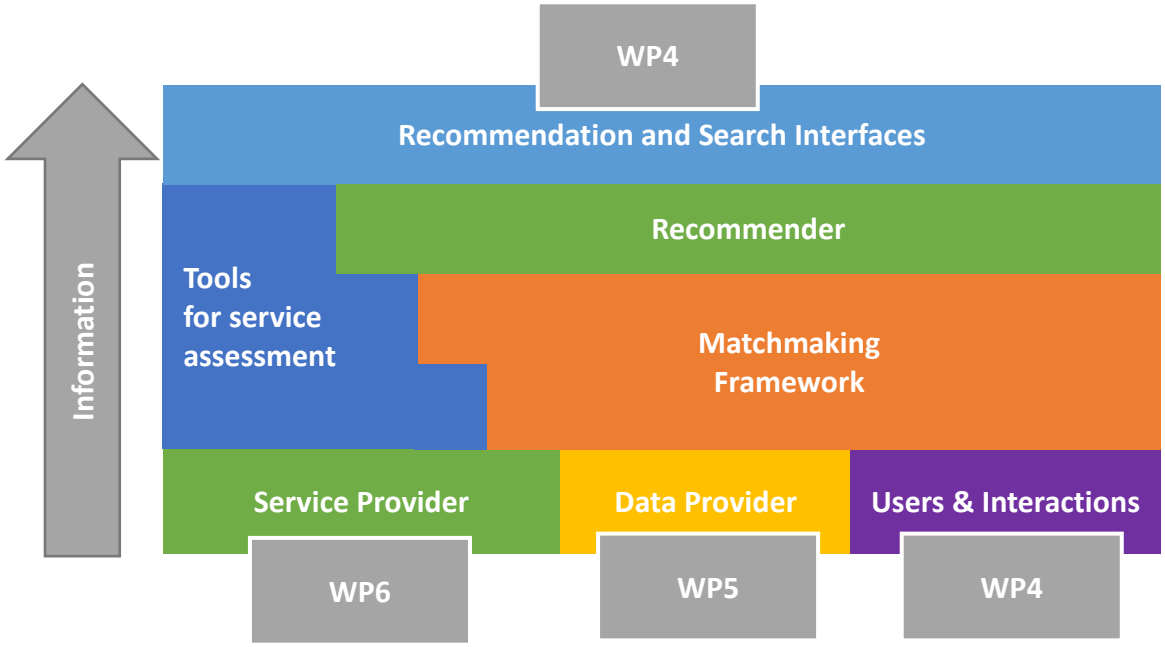

Figure 2: WP7 uses input from WP4, WP5, and WP6 to generate the recommendations. The Recommendations are then provided to the user in the portal maintained by WP4

1.1.1 Matchmaking Framework

The Matchmaking Framework can be interpreted as the platform for the recommender. It established the connection to all three data sources, namely the metadata describing services, the metadata describing datasets, and the user interactions recorded in the web portal. Data from each source is passed through a so-called information extraction stage. In this stage, the incoming data is transformed, filtered, and enriched to fit the data model of the recommender.

1.1.2 Recommender

The recommender is the core of WP7s and is building on the data sources connected by the matchmaking framework. The recommender faces the problem, that there are no or very few interaction data available if a new service or dataset is added to DMA which leads to a cold-start problem. In this phase, the recommender will work mainly based on the metadata describing datasets and services. This modus of operation is called content-based. Later on, with more and more interaction data available describing the users' behaviour for a given service or dataset, the recommender will move away from solely content-based recommendations and incorporate the transaction data as well. Recommender approaches relying completely on interaction data are referred to as using collaborative filtering; recommenders based on content and transaction data are known as hybrid recommenders. The DMA recommender follows the hybrid model.

1.1.3 Tools for Service Assessment

To assess the suitability of services for a specific task, two different aspects need to be considered. First, the output generated by services needs to be evaluated in terms of task-specific efficiency and effectiveness. Second, to asses a complete process consisting of one or more datasets and services, specific and significant evaluation metrics are needed. The requirements for metrics will be examined as part of the development plan and as described further down in this deliverable. The metrics will be developed and made available as services.. Guidance for the users in selecting evaluation metrics and protocols suitable to their task is provided, both in the form of clear documentation, but also through semi-automated recommendations.

2 Related Work

2.1 Information Extraction (IE)

Information Extraction (IE) deals with the problem of extracting structured and meaningful information from potentially very large amounts of unstructed information. In other words, IE tries to extract machine interpretable facts from unstructured data [1]. In WP7, the information extraction will work on the metadata describing services and datasets. The metadata of services and datasets are organised as a list of name-value pairs. The values can be of structured or unstructured nature:

- **Structured values** are limited to a set of pre-defined values, e.g. type of service, licence type. Hence, these values can be processed by the recommender without any further refinement. The IE might handover structured values directly to the recommender or translate these values with a given mapping (e.g. translation of keywords in different languages).
- **Unstructured values** allow the users to write free text, e.g. description of a dataset or service. The free text needs to be processed by the IE to transform it to a machine interpretable representation. The result of the IE will be structured values capturing the essence of the text and making the texts comparable to each other.

In general, two different types of IE are distinguished. Traditional or closed IE and, open IE. While traditional IE deals with a pre-specified and closed vocabulary, open IE does not require this pre-specified vocabulary [2]. The IE process is usually ordered in two stages. In the first stage entities get extracted (e.g. locations or organisations) while the second stage relations between these entities are discovered (e.g. place of residence of people) [3].

2.1.1 Traditional Information Extraction

In traditional IE, the target vocabulary to extract from unstructured text is defined in advance. These systems usually work on homogenous corpora of text with predefined extraction targets. With the creation of lexical databases such as WordNet [4], the required efforts to create an IE database and the spectrum of possible extraction targets was considerably extending. The simplest approach is so called Gazetteer lists holding all words which should be extracted. Also, if a word has multiple spellings or grammar forms, all these forms have to be on the Gazetteer list. The IE simply searches for the words on the list in the texts and annotates all occurrences.

In the early 1990 rule based approaches were established. The rules were hand crafted textual patterns taking into account the surrounding contexts of syntactical and grammatical characteristics. Hindle [5] presented an approach based on predicate-argument structures. Hearst [6] derived lexicostatistical patters to extract hyponymy relations. The large effort to craft and maintain the extraction rules meant that the work was limited to small, predefined domains. But even in such scenarios the derivation of the extraction rules is heavily time consuming. These drawbacks lead to the advent of open IE.

2.1.2 Ontology based Information Extraction (OIBE)

Alternative methods are for example ontology based IE (OBIE) [7]. Predefined ontologies (i.e. specifications of shared conceptualisations) are used to derive information from text origination from a certain domain. Some of these systems rely on rules and information provided by the grammatical structure of the underlying text to extract information. This allows with predefined patterns to extract contextual information of noun phrases. One challenge in OBIE is to explicitly formulate the predefined knowledge as an ontology. Aside from the possibilities to generate this information by hand, several systems where presented that rely on bootstrapping to generate new ontologies based on a set of predefined

ontologies [8] [9]. Meta information existing in the Wikipedia corpus or linguistic information from the WordNet is used to automatically or semi-automatically derive new ontologies.

2.1.3 Open Information Extraction (OIE)

Open Information Extraction (OIE) aims at overcoming the limitations of traditional IE. The idea was popularised by Banko et al. [2] to apply IE on data from the Word Wide Web (WWW). Since the WWW covers many domains, the relations of interests are potentially unknown, and the WWW contains billions of potential web pages, traditional IE methods are not feasible to use. OIE needs to be domain agnostic. Hence, it relies on grammatical and syntactical features of the text as an input for heuristics, rules, and machine learning algorithms to extract the information. This means, that OIE is language dependent. Most current work is done for the English language. Often the information extraction approaches in Natural Language Processing (NLP) are based on statistical linguistic features, for example the frequencies of two, three, or more consecutive characters (often referred to as n-gram frequencies) or terms. Typically, these features are further used in the combination with machine learning techniques to create an OIE system.

2.2 Recommender Systems

In recent years, recommender systems have become an integral part in many applications (such as Amazon, Netflix and MovieLens) that support users in finding interesting information in an overloaded information space. Although the first recommender systems were developed already in the mid-1990s, the interest in this research area still remains high due to its problem-rich characteristics. The main problem that is tried to be solved by recommender systems can be formulated as follows: Suggest for a user *u* a set of domain-specific items *I* (e.g., books, movies, music, etc.) that could be of interest for *u* and at the same time is new to *u*. In order to determine this set of items *I*, several recommender approaches have been developed that are presented in the following section [10].

In the past decade, the field of recommender systems experienced a vast amount of research, mostly focusing on developing novel recommendation approaches [11] and improving recommender accuracy [12]. Thus, many well-known methods are available, such as Content-Based Filtering [13], Collaborative Filtering [14] or Matrix Factorization [15], all having their unique strengths and weaknesses. For example, Content-Based Filtering approaches require meaningful content features to be encoded and struggle with serendipity. Traditional Collaborative Filtering approaches suffer from cold-start and sparsity problems, as well as scalability issues when dealing with large amounts of data. Algorithms based on Matrix Factorization currently dominate the literature, but are typically highly expensive in terms of computation time and are impractical to consider the user's change of interests in real-time.

As traditional recommender systems mostly focus on a single domain (e.g., marketplace, hotel, conference, etc.), they adapt and apply those approaches, which suits the available domain model the best. In the end the approaches are utilized depending on how easily (or hard) a recommendation approach can be adapted to the existing domain model. As the prediction task is usually viewed as a two-dimensional problem, user-item interactions are the easiest to apply, and are therefore the most commonly utilized data feature.

However, supporting a diverse set of domains becomes an important issue in modern recommender systems. One recent work going into that direction is Kibitz [16], an online recommendation system builder. With Kibitz, users can currently upload their domain specific item data and get a recommender system, which provides recommendations based on rating data (i.e., via MP or CF). Since DMA is recommending tuples (services and datasets) this is of advantage since the two can be seen as two domains with different intrinsic relationships.

With the arrival of the big data era, recommender systems are expected to analyse lots amounts of data, to support various data types and to handle streams of new data (i.e., volume, variety and velocity defining the Big Data problem). In such settings, traditional single-domain recommender systems usually analyse the data offline and update the generated model in regular time intervals. However, in many domains, choices made by users depend on factors which are susceptible to change over time. Let's take a shopping mall for example, where a user triggers frequent indoor location updates via a smartphone application while moving through the mall.

Employing an offline model update strategy that lasts hours or days may potentially miss the current location context of the user and fail to provide the right recommendations to match the users real-time demand.

As a consequence, being able to capture users' real-time interests are gaining momentum. This is the case especially in the industry sector, where practical real-time recommender systems are currently of high demand [17], [18], [19], [12], [20], [21].

2.2.1 Recommender Approaches

The literature distinguishes between three types of recommender approaches, namely (i) content-based, (ii) collaborative, and (iii) hybrid methods. These three types of approaches not only differ in the underlying algorithms but also in the data that is processed [22].

2.2.1.1 Content-based Methods

Content-based methods try to recommend items with similar content features as given in the profile of a given user. In this respect, the user profile could consist of (i) demographic information, such as the age or the interests of the user, (ii) the content of the items the user has liked in the past, or (iii) a combination of both. In cases where the user profile and the item profile share the same features (e.g., shared categories or social tags), the recommender can directly try to match users and items based on the profile information. In contrast, if the user profile is given by items the user has liked in the past, the recommender tries to find similar items to those and combines the individual results into a combined list of recommendations [23]. One of the best-known measures for determining the similarity between item profiles, is the term frequency / inverse document frequency measure (TF-IDF) that aims to calculate the importance of a term (i.e., a keyword in a metadata field of the item profile) for a document (i.e., the item profile) [24].

However, one limitation of content-based recommenders is that they are limited by the features they are able to extract from the items and user profiles. Thus, only items can be recommended that are annotated with sufficient metadata. Additionally, they tend to recommend items that are very similar to the ones the user already liked and thus result in recommendations which lack of diversity and novelty, which is still an open problem in the recommender research community [22]. Another use case for content-based methods is to determine alternatives for a given item based on the item profiles (i.e., "More like this").

2.2.1.2 Collaborative Methods

In contrast to content-based methods, collaborative methods (also referred to as Collaborative Filtering in the literature) analyse the interactions between users and items and recommend those items to a given user that similar users have liked in the past. In order to boost novelty, this is often extended to recommend items that are at the same time new to the user. More specifically, in collaborative methods two users are treated as similar if they have liked the same items in the past. This in turn allows us to assume that these two users will also like the same (or similar) items in the future. The similarity between users in collaborative methods is typically determined by a correlation measure, such as the Cosine or Pearson correlation coefficients [14].

One limitation of collaborative methods is that they suffer from sparsity, i.e., only items can be suggested that have already been liked by some of the users in the system [22]. One of the latest strands in recommender research termed as Matrix Factorization, aims to overcome this problem by using a dimensionality reduction technique such as Singular Value Decomposition (SVD) [25]. Cold-start users are another problem for CF. Because of that, other data sources should be utilized. [26].

Additionally, collaborative methods are often used to realize the “Users who bought this also bought that” use case known from online marketplaces like Amazon.

2.2.1.3 Hybrid Methods

In order to overcome the limitations of content-based and collaborative methods and to combine various sources of data, hybrid recommendation methods have been developed. Typically, hybrid recommenders combine the recommended items of various algorithms in the form of weighted, linear combinations [27].

However, research has also come up with integrated combinations of content-based and collaborative methods, e.g., by calculating the user similarities in a collaborative method using a content-based approach [22].

2.3 Brokerage

The “broker” concept is usually associated with an intermediate person involved in commercial negotiations or transactions. Specifically, a broker might be an agent who buys stocks, land or any other kind of goods or assets and sells them for others [28]. From a high-level perspective, the same principle can be applied when facilitating the mapping between offerings and demands of data and services in the DMA.

The state of the art analysis implicates that there are only few research works that built upon the basis of recommender systems and data/service brokerage. The work presented in [29] is among the few that addresses not only the issue of service brokerage between providers and consumers but also the issue of data brokerage. This work proposes a context-aware cloud service brokerage framework as a mechanism which enables the inter-mediation between Software-as-a-Service (SaaS) clients and providers, and supports the process of data integration among SaaS providers. The framework described in [29] has four different components including specification management, service recommendation, service discovery, and context fusion and observation.

The study conducted in the field of personalized media content recommendation [30] makes use of recommender and negotiation services for media content producer and distributor businesses. In this context, the perspectives of business-to-business (B2B) and business-to-consumer (B2C) models are combined. The nature of the relationship between content producers and content distributors is B2B, whereas the nature of the relationship between distributors and viewers is B2C. As a result, a brokerage platform is implemented, which enables the negotiation of the media items on behalf of the media distributors and producers and, additionally, provides viewers with a personalised electronic programme guide (EPG) containing the set of recommended items after negotiation.

Similarly, in [31], researchers present a method and a tool for optimising the cloud service usage, by allowing for cloud service evaluations based on a heterogeneous model of service characteristics.

2.4 Evaluation

The practically unlimited types of services that will be offered on top of the DMA infrastructure make a full description of related work impractical for the purposes of this deliverable. Instead, we shall group here the services based on the general technology area to which they belong:

- Information Retrieval
  - Ranking
- Recommender Systems and Machine Learning
  - Unsupervised Learning
  - Classification
  - Regression

2.4.1 Evaluation in Information Retrieval

In Computer Science, Information Retrieval has developed the most rigorous procedure for evaluating results where the human factor plays an essential role (i.e. relevance assessments associating a document to an information need).

Since the 1960s, even before Information Retrieval was operationalized in computers, efforts have been undertaken to assess in a reliable and reproducible manner the effectiveness of indexing methods [32]. The so-called “Cranfield paradigm” introduced by Cleverdon has been the fundament on which practically all information retrieval evaluation campaigns such as TREC [33] have built upon.

Essentially, the paradigm defines three components necessary for the evaluation of effectiveness in an IR system:

1. A (fixed) document collection
2. A set of information needs (aka topics, queries) that represent typical information needs of a typical user.
3. A relation defined on the set of documents and the set of information needs indicating whether the document is relevant or not (or to what degree) for the information need.

Together, the three items are referred to as a “test collection”. Note that the last item is a relation and not a function, because it is not fully defined. With the exception of the first test collection, that created by Cleverdon at Cranfield, all test collections are too large to have a complete definition of relevance over the cross product of documents and queries.

The use of test collections has been extensively studied and is now the agreed upon best-practice for typical retrieval tasks (so called ad-hoc retrieval, where the system is required to provide a ranked list of relevant documents for each query, without further interaction or guidance from the user) [34].

Systems to actually support evaluation have been developed and proposed to the community (e.g. EvaluateIR system [35], DIRECT [36], Lucene Open Relevance) with limited success.

More recently, renewed efforts went in the direction of using wildly popular open source libraries Lucene4IR [37] and component-based evaluation [38]. DMA builds upon these new efforts.

2.4.2 Recommender Systems and Machine Learning

Under this title we understand a whole range of procedures that take in a dataset and return a model that allows us to identify patterns in the data as a whole or predict numerical or categorical labels for new data items (i.e. unsupervised learning, regression, classification). When the data and the algorithms’ output are used to understand the nature of real-world phenomena, we call it Data Science.

For each of the three groups of methods mentioned above, evaluation procedures exist. For unsupervised learning, notably clustering, there exist both ground-truth based methods (also referred to as external assessments) and blind methods (also referred to as internal assessments). The challenge in this context is that machine learning and recommender systems have been applied in a wide area of domains without having a central point, or even a central practice, to guide the evaluation. Nevertheless, Figure 3 summarizes the Big Data Pipeline, aligning it to the machine learning methods terminology and with the typical

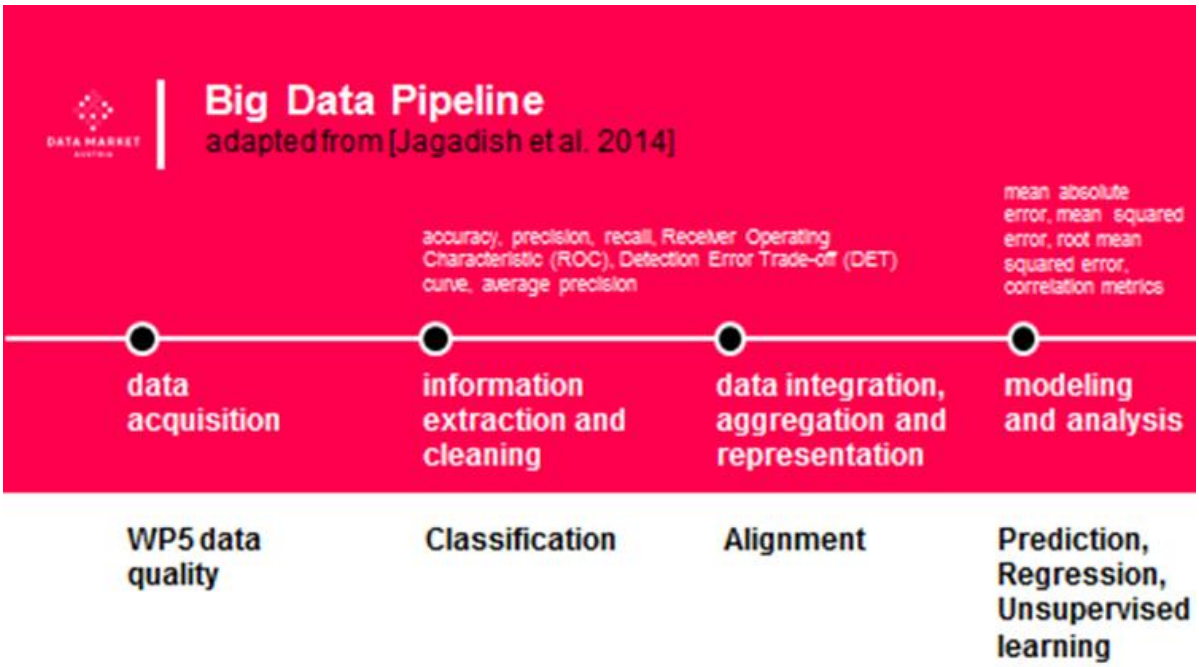

Figure 3: Evaluation metrics in a general big data pipeline, together with typical machine learning methods used in each step.

evaluation metrics used in each step.

Unlike search, for general machine learning there is no historic center of evaluation. Recently, the Data Science Evaluation at NIST [39] has approached this in a similar manner to that of search evaluation: creating a repository of data, tasks, and expected results that can be used to compare between systems, present and future, in a reliable way.

In 2016, the IAD group at NIST has run for the first time a pilot task for data science evaluation, using a Mobility use case, similar in that sense to the pilot used in the Data Market Austria. The task was first to identify errors in a stream of traffic data, and second, to correct these errors.

3 Technology Foundation

3.1 Information Extraction

Many different IE software tools are available. However, most of them have a strong academic background and are not designed to be used in a production environment. In this chapter, some potential candidates for IE software are mentioned, which will be evaluated for their use in DMA. The software presented here is already selected on terms of maturity and compatibility with the Software Stack used for the ScaR Framework.

3.1.1 Stanford CoreNLP[1]

The CoreNLP Toolkit is a Natural Language Processing (NLP) analysis software library. It offers traditional IE support with bootstrapped pattern learning and also OIE tools. Apart of IE, the CoreNLP has extensive capabilities in the field of NLP including Part-Of-Speech (POS) tagging, named entity recognition (NER), coreference resolution system, sentiment analysis, bootstrapped pattern learning, and the open information extraction tools. The Stanford CoreNLP is available in the languages English, Arabic, Chinese, French, German, and Spanish although not all features are available in all languages. The Stanford CoreNLP is implemented in Java and available under the GNU General Public Licence v3 or later. Commercial licencing options are also available for closed source software.

3.1.2 General Architecture for Text Engineering (GATE)[2]

GATE is an extensive NLP processing software that can be included as a library in other software projects or used as standalone software. The feature list of GATE is not limited to IE and NLP but also includes parsers to extract text from different file formats like PDF, HTML, Emails, etc., Although the main focus of GATE lies on English, there are plugins available to process other languages including French, German, Italian, Danish, Chinese, Arabic, Romanian, Hindi, Russian, Welsh and Cebuano. GATE is implemented in Java and available under de the GNU Lesser General Public License. An option for commercial support in different countries is available.

3.1.3 Apache Unstructured Information Management Architecture UIMA3

Unstructured Information Management Architecture (UIMA) was started by IBM and defines a standard architecture for content analysis and is now an OASIS standard. The standard implementation is provided by the Apache UIMA project. UIMA deals with the extraction of information from any kind of unstructured data. The unstructured data is not limited to text but could also be audio or video files. UIMA gained public attention by its use in IBM Watson. UIMA can cooperate with the Apache OpenNLP[4] components to extract information from text. Apache UIMA is based on a Java Framework and can handle components written in Java and C++. It is distributed under the Apache License.

---

[1] https://stanfordnlp.github.io/CoreNLP/

[2] https://gate.ac.uk/

[3] https://uima.apache.org/

[4] https://opennlp.apache.org/

3.1.4 MAchine Learning for LanguagE Toolkit (MALLET)1

MAchine Learning for LanguagE Toolkit (MALLET) is a software library for IE, statistical NLP, document classification, clustering, topic modelling, and other machine learning application to text. MALLET has a strong emphasis on machine learning and numerical optimisation. While the focus of MALLET lies on tools for document classification, it also includes many tools for sequence tagging and IE. MALLET is written in Java and released under the Common Public License.

3.1.5 Calais2

Calais is a product offered by Thomson Reuters to do IE on web pages. It reads the unstructured text and generates a Resource Description Framework (RDF) description of the information found in the text. Calais is provided as a service by Thomson Routers and can be called via an API. Thomson Reuters offer free and paid access to Calais with different levels of support and different amounts of allowed calls per day.

3.2 Scalable Recommender Framework (ScaR)

The Know-Center contributes its framework ScaR [18], which offers highly customizable and scalable recommendation as a service. The framework features state-of-the-art algorithms and novel approaches, which exploit various data sources such as content, social or location data.

ScaR provides real-time recommendations in a multi-domain environment based on the following requirements:

- **Scalability.** The recommendation system needs to be scalable (i.e., have the ability to process a high load of incoming recommendation requests), as significant changes in the request rate can happen at runtime.
- **Online Data Processing.** It should be possible for the recommender system to efficiently handle data streams. Thus, data updates must be immediately taken into account in the next recommendation request (i.e., no expensive recalculations should be necessary).
- **Real-Time Performance.** Recommender approaches need to provide recommendations in real-time, even when utilizing or combining different data features.
- **Multi-Domain Performance.** Performance isolation should be provided for each domain specific recommender.

In order to realize the above-mentioned requirements, ScaR picked up the Software-as-a-Service (SaaS) model, which has had a significant impact on the design choices of software architecture - especially when applied to real-time multi-domain recommender systems. Such architectures are usually characterized by short response times, high scalability, multiple application domains, and individual customization features.

3.2.1 Architecture Overview

A customizable and scalable architecture is the main requirement to support a multi-domain environment. Thus, the core concept of ScaR is its decomposition into several collaborating

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

[2] http://www.opencalais.com/

services called modules. Generally, ScaR adopts the Micro services Architecture design pattern[1] (e.g., like Netflix). This architecture suggests developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms. Basically, every module can be deployed independently from other modules and spin up an embedded Jetty HTTP server at start-up, facilitating configuration and maintenance. Communication between modules is established through a standardized HTTP/REST interface making each module easily exchangeable. The general architecture of ScaR is illustrated in Figure 4 and shows the communication dependencies in a one-domain environment between Apache Solr (see Section 3.2.1.1) and the five core modules of ScaR: (i) Data-Modification-Layer, (ii) Recommender-Engine, (iii) Recommender-Customizer, (iv) Recommender-Evaluator, and (v) Service-Provider.

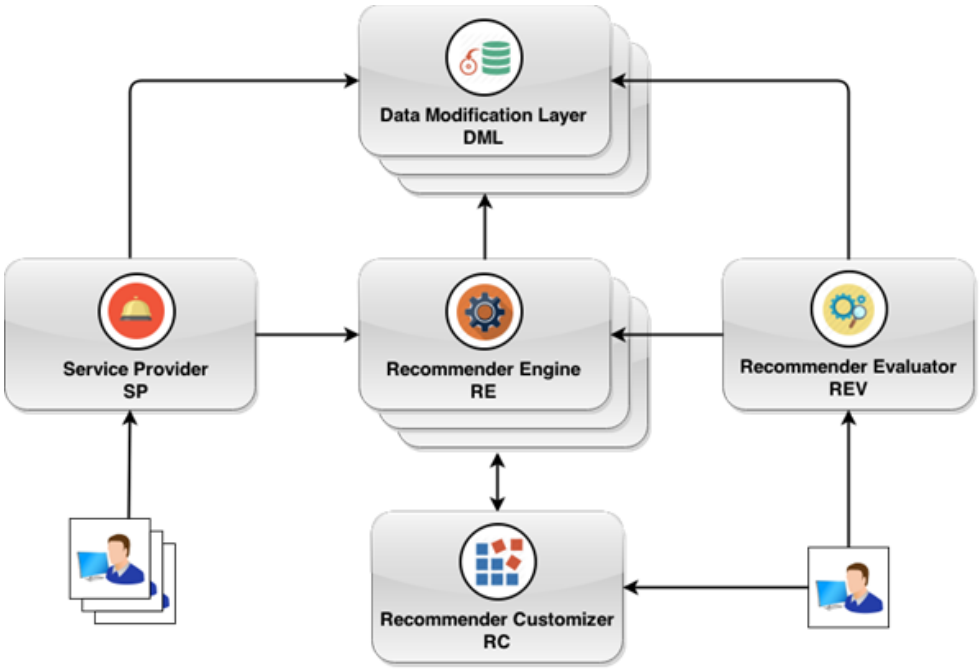

Figure 4: System architecture of ScaR in a one-domain environment showing how the modules work together. Each module is a standalone HTTP server which knows the locations (i.e., URL) of its communicating partners.

3.2.1.1 Data Storage and Access with Apache Solr

With data generation doubling approximately every 40 months [40], nowadays, most systems migrate from traditional databases to distributed systems that can scale more easily and handle incoming massive data streams. The basis for storing data in ScaR is the Apache Solr search engine.

Solr provides horizontal scaling, allowing it to create either shards (i.e., splitting the data into smaller indices to increase the performance of search queries for huge data sets) or replicas (cloning the existing shards to another machine to increase the fault-tolerance of the whole system).

Furthermore, the built-in MoreLikeThis[2] search method uses the vector space model represented by TF-IDF values to find similar items. This functionality is particularly useful for developing recommender approaches which need to process large amounts of data and still provide real-time performance.

[1] http://microservices.io/patterns/microservices.html

[2] https://cwiki.apache.org/confluence/display/solr/MoreLikeThis

3.2.1.2 Data-Modification-Layer (DML) Module

The DML module acts as an agent between the recommender system and the attached external data storage. Currently, Scar supports Apache Solr as data storage backend but the design of the DML module provides flexibility to switch to any other data storage (e.g., a graph database or ElasticSearch) with little effort. Additionally, the system can be easily extended to provide data mixed from different storage systems. For example, one could decide to integrate RecDB [41] for Collaborative Filtering and Apache Solr for Content-Based Filtering approaches.

ScaR is highly customizable and currently, it can handle item (e.g., ratings, purchases, etc.), social (e.g., likes, interests, etc.) and location data (e.g., check-ins, in-door locations tracked through mobile phones, etc.).

Explicit Data Handling

Depending on the available data from the domain in which ScaR is used, it already supports information about user profiles, items, shared locations, social streams and user actions.

- **User Profiles.** A set of user profiles $U = \{u_1, \dots, u_n\}$, where each user entity u_i can have both demographic and social features assigned to it: date of birth, sex, language, address, education, work, favourite locations, groups belonging to, friends, biography, interests. In addition, a user can have a list of assigned devices (e.g., a smartphone MAC address tracked by indoor positioning systems).
- **Items.** A set of items $= \{i_1, \dots, i_n\}$, where each item i_i can be described by the following features: name, description, categories, manufacturer, seller, selling location (either coordinates or a self defined location zone), price, currency and time when the item starts to be valid (e.g., can be purchased).
- **Shared Locations.** A set of location data (e.g., check-ins) $SL = \{sl_1, \dots, sl_n\}$, where shared location entities sl_i are described either by location coordinates or an arbitrary defined location zone.
- **Social Streams.** A set of social stream data (e.g., Facebook wall posts) $SS = \{ss_1, \dots, ss_n\}$, where each social stream entity ss_i has the following features defined to it: the source user, the target user, content (e.g., textual content of a comment), type of action (e.g., like, comment, etc.), data source (e.g., Facebook, Twitter, etc.) and a timestamp. This also works when a user does not directly interact with another user but with a social entity of this user (e.g., when a user comments on another user's comment).
- **User Actions.** A set of actions of a user to an item $UA = \{ua_1, \dots, ua_n\}$. User actions can either be implicit (e.g., purchases, views) or explicit (e.g., ratings). In case of reviews, it is also possible to provide the textual review content in addition to the rating.

These various data types are the baseline for the DMA and will be modified and extended to fit the needs of recommending not only one type of object (dataset vs. service) but also a tuple and multiple combinations.

Inferred Data Handling

As some information is not directly available to the recommender system (e.g., network information such as connected neighbours), ScaR supports inferring network structure information of social and location networks.

- **Social Network.** Information from the social stream usually contains multiple features which could be used for creating a social network. A social network can be created by connecting users who have liked, commented or shared content with each other. The resulting data is stored in form of a network where users are nodes and edges between users indicate a social connection.

- **Location Network.** The user's location data is usually represented as raw location data (i.e., the coordinates of the visited locations). To model a neighbourhood between two users, we construct a network where two users have ties between each other if they visited the same location within the same day and hour.

In terms of DMA this will be interesting if users are able to connect their social network profiles (LinkedIn, Xing, etc.) to find more suitable recommendations. Also other contextual information may be inferred to create a better understanding of the users needs.

3.2.1.3 Recommender-Engine (RE) Module

In live settings, besides being accurate, recommendations need to be provided fast since most users are not willing to wait, especially if the recommendation was not explicitly requested. In order to support this, the Recommender-Engine module currently contains four types of recommender algorithms that make use of the efficient query language of Apache Solr, namely: User-Based Collaborative Filtering (CF), Content-Based Filtering (CBF), Most Popular (MP), and Hybrid approaches. We currently support the following prediction tasks:

- **Item recommendation.** For example, items could be products in a web shop, hotels in a booking portal or venues in a location-based search and discovery service like Foursquare. In case of DMA it would fit a dataset but also a service. Since we are going to recommend combinations of both the search space is extended and the task to find the right combination is harder.
- **Category recommendation.** Items can have different categories assigned to them, like electronic products or levels-of-service when booking a hotel. In relation to DMA each dataset and service will be associated with one or more categories. These can be used to guide the user to find the right direction.
- **User recommendation.** Such prediction task is usually associated when peers or experts in social networks need to be found. Again this can be related to DMA were certain users (broker) are considered experts and together with their interactions they can be associated with a category or group of items.
- **Location recommendation.** Specific locations, also called points-of-interest, that someone may find useful or interesting (e.g., next city to visit).

User-Based Collaborative Filtering

Collaborative Filtering is usually accomplished in two steps: first, find the k-nearest neighbours based on some similarity metric (e.g., Cosine or Jaccard similarity), and afterwards recommend items of these neighbours that the target user still has not consumed [14]. Based on related studies (e.g., [42], [43]), the implemented Collaborative Filtering approaches calculate similarities between users in two different ways: either calculating similarities between users on the content directly provided by the users (e.g., item ratings) or on the network structure inferred from the interacting user profiles (e.g., social or location-based networks).

Content-Based Filtering

Content-Based Filtering approaches analyse item meta-data to identify other items that could be of interest for a specific user. This can be done based on various types of metadata tied to items or users [13]. ScaR provides an internal API for building queries which will use Apache Solr's Vector Space Model to calculate similar documents (representing, for example, items or users) using TF-IDF.

Most Popular

A simple Most Popular approach is also available in ScaR. It recommends the most popular items based on the available explicit (e.g., ratings) or implicit (e.g., purchases or views) item interaction data. Also to be noted, usually a Most Popular approach is unpersonalized and thus, every user would receive the same recommendations. ScaR's default behaviour is to

encourage novel items for the user and does it by filtering items that are already known (e.g., have been rated) to the user.

Hybridization strategies

ScaR also provides the ability to create hybrid combinations of the recommender approaches mentioned above. For this reason, we implemented three well-known hybridization strategies, namely Mixed Hybrid, Weighted Sum and Cross-source (see [44] for more details), that can be used to create and evaluate various kinds of recommender combinations. Furthermore, the latter two (i.e., Weighted Sum and Cross-source) allow for setting the weight of the components dynamically at runtime.

3.2.1.4 Recommender-Customizer (RC) Module

The Recommender-Customizer module allows it to configure the recommender approaches that are implemented in ScaR. It provides so-called recommender profiles, whose sole purpose is to customize a single recommender approach and to provide a reference to it.

A profile (as seen in Listing 1) is defined by a unique reference-id, the name of the algorithm being configured (i.e., the class names from the Recommender-Engine module), and some additional parameters and filter criteria. For instance, a Collaborative Filtering approach based on location network data can be configured to use different kinds of similarity metrics (as shown in [43]) and use either explicit (e.g., ratings) or implicit data (e.g., purchases or item views) for recommending items of similar users. It is also possible to go a step further and define post filtering for the recommended items (e.g., the price range of the items, suitable for minors, only novel items, within a location, etc.), which allows to customize and reuse the same recommender algorithm for different use cases. Please note that these post-filtering criteria can also be passed to the recommender service at runtime.

Whenever a recommender profile gets updated (e.g., when parameters are edited) or a new profile is created, the changes will be propagated throughout the entire system and every deployed RE module will be informed about the changes and the altered profile will be used as soon as a new request comes in.

```
id: basicCF
name: "basic collaborative filtering"
algorithm: basicCF
parameters:
    # how to calculate user similarity?
    similarityFunction: JACCARD_ENTITIES
    # what to recommend?
    recommendationType: ITEMS
    # what data to use for item ranking?
    useRatings: true
    usePurchases: true
    useViews: false
filterCriteria:
    # recommend items suitable by brokers?
    checkBrokerRatings: false
    # recommend items which belong to the "heatmap" category
    category: heatmap
    languageOfRecommendedItems: ENG
    # filter by price range
    maxPrice: 5000
    minPrice: null
    currency: EUR
    # filter items already interacted with?
    novelty: true
```

Listing 1: Example of a recommender profile which performs a User-Based Collaborative Filtering approach using the user's location network data (i.e., a network with connections between users if they were at the same place at the same time).

3.2.1.5 Recommender-Evaluator (REV) Module

Before a recommender system is deployed for live usage, offline evaluations of the available approaches and their parameters are carried out. Using the available data, an offline evaluation aims to simulate user behaviour by splitting the data into training and test sets (see, e.g. [45]). That is, for each user, a given number of items is removed from the training set and added to the test set to be predicted. The similarity between the recommendation list and the real data in the test set is then used afterwards to determine the success of the prediction based on standard evaluation metrics.

Online evaluations are another evidence regarding the performance of recommender systems. Letting real users use the system and perform real tasks is by far the most trustworthy indicator how successful recommendations are. As ScaR logs all generated recommendations by default, it is possible to track if a user makes use of a recommended item (e.g., the user clicks on it, likes it, etc.). The REV module in return offers services to get statistics of the recommendation feedback in form of the number of recommended items that have actually been clicked (i.e., click rate), purchased (i.e., purchase rate), or liked (i.e., like rate) by the user.

Most recommender systems are evaluated based on accuracy estimates. This is indeed crucial but not the only deciding factor to determine the success of a deployed recommender system. Users may not only want to get their exact preferences to be predicted but rather have different context-bound expectations (e.g., find novel or explore diverse items). To be able to identify different properties that may influence the user's expectations, the REV module also provides the possibility to conduct either A/B or multivariate tests.

An A/B test is a form of statistical hypothesis testing where two variants of an approach (hence, A and B) are compared. Those approaches are usually identical except for one property that might affect a user's behaviour [46]. Similarly, multivariate testing evaluates several properties between different approaches. More precisely, to define a test, ScaR accepts a time slot, a list of recommender profiles that should be compared, as well as the test set size relative to the entire user set (e.g., 30% of the users should be evaluated). Depending on the number of profile ids and the testing set size, randomly assigned users are split into different evaluation sets (i.e., one evaluation set for one profile id). Moreover, it is possible to define a test as adaptive, meaning that users which are not part of any evaluation set (e.g., newly registered users) will be randomly assigned to one the next time they request recommendations. After an A/B or multivariate test is defined, the Service-Provider module checks if the defined time slot is currently valid and if it is, intercepts the incoming recommendation requests and modifies them to perform the test.

3.2.1.6 Service-Provider (SP) Module

To make ScaR easier to access, we developed the Service-Provider (SP) module. The SP acts as a proxy for incoming HTTP requests, providing an API (as seen in Figure 5) to upload streaming data, request recommendations, or even to search the stored data. If ScaR is deployed in a distributed manner, it acts as a load balancer and reroutes incoming requests to the currently available processing modules, thus increasing the query throughput. Note that although each service has its own REST API, only special clients, which we refer to as client admins, have direct access to the modules in ScaR. All other clients need to use the SP module as seen in Figure 5.

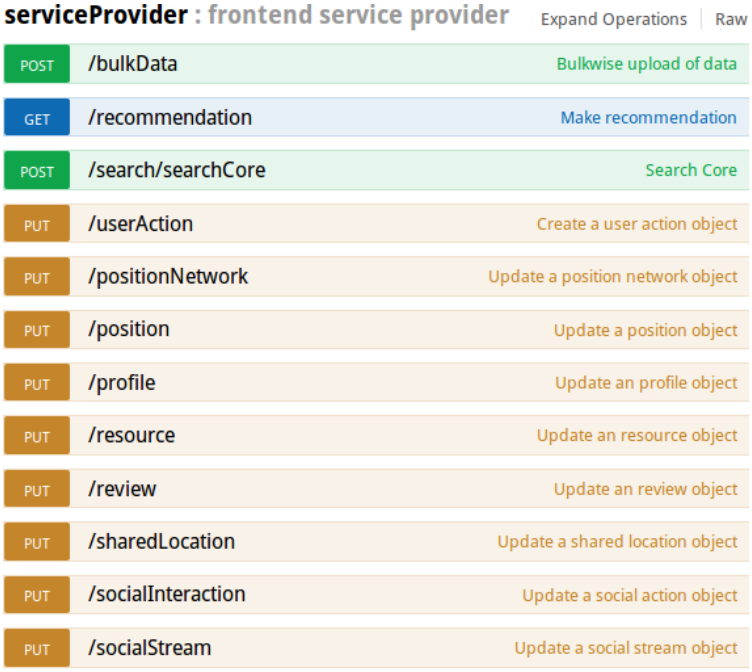

Figure 5: Example for the interactive REST API of the Service-Provider module. Note that every other deployed module has a similar auto-generated interactive API documentation.

As already mentioned in Section 3.2.1.5, when recommendations are requested, the SP module also checks if an A/B or multivariate test is defined. If a test is initiated for the current time slot and the target user, the provided recommendation profile (containing the recommendation approach) will be replaced by the profile the target user was randomly assigned to. In case the test was defined as adaptive and the user is not contained in any of the created test sets (i.e., was not assigned at all or is new to the system), the SP module

will dynamically assign the user to the smallest currently available test set and, respectively, use the profile assigned to it.

3.2.2 Deployment

This section describes how ScaR can be deployed, either on a local machine or distributed on multiple machines.

3.2.2.1 Local Deployment

Each of the modules mentioned in the previous section is a standalone server, which can be deployed locally on any desired port. To deploy each module, simply the path to the environment configuration file (e.g., config.yml) needs to be specified. In case of a local deployment, the configuration file only needs to contain the ports of the depended modules running on the same local machine.

3.2.2.2 Distributed Deployment

Each of ScaR's modules can be deployed and started multiple times, either on the same or on different machines, thus supporting horizontal scaling. To keep track of all deployed modules, we make use of Apache ZooKeeper[1]. All registered services are coordinated by ZooKeeper and they can be divided into hierarchical namespaces to deploy several different recommender domains on one system. Hence, a module's configuration file does not need to contain the exact location of the depended modules, just the location of the ZooKeeper cluster.

Fault Tolerance

Deploying a larger number of modules in a distributed manner increases the probability that a failure will happen (e.g., unexpected hardware shutdown, I/O problems, software bugs, etc.). To detect and react to such problems, each deployed module provides a health-check service to verify its status. As ScaR is built on a micro-service architecture, it does not need to cope with central node failures as it is the case within a master-slave architecture. In case a module fails, Zookeeper removes the faulty module from its list of live nodes and no further requests will be redirected to this particular module. Thus, the module will not necessarily cause any major problems as long as there exists another module of the same type available. When experiencing a high request load, it is possible to deploy and register an additional module to ZooKeeper cluster on-the-fly. To further improve the reliability of the system, we run multiple ZooKeeper instances in a cluster mode to overcome the outage of single instances.

3.3 Evaluation components

3.3.1 Search evaluation components

The most widely-used evaluation software for information retrieval benchmarking is the trec-eval, a C-implemented program to take in ranking results and query relevance judgements and output a range of over 30 evaluation metrics. The code is publicly available at GitHub[2] and a docker image for it has already been created in the context of DMA[3].

[1] http://zookeeper.apache.org/

[2] https://github.com/usnistgov/trec_eval

[3] https://hub.docker.com/r/mihailupu/trec_eval/

Trec-eval hast the advantage of being extremely easy to use and completely open, but it does require specific input formats. In particular, the ranking results of a system S1 must be in the form:

```
topic Q0 document-id rank score system-name
```

where

- `topic` is an identifier for an information need;
- `document-id` is an identifier for the document
- `rank` is an integer indicating the rank of the object in the sorted list (normally, this should be ascending from the first line to the last line)
- `score` the similarity score calculated (normally, this should be descending from the first line to the last line)
- `system-name` the name of the system tested

Here is a short example of a potential output for a topic:

- `topic1 Q0 misc.forsale\76442 1 2.813525 s1`
- `topic1 Q0 misc.forsale\76056 2 1.0114759 s1`
- `topic1 Q0 rec.autos\102958 3 0.58848727 s1`

3.3.2 Machine learning evaluation components

Every machine learning library will have implemented the evaluation metrics for clustering, classification, and regression, because in most cases the same methods are used as error functions in the process of learning itself.

For this particular report, we can focus on what is now the most widely-used language for data science, Python, and in particular on its scientific toolkit, the scikit-learn[1].

3.3.2.1 Classification

We will be looking at examples for five evaluation metrics for classification:

1. Classification Accuracy.
2. Logarithmic Loss.
3. Area Under ROC Curve.
4. Confusion Matrix.
5. Classification Report.

For each of them, we assume that there is some data loaded and a classification method has been applied. For instance, the data import for a logistic regression is shown in Listing 2. The code to evaluate a classification is given in Listing 3, in order to obtain a result that indicates how often the correct class has been assigned. While commonly being used, this approach is often misapplied under the assumption that all classes are equally populated and of equal importance.

[1] http://scikit-learn.org/stable/

```
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/pima-indians-diabetes/pima-indians-diabetes.data"

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi',
'age', 'class']

dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = LogisticRegression()
```

Listing 2: Data import for logistic regression.

```
scoring = 'accuracy'
results = model_selection.cross_val_score(model, X, Y, cv=kfold,
scoring=scoring)
print("Accuracy: %.3f (%.3f)") % (results.mean(), results.std())
```

Listing 3: Evaluate a classification.

Logarithmic loss (loglos) is particularly applied when the method to classify predicts probabilities of an object belonging to a class as shown in Listing 4.

```
scoring = 'neg_log_loss'
results = model_selection.cross_val_score(model, X, Y, cv=kfold,
scoring=scoring)
print("Logloss: %.3f (%.3f)") % (results.mean(), results.std())
```

Listing 4: Evaluation of logarithmic loss.

Similarly, Area under the ROC Curve, a method applied to estimate the ability of the system to distinguish between positive and negative classes, is applied just by changing the ‚scoring' function in Listing 4 to the configuration shown in Listing 5

```
scoring = 'roc_auc'
results = model_selection.cross_val_score(model, X, Y, cv=kfold,
scoring=scoring)
print("AUC: %.3f (%.3f)") % (results.mean(), results.std())
```

Listing 5: Evaluation of ROC Curve area.

The two metrics in Listing 6 provide a more detailed view. The confusion matrix shows how often objects were classified in their ‚right' or ‚wrong' class (true/false positives and

negatives), while the classification report shows both Precision and Recall, in order for the user to understand whether the averages obtained are meaningful or not[1].

```
X_train, X_test, Y_train, Y_test =
model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
model = LogisticRegression()
model.fit(X_train, Y_train)
predicted = model.predict(X_test)
matrix = confusion_matrix(Y_test, predicted)
print(matrix)
report = classification_report(Y_test, predicted)
print(report)
```

Listing 6: Evaluation of Confusion matrix.

3.3.2.2 Regression

For regression, typical error metrics are Mean Absolute Error (MAE), Mean Squared Error (MSE) and R^2.

3.3.3 Dockerisation

We currently are running a Docker-based evaluation infrastructure for information retrieval tasks, where everything, from the search engine variants, to the visualisation components are Docker containers against a common storage. Figure 6 shows this existing infrastructure.

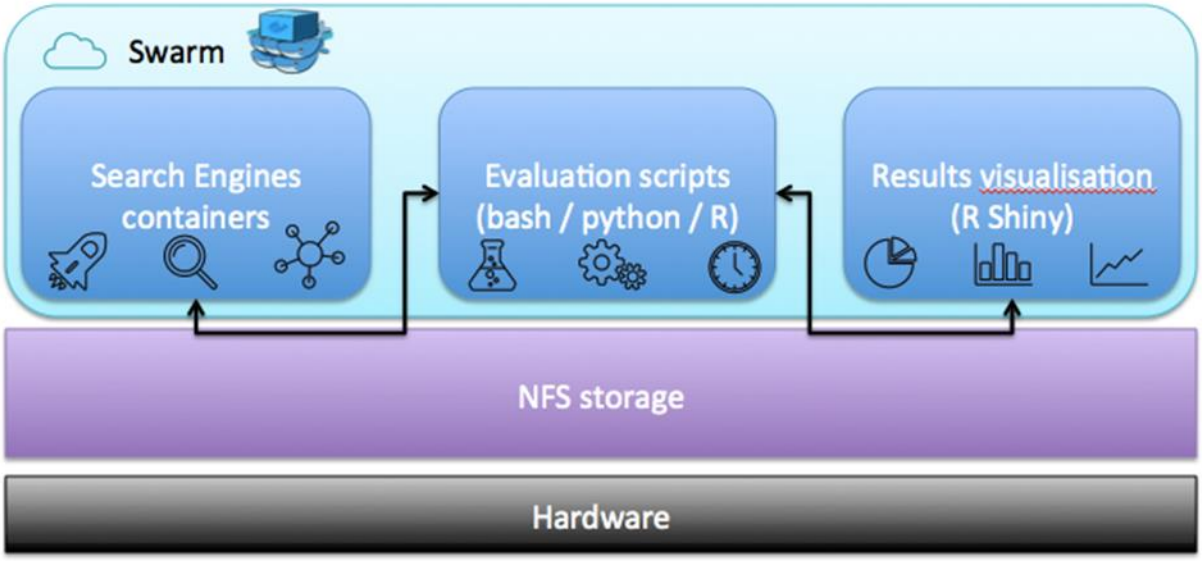

Figure 6: Existing Docker infrastructure for evaluation.

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

4 Proposed System Architecture

The proposed evaluation infrastructure, in line with the general DMA architecture, is completely Docker-based and follows the technology and metadata decisions taken in work packages 4, 5, and 6.

The central architecture (illustrated in Figure 7) is focused around the match-making framework, which will be responsible for data ingestion and information enrichment. The enriched information will further be employed by the search and recommender engine. The internal service communication is using RESTful API's as described in Section 3.2.1.

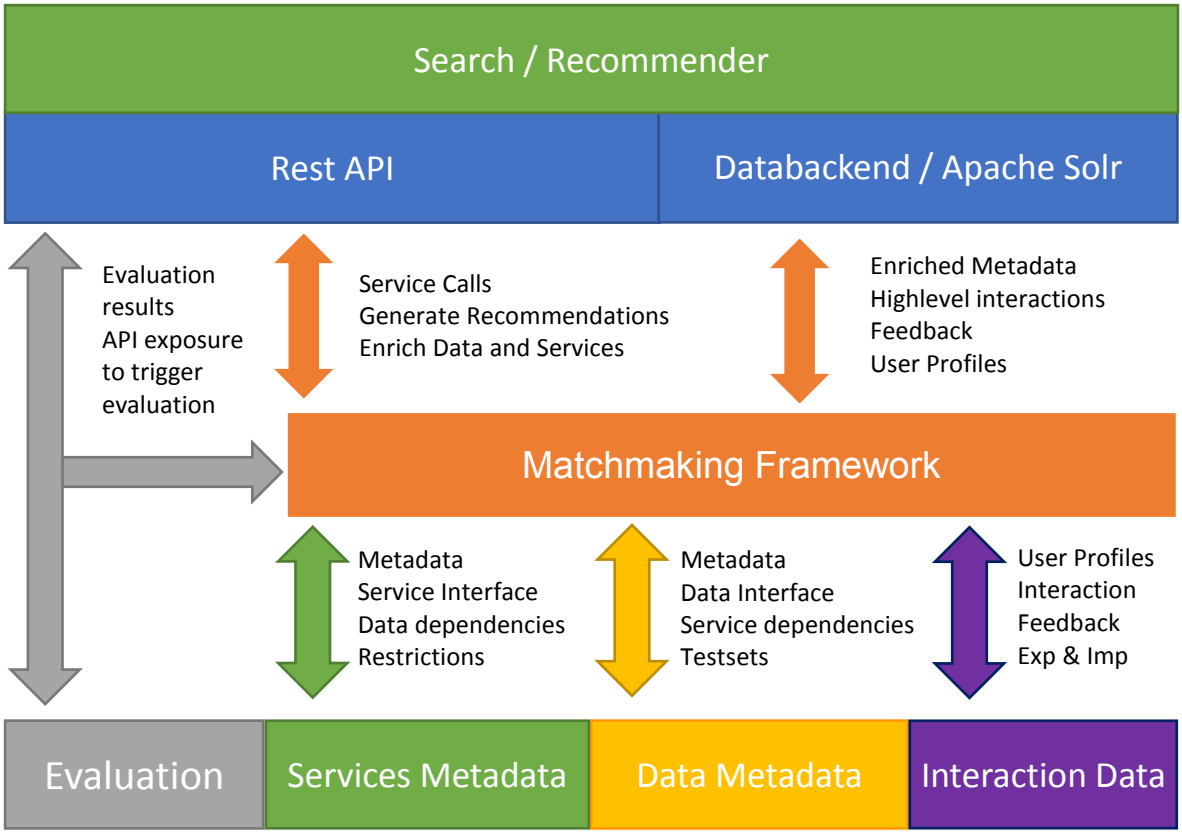

Figure 7: General system overview showing the various interactions between WP7 services and the central DMA services.

All the different parts of data ingestion and output services are described in the following section in more detail. Prior, the currently defined metadata schema is elaborated.

4.1 Metadata Schema

In order to create meaningful recommendations, metadata profiles need to be extracted for services, datasets. Since there are no gold standards defining all the different components of the data market, the technical work packages created metadata schemas based on DCAT-AP. The metadata schemas are described in detail in the corresponding deliverables of WP5 (Datasets) and WP6 (Services). WP7 is focusing on the mapping and the interconnections, as well as the interactions gathered from the DMA usage and the DMA user. Each dataset and service in the DMA platform will have a corresponding metadata description. The description consists of tuples of attributes and values. The following schemas might change throughout the development of the first prototype due to incoming dependencies of the other central services.

4.1.1 Datasets

Each dataset in the DMA platform will have a corresponding metadata description. The core metadata schema for the datasets consists out of 18 fields and 41 in total. A dataset is a collection of data, published or curated by a single source, and available for access or download in one or more formats. Each dataset is associated with a Data Catalogue as shown in Figure 8, which is a collection that hosts and provides some information of the datasets being described. Further a data catalogue is associated with a Data Portal, which is a Web-based system that contains at least one data catalogue with descriptions of datasets and provides services, enabling discovery and re-use of the datasets. In a first step, the DMA will be the only data portal in the system, yet the portal's structure is based on DCAT-AP and therefore provides a high level of connectivity regarding future other system.

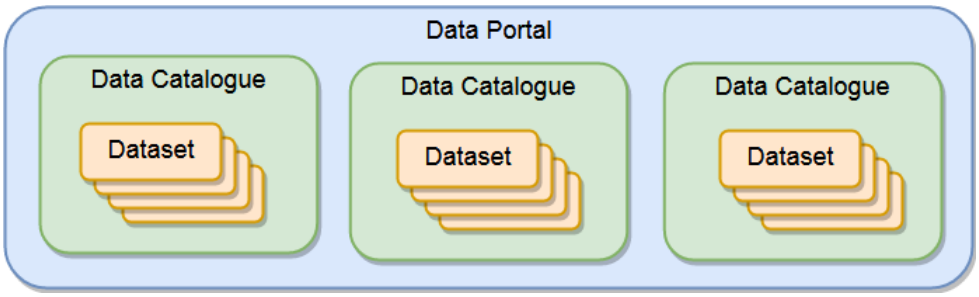

Figure 8: Data Portal, Data Catalogue, and Dataset structure.

4.1.2 Services

Each service in the DMA platform will have a corresponding metadata description. The core metadata schema for the services consists out of 16 fields and 33 in general. The attributes are divided into 5 topical categories:

- General Service Properties (20 items)
- Technical Properties (5 items)
- Performance Properties (5 items)
- Security Properties (2 items)
- Rating Properties (1 item)

Compared to datasets, single services are not nested in any form but can store certain service dependencies, e.g., pre-processing services, and dataset dependencies, e.g., language thesaurus, or baselines. Continuously developed services are labelled with version information (version number, description, last update).

4.1.3 User & Interactions

Each user in the DMA platform will have a corresponding metadata description. The core metadata schema for a user is still in process and will be further developed in collaboration with WP4 and the creation of the platform itself. These elaborated attributes are necessary in terms of recommendation and match-making. To gain further insights about the users within the DMA platform, their interactions are also tracked and stored. With this kind of information, the future behaviour can be modelled and, in turn, meaningful recommendations for similar users can be derived. Every user is associated with an organisation. If a user is providing a dataset or a service to the DMA, these have to be associated with at least a "dummy" organisation of the same name as the user that submitted the dataset or registered the service. Further, a user can also be a consultant which will be main contact point for certain domain areas. The will be recommended if datasets and services are needed from a particular domain. These users will than operate as a physical broker to find suitable combinations and even establish connections to third party participants outside of the data

market. The current list of metadata properties for users and organisation are shown in Table 1 and Table 2.

Identifier	Definition / Description
UID	Unique identifier of the DMA user
Name (Title)	The name identifier of the DMA user
Verified	Checks if a user's identity is verified
Domains	Main operating domains of the user (in case of broker)
Interaction UUIDs	List of the past interaction UUIDs
Dataset UUIDS	List of Dataset UUIDs used by this user
Service UUIDs	List of Service UUIDs used by this user

Table 1: General Metadata properties of a user.

Identifier	Definition / Description
UID	Unique identifier of the organisation
Name	The name identifier of the organisation
Description	A description of the DMA user. (a user might represent a company or department)
Verified	Checks if organisation identity is verified
Interaction UUIDs	List of the past interaction UUIDs
Dataset UUIDS	List of Dataset UUIDs owned by the organisation
Service UUIDs	List of Service UUIDs owned by the organisation

Table 2: General Metadata properties of an organisation.

Interactions are a crucial part of the recommendation process - without them, it would not be possible to identify the requirements of users. The basic metadata of an interaction is defined in Table 3. The various types of an interaction are further defined during the development of the DMA portal. The following types represent a basic set of interactions to be tracked:

- Fired Search Request
- Resource Viewed
- Resource Created
- Resource Rated

where a resource can be a dataset, service, or a user of the DMA platform. In addition to those basic types, the combined interactions are of particular interest. For example, the combination of a search request and the resources viewed associated to the request. These combinations can be used for matchmaking and the evaluation of the recommendations.

Identifier	Definition / Description

UID	Unique identifier of the DMA interaction
Type	The type of the DMA interaction
Owner	UID of the DMA user which created the interaction
Created at	Date and Time of the interaction
Version	Version of the DMA Portal
Service UUIDs	Services which are related to the interaction
Dataset UUIDs	Datasets which are related to the interaction
Data Catalogue UUIDs	Data Catalogues which are related to the interaction
Data Portal UUIDs	Data Portal which are related to the interaction
User UUIDs	Users which are related to the interaction
Organisation UUIDs	Organisations which are related to the interaction

Table 3: General Metadata properties of an interaction within the DMA platform.

4.1.4 Interaction and Connection Overview

The unity of all before-mentioned metadata profiles enables the match-making framework to establish connections between various elements and to construct rules and baselines for combinations to recommend. The connection and interaction overview in its general form is illustrated in Figure 9. The basic connections between services and datasets are either created by the user who is interacting with them (viewing, purchasing, rating, etc.) or through predefined dependencies within the services. As described in the previous section, user can be part of an organisation if they are contributing to DMA.

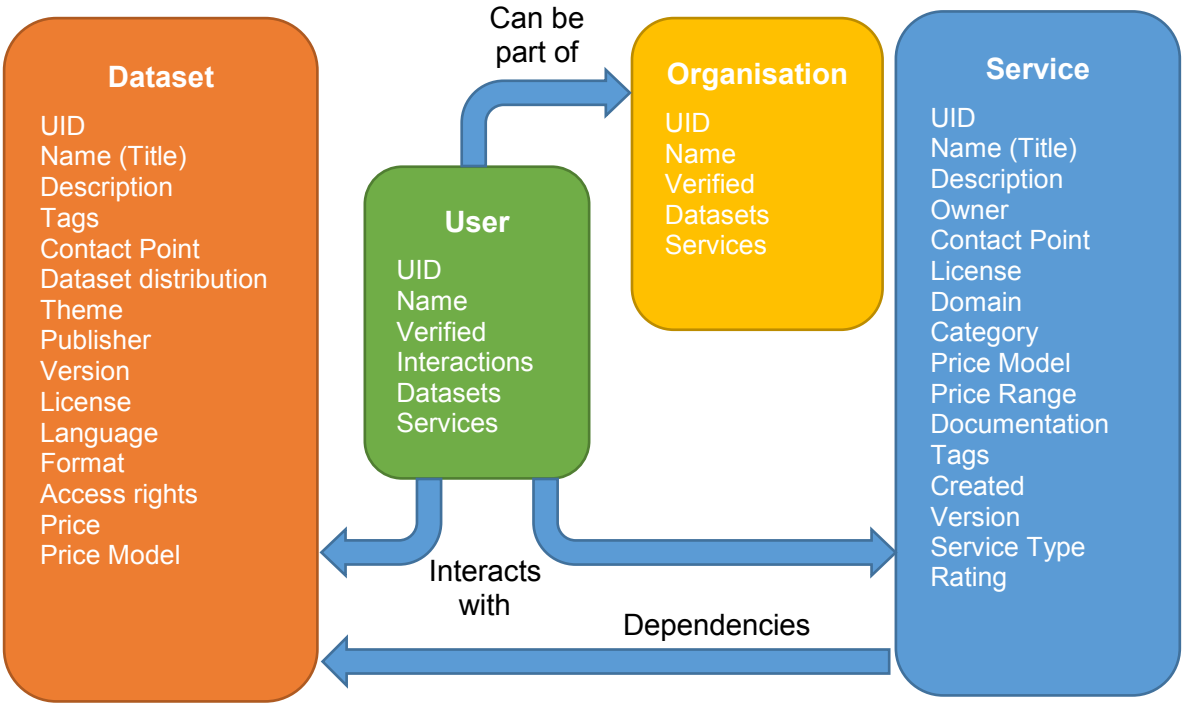

Figure 9: Meta data schema interaction and connection overview.

4.2 Services

This section provides an overview about the 4 main services, which are going to be developed in WP7. The first part explains the initial service for data input from data ingestion and information extraction. The second part will point out the functionalities of the metadata search functionality provided for DMA users and admins. The third part will elaborate on the recommendations and the final part is about the service evaluations.

In order to support multi-tenancy, which in this case are the different stakeholders of the platform as well as multiple instances of DMA deployed on one system, each module will be configured to be part of an isolated environment. We also make use of recent Software-as-a-Service virtualization technologies. Recently, virtualization technologies (i.e., containers), such as Docker[1] or LXC[2] have gained popularity as an alternative to using virtualization regarding automating the process of deployment. These lightweight resource containers provide several promising features, such as portability, more efficient scheduling, and resource management, as well as less virtualization overhead. Moreover, multi-tenancy is also supported by creating isolated environments, if multiple instances of the same application are deployed.

To be specific, key elements of ScaR will be redesigned to use Docker for isolating and deploying its modules in a multi-tenant environment. An example structure of this environment is depicted in Figure 10.

[1] http://www.docker.com

[2] https://linuxcontainers.org

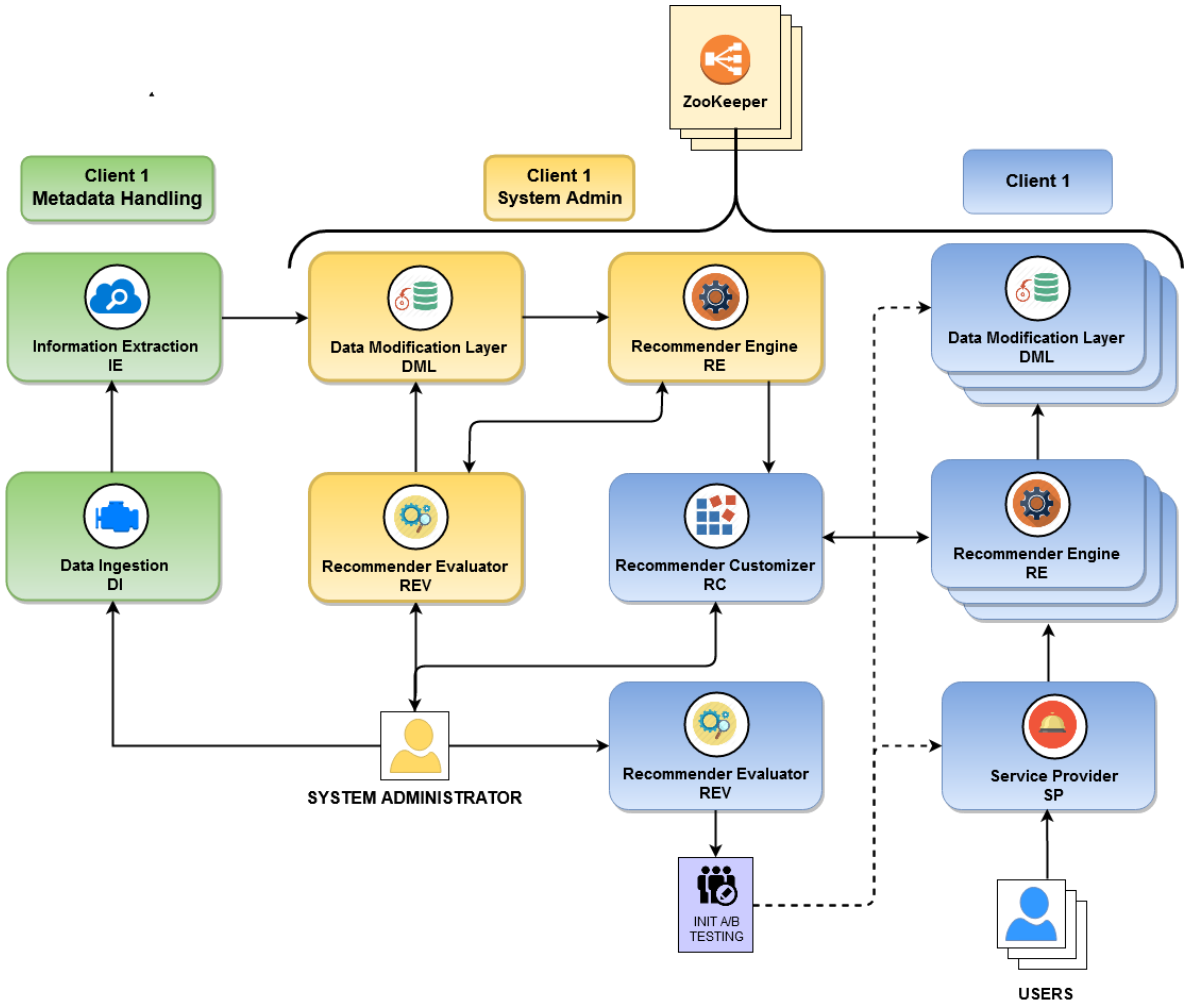

Figure 10: Architecture of Scar system for DMA platform.

4.2.1 Data Input

The data input service is divided into separate modules, one for each data source. Hence, there are four modules: one for dataset metadata, one for service metadata, one for user profiles, and one for interaction data. Each module encapsulates the API of the respective data source to minimise the required changes if an API gets changes or extended. The preferred information exchanges model for each module is a so-called “push-oriented” approach. This means that each source actively pushes new data or updates to the source connection modules, minimising the latency between the occurrence of a data change and the effect of the data change on the recommendation and search service.

The next step in the data input service is the IE, which itself is divided into a pre-processing and a main stage. The pre-processing step is executed for each data source individually. It covers merely a unification of the vocabulary in the input data and generates a common data representation. In the main stage, the actual IE is carried out. The IE transforms the unstructured texts from the input metadata into structured information interpretable by the recommender. Furthermore, a cross-referencing and validation of the input data between all input sources is done.

The final step in the data input service is the hand over from to the DML where the processed input data gets injected into the recommendation and search service.

4.2.2 Search

The search service operates on the service and dataset metadata, i.e., only on content information, while sensitive user information about their transactions is not part of the search. The search enables users to execute queries against the metadata catalogue to find services and datasets that they are looking for. This search service offers an API to integrate it into the DMA portal. Users can specify queries to get search results. The search results can than further be filtered by so called search facets. Facets are filters to narrow the search results and help the users to find the right results.

4.2.3 Recommender

The recommender service is the core of all WP7 services. It automatically generates suggestions of possible datasets and service combinations on the DMA portal. In contrast to the search, it uses also user profiles and the interactions made on the platform to generate the suggestions. The recommendations are automatically generated, meaning that they are not triggered by the users explicitly but implicitly when they navigate to a page of the portal. Hence, the recommender offers an API to provide the suggestions for any given user in the context of the current page in question.

4.2.4 Evaluation

Figure 11 depicts an updated version of the existing technology framework, in the context of the DMA architecture.

At a basic layer, we transition from Swarm to Kubernetes on the Docker orchestration side, and from a NFS to CEPH. This transition implies some effort in adopting a new set of tools, but does not fundamentally change the existing Docker containers. As far as the scripts and code inside those containers, the storage change should be transparent.

In terms of Docker components, we expose the existing and future evaluation scripts as DMA Services, using the description metadata presented in Deliverable 6.1. The Evaluation scripts interact with the services to be evaluated via the standard API and make use of Test Data represented as DMA Datasets (see Deliverable 5.1). At the same time, the evaluation scripts generate data themselves, and using the Data Ingest Pipeline (see Deliverable 5.1) store it with the appropriate identifiers and metadata in DMA.

Results visualisation services can then reuse this data to expose it to the end-user, as any other data visualisation service on the platform would.

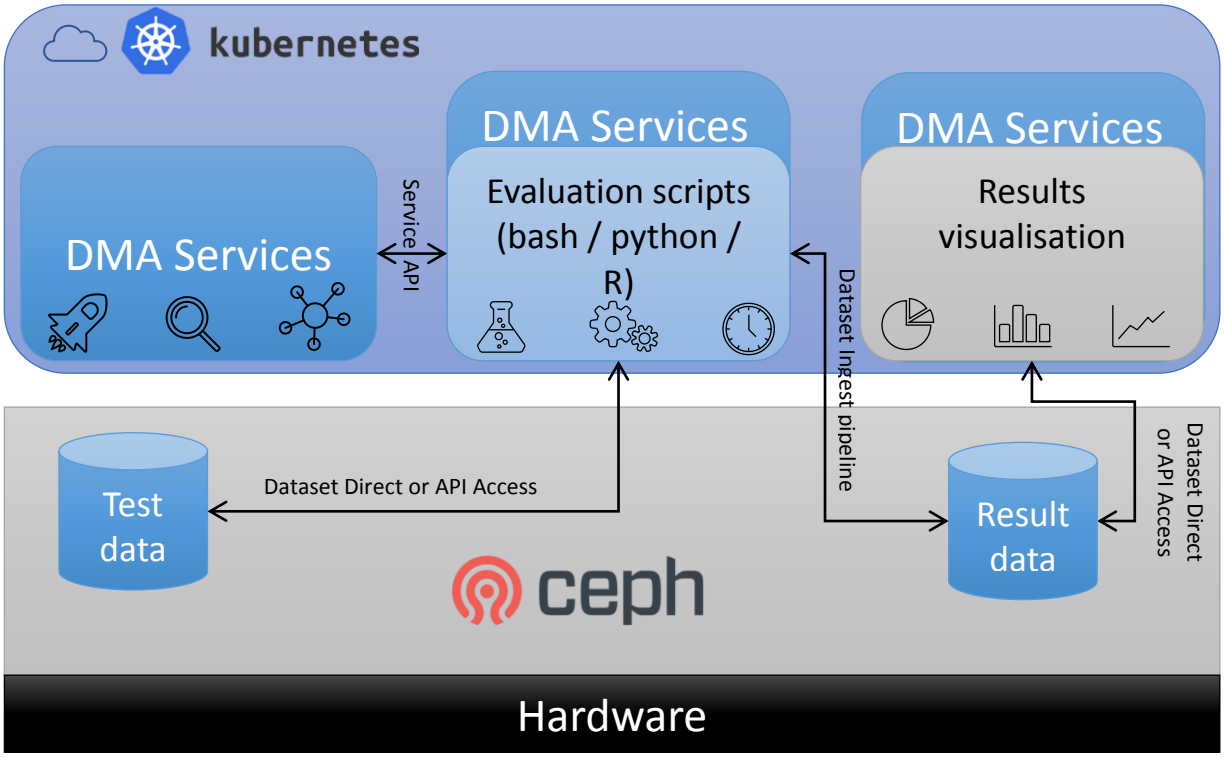

Figure 11: Proposed architecture of the evaluation platform

5 Development Plan

The development of WP7's components is divided into two main phases. The first phase titled "prototype release" is carried out in the first 18 months of the project. The results of phase 1 will be summarised in the second deliverable "D7.2: Initial Release of Foundational Broker and Assessment Technology Prototypes". The second phase builds on the results form phase one and completes the work of WP7. The results are summarised are in the third deliverable "D7.3: Final Release of Foundational Broker and Assessment Technology Prototypes".

5.1 Phase 1: Prototype Release

The first phase will focus on establishing the communication with other central services of DMA and the creation of basic services supporting metadata search, basic data ingest, evaluation as well as basic recommendations. It will include required interface specification for dataset metadata profiles, service metadata profiles, user metadata profiles, interaction logs, and evaluations.

5.1.1 Relation to other Work Packages

5.1.1.1 Task 7.1

a) **Requirements (WP2):** The information extraction has to focus on metadata aspects regarded as important discriminative features in the community driven requirements.
b) **Dataset Metadata Specification (WP5):** The format, technical interface, information flow, and update mode for dataset metadata have to be agreed upon.
c) **Service Metadata Specification (WP6):** The format, technical interface, information flow, and update mode for service metadata have to be agreed upon.
d) **Interaction Data Specification (WP4):** The format, technical interface, information flow, and update mode of user interaction with datasets and services have to be agreed upon.
e) **User Profile Specification (WP4):** The format, technical interface, information flow, and update mode of user profiles to be agreed upon.

5.1.1.2 Task 7.2

f) **Requirements (WP2):** The recommendation and search functionality have to be aligned with the community driven requirements.
g) **Interface for Recommendation (WP4):** Definition of format and technical interface to access the recommendation functionally by the DMA portal.
h) **Interface for Search (WP4):** Definition of format and technical interface to access the search functionally by the DMA portal.

5.1.1.3 Task 7.3

a) **Dataset Metadata Specification (WP5):** The format, technical interface, information flow, and update mode for dataset metadata have to be agreed upon and include standards for specifying test collections.
b) **Service Metadata Specification (WP6):** The format, technical interface, information flow, and update mode for service metadata have to be agreed upon, and include possible evaluation metrics.
c) **Requirements (WP2):** The higher-level evaluation components have to match requirements from the community with respect to the services to be evaluated.

5.1.2 Results

At the end of phase one a baseline for the WP7 functionality will be established. This includes the communication with other DMA modules and a preliminary search and recommendation service. The end of phase one serves as a synchronisation point with other work packages.

5.1.2.1 Task 7.1

a) **Dataset Metadata interface:** Defined and implemented
b) **Service Metadata interface:** Defined and implemented
c) **Interaction Data interface:** Defined and implemented
d) **User Profile Interface:** Defined and implemented
e) **ScaR DML Interface:** All collected data can be transformed so that it is publishable on the DML interface of the ScaR framework.
f) **Basic IE Features:** Unstructured textual data part of Service or Dataset metadata can be processed to
g) **Deployment as DMA Central Service:** The developed components can be deployed as Docker Container integrated in the central DMA software infrastructure.

5.1.2.2 Task 7.2

h) **Portation of the ScaR Framwork to DMA Architecture:** The existing ScaR framework and its components is ported to the Docker based DMA architecture.
i) **Basic Search Functionality:** A basic metadata search functionality is provided to the DMA portal.
j) **Basic Recommendation Functionality:** A separated CB and CF recommendation are provided to the DMA portal.

5.1.2.3 Task 7.3

i) **Basic Services Evaluation Components:** Evaluation components (services in themselves) for Ranking, Clustering, and Classification services will be provided.
j) **Metadata Vocabulary:** On top of existing metadata for data and services, a set of vocabulary items will be created to indicate the usability of a particular dataset as a test collection and as an output of an evaluation task.

5.2 Phase 2: Final Prototype Release

While the first phase focused on establishing communication with other DMA central services and only provides basic functionality, the second phase will focus on extending and incorporating a growing list of features into the developed components. This will include more elaborate IE schemes, a hybrid and adaptive recommender, and a higher degree of automation in the service assessment.

5.2.1 Relation to other Work Packages

The interfaces to the components developed and maintained by other work packages are already an objective of phase 1. In the second phase these interfaces will mostly be unchanged; exceptions are only allowed to correct errors or to cater for new and extended functionality. Each task in WP7 will manage changes in the interfaces directly with the effected work packages.

5.2.2 Planned Extensions over Phase 1

5.2.2.1 Task 7.1

The work on this task will focus on more elaborate IE schemes. First and foremost, better information a move to more elaborated OIE methods is planned to increase the quality of information extracted from unstructured text. Additionally, the coupling of IEs from different sources (e.g., transaction data and service metadata) will be researched.

5.2.2.2 Task 7.2

The recommendation functionality will be increased in this task and phase by moving toward a hybrid recommender. Hence, different approaches to combine CF and CB recommendation are researched targeting the cold-start problem. The recommendation quality should get better terms of accuracy and personalisation.

5.2.2.3 Task 7.3

Based on the first phase's experience in providing our own evaluation components, more complex evaluation components, dedicated to higher level services, will be encouraged by the provision of documentation and by the direct deployment in agreement with the, at-that-time participating actors.

6 Conclusion

This document presented the “Technology Specification” as well as the “Development Roadmap” of the Broker and Assessment services to be developed in WP7. The document covered multiple aspects of the topics IE, recommender, and service assessment comprising WP7. The document starts by giving an overview about the current state of the art for each of the topics. In the field of IE, traditional IE methods and OIE approaches are presented. Regarding recommender systems, recommendation approaches, and the brokerage model are discussed. Evaluation metrics from information retrieval and machine leaning make up the last part of the state of the art review.

In the second part of this deliverable, already existing software components proving the foundation for the work in WP7 are described. Different IE libraries and systems capable of OIE are presented alongside the ScaR framework. The description of evaluation components and their deployment in a Docker-based infrastructure closes this chapter.

The third part describes the software architecture for the components developed in WP7. In this chapter, the components and the interfaces between them are outlined. Furthermore, the communication with other parts of the DMA core infrastructure is described. The presented architecture is designed to integrate well into the general DMA core infrastructure. The components described in the second part, with the ScaR framework leading the way, are well suited to be integrated in the architecture of WP7 and henceforth in the DMA core infrastructure.

The fourth part outlines the development plan for WP7 and describes the two phases of the undertakings in WP7. The first phase will focus on defining the interfaces to other components of DMA. After these definitions are agreed upon, a first version of the components will be implemented, realising a preliminary system. In the second phase the emphasis lies on improving and extending the components functionality and quality of service.

Currently, the metadata and interface definitions are in active development. Ongoing discussions with the other work packages have already lead to preliminary definitions. These definitions will be finalised in the next couple of weeks. With the interface definitions and further input from the requirements described in Deliverable D2.2, the system architecture will be validated. After this validation is done, the first system implementation can start. At the beginning of the second phase there will also be an extensive screening, validation, and evaluation of the then current implementation state. The results of this evaluation will determine the detailed work planning for the second phase.

References

[1] M.-F. Moens, Information Extraction: Algorithms and Prospects in a Retrieval Context, Dordrecht, Netherlands: Springer, 2006, p. 246.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni, “Open Information Extraction from the Web,” in *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, Hyderabad, India, 2007.

[3] T. Poibeau, H. Saggion, J. Piskorski and R. Yangarber, Eds., Multi-Source, Multilingual Information Extraction and Summarization, Berlin Heidelberg, Germany: Springer, 2013.

[4] C. Fellbaum, Ed., WordNet: An Electronic Lexical Database and Some of its Applications, Cambridge, USA, Massachusetts: MIT Press, 1998.

[5] D. Hindle, “Noun Classification from Predicate-argument Structures,” in *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics (ACL'90)*, Pittsburgh, USA, 1990.

[6] M. A. Hearst, “Automatic Acquisition of Hyponyms from Large Text Corpora,” *Proceedings of the Fourteenth International Conference on Computational,* pp. 539-545, July 1992.

[7] D. C. Wimalasuriya and D. Dou, “Ontology-based Information Extraction: An Introduction and a Survey of Current Approaches,” *Journal of Information Science,* vol. 36, no. 3, pp. 306-323, June 2010.

[8] E. Riloff and R. Jones, “Learning Dictionaries for Information Extraction by Multi-level Bootstrapping,” in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence (AAAI '99)*, Menlo Park, USA, 1999.

[9] D. S. Weld, R. Hoffmann and F. Wu, “Using Wikipedia to Bootstrap Open Information Extraction,” *SIGMOD Record,* vol. 4, pp. 62-68, March 2009.

[10] F. Ricci, L. Rokach and B. Shapira, “Introduction to recommender systems handbook,” 2011.

[11] G. Shani and A. Gunawardana, “Evaluating Recommendation Systems,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, Eds., Springer US, 2011, pp. 257-297.

[12] C. Rana and S. K. Jain, “A study of the dynamic features of recommender systems,” *Artificial Intelligence Review,* vol. 43, pp. 141-153, 2015.

[13] M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*, Springer, 2007, pp. 325-341.

[14] J. B. Schafer, D. Frankowski, J. Herlocker and S. Sen, “Collaborative filtering recommender systems,” in *In The adaptive web*, 2007.

[15] H. Ma, D. Zhou, C. Liu, M. R. Lyu and I. King, “Recommender systems with social regularization,” in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, Hong, 2011.

[16] Q. Liu and D. R. Karger, “Kibitz: End-to-End Recommendation System Builder,” in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015.

[17] E. Lacic, “Real-Time Recommendations in a Multi-Domain Environment,” in *27th ACM Conference on Hypertext and Social Media (HT'2016)*, 2016.

[18] E. Lacic, M. Traub, D. Kowald and E. Lex, “ScaR: Towards a Real-Time Recommender Framework Following the Microservices Architecture,” in *Proceedings of the Workshop on Large Scale Recommender Systems (LSRS2015) at RecSys 2015*, Vienna, Austria, 2015.

[19] Y. Huang, B. Cui, W. Zhang, J. Jiang and Y. Xu, “TencentRec: Real-time Stream Recommendation in Practice,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[20] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme and W. Nejdl, “Real-time top-n recommendation in social streams,” in *Proceedings of the sixth ACM conference on Recommender systems*, 2012.

[21] B. Chandramouli, J. J. Levandoski, A. Eldawy and M. F. Mokbel, “StreamRec: A Real-time Recommender System,” in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, 2011.

[22] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” in *Knowledge and Data Engineering, IEEE Transactions on*, 2005.

[23] P. Lops, M. De Gemmis and G. Semeraro, “Content-based recommender systems: State of the art and trends,” in *In Recommender systems handbook*, 2011.

[24] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” in *Information processing & management*, 1988.

[25] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, “Application of dimensionality reduction in recommender system-a case study,” 2000.

[26] E. Lacic, D. Kowald, M. Traub, G. Luzhnica, J. Simon and E. Lex, “Tackling Cold-Start Users in Recommender Systems with Indoor Positioning Systems,” in *In Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'2015)*, 2015.

[27] Y. Koren, R. Bell and C. Volinsky, “Matrix factorization techniques for recommender systems,” in *Computer*, 2009.

[28] D. Bissinger, “Brokering Data and Processes, What Does it Mean?”.

[29] T. Laleh, S. Mokhov, J. Paquet and Y. Yan, “Context-Aware Cloud Service Brokerage: A Solution to the Problem of Data Integration Among SaaS Providers,” 2015.

[30] B. Veloso, B. Malheiro and J. Burguillo, “A Multi-Agent Brokerage Platform For Media Content Recommendation,” in *Int. J. Appl. Math. Comput. Sci.*, 2015.

[31] I. Patiniotakis, Y. Verginadis and G. Mentzas, “Preference-based Cloud Service Recommendation as a Brokerage Service,” in *Workshop on CrossCloud Brokers’14*, 2014.

[32] C. Cleverdon, Report on the Testing and Analysis of an Investigation into the Comparative Efficiency of Indexing Systems, Cranfield Collection of Aeronautics, 1962.

[33] D. Harman, “Overview of the First Text REtrieval Conference (TREC-1),” in *Proc. of {TREC}*, 1992.

[34] M. Sanderson, “Test Collection Based Evaluation of Information Retrieval Systems,” *Foundations and Trends in Information Retrieval,* vol. 4, 2010.

[35] T. G. Armstrong, A. Moffat, W. Webber and J. Zobel, “EvaluatIR: An Online Tool for Evaluating and Comparing IR Systems,” in *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2009.

[36] M. Dussin and N. Ferro, “DIRECT: applying the DIKW hierarchy to large-scale evaluation campaigns,” in *Proceedings of {JCDL}*, 2008.

[37] L. Azzopardi, Y. Moshfeghi, M. Halvey, R. S. Alkhawaldeh, K. Balog, E. Di Buccio, D. Ceccarelli, J. M. Fernández-Luna, C. Hull, J. Mannix and others, “Lucene4IR: Developing Information Retrieval Evaluation Resources using Lucene,” *ACM SIGIR Forum,* vol. 50, 2016.

[38] H. Müller, J. Kalpathy-Cramer, A. Hanbury, K. Farahani, R. Sergeev, J. H. Paik, A. Klein, A. Criminisi, A. Trister, T. Norman, D. Kennedy, G. Srinivasa, A. Mamonov and N. Preuss, “Report on the Cloud-Based Evaluation Approaches Workshop 2015,” *SIGIR Forum,* vol. 50, pp. 38-41, #jun# 2016.

[39] NIST, “IAD DSE Pilot Evaluation Plan,” 2016.

[40] A. McAfee and E. Brynjolfsson, “Big Data: The Management Revolution,” *Harvard Business Review,* vol. 90, pp. 60-68, 2012.

[41] M. Sarwat, J. Avery and M. F. Mokbel, “RecDB in Action: Recommendation Made Easy in Relational Databases,” *Proc. VLDB Endow.,* vol. 6, pp. 1242-1245, #aug# 2013.

[42] M. Steurer and C. Trattner, “Acquaintance or Partner?: Predicting Partnership in Online and Location-based Social Networks,” in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara, 2013.

[43] E. Lacic, D. Kowald, L. Eberhard, C. Trattner, D. Parra and L. Marinho, “Utilizing Online Social Network and Location-Based Data to Recommend Products and Categories in Online Marketplaces,” in *Mining, Modeling, and Recommending 'Things' in Social Media*, Springer, 2015, pp. 96-115.

[44] S. Bostandjiev, J. O'Donovan and T. Höllerer, “TasteWeights: a visual interactive hybrid recommender system,” in *Proc.*, 2012.

[45] D. Parra-Santander and P. Brusilovsky, “Improving Collaborative Filtering in Social Tagging Systems for the Recommendation of Scientific Articles,” in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, 2010.

[46] R. Kohavi, R. M. Henne and D. Sommerfield, “Practical Guide to Controlled Experiments on the Web: Listen to Your Customers Not to the Hippo,” in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[47] R. Burke, “Hybrid recommender systems: Survey and experiments,” in *User modeling and user-adapted interaction*, 2002.

[48] D. Somefu and J. La Poutr´e, “Bundling and Pricing for Information Brokerage: Customer Satisfaction as a Means to Profit Optimization,” in *IEEE/WIC International Conference on Web Intelligence*, 2003.

[49] E. Lacic, M. Traub, D. Kowald and E. Lex, “ScaR: Towards a Real-Time Recommender Framework Following the Microservices Architecture,” in *In Workshop on Large Scale Recommender Systems (LSRS'2015) co-located with the 9th ACM Conference on Recommender Systems (RecSys'2015)*, 2015.