Crowd-powered recommendation for continuous digital media access and exchange in social networks

FP7-610594

D2.4

Second Reference Framework Release and Evaluation Report

Dissemination level:	Public
Contractual date of delivery:	M18, 31 March 2015
Actual date of delivery:	M18, 31 March 2015
Workpackage:	WP2. Requirements and Reference Framework
Task:	T2.3 Reference framework implementation T2.4 Reference framework evaluation
Type:	Report
Approval Status:	Final
Version:	1.0
Number of pages:	40
Filename:	D2.4_SecondReferenceFrameworkRelease_and_EvaluationReport_v1.0

Abstract

This deliverable reports the results achieved in WP2 "Requirements and Reference Framework" about the second release of the reference framework and its evaluation. With respect to the first release (Deliverable D2.2 "First Reference Framework Release and Evaluation Report"), the four-component architecture of the reference framework – publicly known as **Idomaar** – has been re-designed in order to fulfill the new requirements collected in Deliverable D2.3 "Second Iteration Requirements".

Idomaar continues to make possible experimenting with recommendation algorithms in an architecture-independent environment that grants the consistency and reproducibility of results. In this release, we focused on the integration of open-source, state-of-the-art technologies enabling the management and processing of stream of data in a distributed and scalable way. The second release of Idomaar is being used in the settings of the NewsReel challenge.

Next iterations will consolidate the revised data workflow and will expand the collection of ready-to-use recommendation algorithms distributed within Idomaar.

co-funded by the European Union

History

Version	Date	Reason	Revised by
0.1	2015.03.02	Created template	Roberto Turrin
0.3	2015.03.18	Progress in architecture, NewsReel	Roberto Turrin
0.4	2015.03.20	Progress in dissemination/development	Roberto Turrin
0.5	2015.03.23	Progress in introduction	Roberto Turrin
0.6	2015.03.24	Evaluator and data format	Andrea Condorelli
0.8	2015.03.27	Revision of the full report	Roberto Turrin
0.9	2015.03.31	Finalized draft, ready for final review	Roberto Turrin
1.0	2015.03.31	Final review of draft, ready to submit	Martha Larson

Author list

Organization	Name	Contact Information
MOV	Roberto Turrin	roberto.turrin@moviri.com
MOV	Davide Malagoli	davide.malagoli@moviri.com
TUB	Andreas Lommatzsch	andreas@dai-lab.de
GRA	András Serény	sereny.andras@gravityrd.com
MOV	Andrea Condorelli	andrea.condorelli@moviri.com
GRA	Balázs Hidasi	hidasi.balazs@gravityrd.com
TUD	Mark Melenhorst	M.S.Melenhorst@tudelft.nl
TUD	Martha Larson	m.a.larson@tudelft.nl

Executive Summary

This deliverable describes the second release of the reference framework – **Idomaar** – and its evaluation at the end of the first semester of the second year (M18). The document contributes to Milestone MS2 "Second Reference Framework Release and First Iteration Real-world Deployment".

This release takes into account the new requirements collected in D2.3 "Second Iteration Requirements" and implements the functionalities anticipated in the outlook of deliverable D2.2 "First Reference Framework and Evaluation Report" by (i) targeting the evaluation of stream of data and (ii) expanding the Idomaar's components.

In the second release of Idomaar we mainly focused on re-designing the full data workflow. We re-implemented the orchestrator, simplifying the process and introducing new state-of-the-art technologies to expand the capabilities to treat stream recommendations.

In addition to the orchestrator, also the evaluator has adopted a portion of such state-of-the-art technologies, in order to process a stream of data in a distributed and scalable way.

Finally, Idomaar is being tested in the settings of a real application: the NewsReel challenge. The added value of applying Idomaar mainly derives from the fact that it grants consistent and reproducible results. In fact, Idomaar provides a "safe" testing environment where: (i) the stream of data is constrained, (ii) everyone is constrained with respect to the computing resources, (iii) business metrics such as the response time can be measured within this controlled environment.

The second release of Idomaar affected all the components of the reference framework.

Datasets. The data format has been further validated, existing data have been enriched and new data sets have been added.

Orchestrator. The orchestrator uses advanced, scalable and distributed technologies such as Apache Flume, Apache Kafka, and Zookeeper to stream training and test data, as well as requesting for recommendations.

Evaluator. The evaluator evolved by integrating the same new stream technologies as adopted by the orchestrator.

Computing environment. The computing environment has to expose interfaces to the innovative technologies in order to receive the stream of data. In particular, it can communicate either using the HTTP protocol or Apache Kafka.

The core stream evaluation functionalities of Idomaar are reaching a satisfactory level of maturity. We expect the NewsReel challenge will provide us additional feedback that will be taken into consideration in the next reference framework releases to further consolidate the Idomaar data stream processing. In addition to such improvement deriving from the first real-

world application of Idomaar, we envisage the expansion of the recommendation algorithms being developed in WP3 “Stream Recommendation Algorithms” and WP4 “Crowd Engagement Algorithms” and that are to be evaluated and compared to derive a selection of solutions to be deploy and tested in WP5 “Large-scale real-world application and deployment”.

Table of Contents

1. Introduction

This deliverable describes the main results achieved in WP2 “Requirements and Reference Framework” during the first semester of the second year and it contributes to Milestone 2, in particular focusing on the **second reference framework release and evaluation report**. The reference framework implementation (Task T2.3 “Reference Framework Implementation”) has been driven by the requirements elicited from the social network and the SME partners and collected in Deliverable D2.3 “Second Iteration Requirements” (Task T2.1 “Requirements”). Finally, the current release of the reference framework has been evaluated in Task T2.4 “Reference framework evaluation” to understand its suitability for deployment in large-scale environments.

The reference framework, as already mentioned in the deliverable D2.2 “First Reference Framework Release and Evaluation Report” (Section 1) has been publicly disseminated with the name **Idomaar**, which means animal *tamer* in a Hungarian dialect.

The main activities concerned the addressing of real-world use cases and the convergence of the framework towards state-of-the-art and innovative technologies used by enterprise systems to manage big data streams in a distributed and scalable way. We identified the following examples of applications of Idomaar:

- Implementation benchmarking: an "agnostic" appraisal platform able to test both the algorithmic effectiveness and the implementation efficiency.
- Tendering – external evaluation: a benchmarking tool able to evaluate and compare different recommendation solutions for operators and assist business in selecting the one that fits best to its needs.
- Algorithm benchmarking – internal evaluation: a method that is capable of comparing recommendation algorithms working on either static or stream data.
- 3D evaluation: a tool able to perform end-to-end measurement of several dimensions of any recommender algorithm, such as the quality, robustness, and scalability of each implementation.

Several improvements have been implemented in Idomaar with respect to the previous release described in deliverable D2.2 “First Reference Framework Release plus Evaluation Specifications”, in particular:

- Integration with “big data” and data streams
- Use of scalable and “production-ready” technologies (e.g., Apache Flume and Kafka)
- Integration of state-of-art machine learning components (e.g., Apache Spark)
- Generalization of the evaluation process
- Easy integration with Newsreal challenge

The progress involved all the reference framework’s components, as detailed in the following.

Datasets. The data format has been further validated. In addition, the dataset FilmTweetings was enhanced with IMDb data, and the CLEF NewsReel 2015 dataset was released to the NewsReel challenge participants.

Evaluator. Evaluation strategies and metrics. The evaluator evolved in accordance with the integration of the new stream technologies. In particular, the evaluator was divided into two tasks, the dataset splitting and the recommendation evaluation. Both tasks consist of simple scripts using programming tools such as Python and Apache Spark, allowing a straightforward extension with custom solutions.

Orchestrator. The orchestrator uses advanced, scalable and distributed technologies such as Apache Flume, Apache Kafka, and Zookeeper to stream training and test data to the computing environment, as well as requesting for recommendations to the computing environment.

Computing environment. The computing environment has to expose interfaces to the innovative technologies in order to receive the stream of data. In particular, it can communicate either using the HTTP protocol or Apache Kafka. The support to HTTP will provide interfaces and communication protocols common between task 1 and task 2 of the NewsReel challenge, thus facilitating the participants who will have to expose a single interface.

The development of the second release of the reference framework – Idomaar – required multiple interactions between the partners involved in the implementation, especially in order to define which changes and improvements were necessary to accomplish the new requirements defined in Deliverable D2.3 *"Second Iteration Requirements"*. We came out, in the end, with a selection of open-source, state-of-the-art technologies that are getting popular in the settings of several enterprise realities that regularly receive, process, and store big streams of data.

In March, we organized a workshop at Moviri in Milan both to share the new release of Idomaar with the partners in the CrowdRec consortium and to describe and motivate the technological changes in the reference framework. The event was open to the public and a number of students (about 60) from the university Politecnico di Milano attended the courses.

A specific effort has been spent in making Idomaar compliant with the NewsReel challenge (task 2), that is being acting as a practical prove of the capabilities of Idomaar. We expect to receive several feedback while the challenge is running as well as at the end, which will further help us in improving the framework.

The activities involved in this deliverable concern the following two tasks.

T2.3 - Reference framework implementation

The task refers to the implementation of the reference framework, whose goal is allowing testing a set of recommendation algorithms before being deployed in real-world scenarios for user tests.

The reference framework provides (i) the environments to execute, experiment, and test recommendation algorithms, (ii) a set of implemented recommendation algorithms, and (iii) the data set to use.

At the end of the second semester of the second year we have completed the second release of the reference framework Idomaar, which is being adopted to evaluate the second task of the NewsReel competition, proving its capabilities in a real domain.

The task will continue during the rest of the project, implementing any feedback that will come out during the NewsReel challenge, supporting the deployment of recommendation algorithms, integrating further evaluation strategies and metrics, and including new datasets.

T2.4 - Reference framework evaluation

The task refers to the evaluation of the algorithms that are implemented in the reference framework in Task T2.3, testing their performance and suitability for deployment in large-scale social networks.

The evaluation of such algorithms firstly requires to evaluate the Reference Framework itself by:

- verifying that it meets the requirements, e.g., consistency and reproducibility, architecture independence, and use of standard components
- granting that it fits the 3D model in terms of business, technical, and user dimensions
- validating its feasibility on real work applications (e.g., NewsReel scenario)

The task will proceed during the rest of the project to constantly grant the suitability of the reference framework for the eventual deployment in large-scale environments.

1.1. Document outline

The rest of the document is organized as follows. Section 2 describes the main driving requirements reported by Deliverable D2.3 "Second Iteration Requirements", in particular emphasizing the changes with respect to previous requirement cycle. Section 3 presents the NewsReel challenge: the scenario, the motivation, the dataset, the evaluation, explaining the role of Idomaar. The architecture of Idomaar is illustrated in Section 4, where the new data workflow and the state-of-the-art technologies for data streaming processing are detailed. Section 5 presents the development process and the dissemination activities. Section 6 report the evaluation of the current version of the reference framework. Finally, Section 7 draw some conclusions and plan the activities of the next iterations.

2. Driving requirements

The continued development of the Idomaar reference framework has driven by the updated requirements specifications in D2.3 “Second Iteration Requirements”, Section 6.1. The changes were the result of feedback from the first year review and the definition of CrowdRec Collaborative Products, leading to new requirements and shifted priorities. This particularly holds for NewsReel. Below we summarize the Idomaar requirements that have been changed since the first requirements iteration.

8	**Data sources**	Idomaar must be able to handle stream datasets.	Higher priority
9	**Data sources**	Idomaar must be able to access and evaluate algorithms accessing datasets on “big data” platforms such as Amazon S3 or HDFS.	New requirement
10	**Production readiness**	Idomaar must interface with computing environments with interfaces and components that could be easily replaced to let the algorithms work in a production environment.	Higher priority
11	**Production readiness**	Idomaar must use components that are widely adopted by the open source community in order to involve as much contributor as possible and to promote the use of the framework.	Higher priority
13	**Production readiness**	Implementation of support for a specific real-world use scenario, as a case study: Newsreel	Higher priority
12	**Documentation**	Idomaar must have a public accessible documentation that could be easily used to create a full end-to-end process to evaluate algorithms.	higher priority

As stated in D2.3 “Second Iteration Requirements”, for the second Idomaar release the following focal points were defined:

- Incorporating improvements that were implemented to support Newsreel
- Providing adequate documentation on Github
- Integration and release of new components, as advancements to the basic architecture

Information on how these requirements were addressed is integrated into the sections of the deliverable, at the points that it is most relevant.

3. Newsreel

NewReel is a two-task challenge organized by CLEF 2015 (http://clef2015.clef-initiative.eu/CLEF2015/) lab that addresses real-time news recommendation. In particular, the second task simulates the real-time recommendation task using Idomaar as reference framework for the evaluation. This works as a proof-of-concept of the functionalities of Idomaar in testing and experimenting with the fair and reproducible comparison of recommendation algorithms.

In particular, as mentioned in *D2.3 "Second Iteration Requirements",* Section 6.2, the deployment of Idomaar in the settings of NewsReel will provide validation for some key features of the reference framework Idomaar:

- Architecture independent: the participants will have the option to use their preferred environments, the only requirement being connecting to two given communication interfaces.
- Consistency and reproducibility: the evaluation is fair and consistent among all participants.
- Stream management: CrowdRec is meant to manage stream recommendation algorithms. NewsReel will be used to prove the effectiveness of the state-of-the-art technologies implemented by Idomaar to manage stream of data.

3.1. Motivation

The evaluation of algorithms is crucial for benchmarking and optimizing algorithms. Over the years, various data corpora have been released covering different domains such as Patents, Blogs, or Multimedia. Most of these corpora are static and do not provide an order of the elements of the dataset. The evaluation based on static dataset is typically built on cross-validation splitting the dataset randomly into a training set and a test set.

In our fast changing world, algorithms taking into account current trends and changes in the user preferences get in the focus of interest. Powerful recommender algorithms should be able to predict the relevance of new items and support the user in finding fresh interesting items matching the individual preferences.

The development of algorithms and evaluation metrics for stream-based scenarios is still an important research topic. CLEF NewsREEL boosts the development of new recommender approaches by providing large datasets as well as a framework enabling the reproducible, context-aware evaluation of stream-based recommender algorithms. The NewsREEL lab provides components for the online as well as the offline evaluation of recommender algorithms. Beside the recommendation precision also technical as well as business-model oriented aspects are considered in the evaluation.

3.2. NewsReel scenario and the related challenges

In the CLEF NewsREEL challenge we analyze the problem of recommending news articles in a web portal. The task consists in recommending a user news articles most relevant in the specific context. The recommendations are presented embedded in a news web page; the performance of the recommender is measured based on the Click-Through-Rate (CTR) describing the proportion of clicked recommendations to the number of recommendations requests.

3.2.1. News recommendation challenges

Recommending news articles leads to several different challenges:

1. In contrast to traditional recommender systems working based on a static set of users and items, the set of valid users and items is highly dynamic in the news recommendation scenario. New articles must be added to the recommender model; outdated news articles must be removed in order to ensure that the recommended articles are "new". Thus, one big challenge of the news recommender system is the continuous cold-start problem: New articles potentially more relevant than old articles are only sparsely described by meta-data or collaborative knowledge.
2. Noisy user-IDs are an additional challenge in the analyzed web-based news recommendation scenario. Since the users do not have to explicitly register on the news portals the user tracking is implemented based on cookies and JavaScript. Some of the users may apply obfuscating tools (such as Ad-Blocker) leading to noisy user-IDs. The implemented recommender algorithm must be aware of the challenge and should apply algorithms providing highly relevant recommendations even if the user tracking is noisy.
3. The user preferences in news highly depend on the domain and on the hour of the day. In the morning users usually do not have much time. Thus the users are interested in the top news from the domains politics and sport. In the evening the users usually have more time reading also longer, detailed news articles from diverse domains. Thus the news recommender algorithms must consider different aspects of context such as the news domain, the time of the day and the user's device.
4. In the online news recommendation scenario there is the additional constraint, that the requests must be answered within 100ms. The response time constraint is defined due to the requirement that the recommendations must be integrated seamlessly into the online news web page.

As discussed the news recommendation scenario raises several challenges. An additional problem is, that news articles usually depend on current events that cannot predicted. Thus a recommender trained on data collected in the past may perform poorly in the future due to unexpected events and changes in the user preferences.

3.2.2. Challenges while evaluating stream-based recommendations

Recommender systems implemented for handling are usually evaluated using the metrics Precision, Mean-Average-Precision or Root-Mean-Squared Error. In the academic evaluation a dataset is usually split into a training set and a test set using the principle of cross-validation.

The evaluation of streams has its specific challenges induced by the continuous changes in the stream. Usually, the baseline recommendation precision changes over one day and is strongly influenced by the context. Thus, a longer period of time (based on similar contexts) should be considered when comparing two recommender algorithms. Another big challenge when benchmarking recommender systems online in a living lab setting is that the experiments cannot be reproduced since the measured recommendations precision is based on direct user feedback.

3.3. Online vs. offline evaluation

In the CLEF NewsREEL lab two tasks are offered.

Task 1 focusses on the online evaluation. The recommender teams register at the ORP web site (powered by the Plista GmbH). When a user visits a news web page part of the NewsREEL challenge, a recommendation request is sent to a registered team. The recommender team has to provide up to 6 recommendations. The time constraint for completing the recommendation request is 100ms. In addition to the recommendation requests there are messages describing the creation, removal, or update for news articles. The performance of the recommender algorithms is measure based on the CTR for predefined weeks.

Task 2 focusses on the offline evaluation of stream-based recommender algorithms. The offline evaluation enables the reproducible evaluation of different recommender algorithms on exactly the same data. In addition, different parameter configuration for one algorithm can be analyzed in detail. Beside the analysis of the recommendation precision Task 2 also enables the analysis of the technical complexity of different algorithms. Using virtual machines simulating different hardware settings the offline setting allows us investigating the effect of the hardware settings and the load level on the response time and the recommendation precision.

Since Task 1 and Task 2 use very similar data formats implemented recommender algorithms can be easily tested in both online and offline evaluation. This allows the comprehensive evaluation of the strength and weaknesses of the implemented strategies.

3.4. The dataset

The basis for the dataset used in the offline evaluation is a stream of messages in the online scenario.

The communication between the ORP server and the recommender algorithms in Task 1 is based on an http-based web service. The messages consist of a message type and payload field. The payload field is formatted as JSON. The JSON data are organized in a tree containing information about the news item, the user (as well as the user's environment), and the user-item interaction. Recommendation request must also be answered based on a message formatted as JSON providing a list of recommended articles.

The dataset used in the offline evaluation has been recorded between July 1st, 2014 and August 31st, 2014. The dataset describes three different news portals: One portal providing general as well as local news, the second portal provides sport news; the third portal is a discussion board providing user generated content.

In total the dataset contains approximately 100 million messages.

	Item create/ item update	**User-item interactions**	**sum**
July 2014	618487	53323934	53942421
August 2014	354699	48126400	48481099
sum	973186	101450334	102423520

3.4.1. The dataset format

For representing the data, CLEF NewsREEL uses a hybrid data format combining the strength of the well-structured tab-separated value (TSV) format and flexible structured JSON data format.

Motivation

The central information in the news recommendation scenario are the news items and the interaction of users with the news items. In contrast most traditional recommendation dataset, users may have multiple interactions with one item. Thus, data structure used in NewsREEL is inspired by a multi-graph allowing us representing several different interactions between one user u and a news item i.

The Data Structure

For representing the news items as well as the user-item interactions we use a 5-column data format. The columns are separated by tab characters. Each of the 5 columns can either contain an atomic value (typically a long value) or a JSON string.

The News Items

News items in the NewsREEL dataset describe items published on a news portal. The 5 columns provide the following information:

(1) The first column defines whether a new item has been created or whether an existing item has been updated. Valid identifiers are the strings: "item-created" and "item-updated".
(2) The second column provides a unique identifier for the news item. The item-ID is provided as a long value.
(3) The third column provides a timestamp.
(4) The fourth column provides detailed meta-data for the news item, e.g. the news article's URL or the item abstract. This column is formatted in JSON.
(5) The fifth column provided information to linked resources such as the news domain. This field is also formatted as JSON.

Information about removed or outdated items is not part of the dataset.

The User-Item Interactions

The user-item interactions are also described in a 5-column data format. The columns provide the following information:

(1) The first column defines the type of interaction. In the dataset the type is for all interaction "impression".
(2) The second column provides a unique identifier for the user-item interaction. The ID is represented as a long value.
(3) The third column provides a timestamp for the user-item interaction.
(4) The fourth column describes the meta-data of the interaction. This field is formatted as JSON. The column provides detailed information about the user's environment (e.g. used device) and assumptions about the user.
(5) The fifth column provides references to linked resources, such as the user, the new item, and the news domain. The column is formatted using JSON.

The information whether recommendations must be provided for an impression is stored in column 4. If recommendations are expected the JSON object sets to true.

Examples

In this subsection exemplary lines from the dataset are explained.

Item created

item-created <TAB> 186529858 <TAB> 2014-07-01 00:25:53 <TAB> {"domainid": 1677, "img": "", "title": "Ken Jebsen: \"Ich wei\u00df, dass ich h\u00f6her getaktet bin\"", "url": "http://www.tagesspiegel.de/meinung/radio-moderator-ken-jebsen-ich-weiss-dass-ich-hoeher-getaktet-bin/5813468.html&hspart=ddc&hsimp=yhs-ecosia_03&vm=p", "text": "Dem Moderator der RBB-Radioshow KenFM wird Antisemitimus vorgeworfen. Ken Jebsen selbst sagt:", "created_at": "2011-11-08 17:25:00", "updated_at": "2014-07-01 00:25:53", "flag": 8, "version": 1, "kicker": "Radio-Moderator", "id": 186529858} <TAB> {"itemID":186529858, "domainID":1677}

Impression

impression <TAB> 9471305001677 <TAB> 1404165599401 <TAB> {"recs": {"ints": {"3": [186337307, 183665084, 186097159, 186236234, 180818566, 186374324]}}, "event_type": "recommendation_request", "context": {"simple": {"62": 1979832, "63": 1840689, "49":

```
48, "67": 1928642, "68": 1851453, "69": 1851422, "24": 2, "25": 186312774, "27":
1677, "22": 61970, "23": 23, "47": 654013, "44": 1851485, "42": 0, "29": 17332, "40":
1788350, "41": 25, "5": 280, "4": 40293, "7": 18873, "6": 952253, "9": 26889, "13":
2, "76": 1, "75": 1919968, "74": 1919860, "39": 748, "59": 1275566, "14": 33331,
"17": 48985, "16": 48811, "19": 52193, "18": 5, "57": 71131568, "56": 1138207, "37":
1978123, "35": 315003, "52": 1, "31": 0}, "clusters": {"46": {"472375": 100,
"472420": 100, "472376": 100}, "51": {"2": 255}, "1": {"7": 255}, "33": {"32962448":
6, "2559246": 2, "328109": 8, "10758": 1, "2033354": 0, "556907": 7, "236223": 3,
"4146": 5, "39698": 3, "399975": 10, "60664": 2, "32941223": 5, "7354": 7, "101707":
1, "2566836": 1, "51732": 2, "404677": 5}, "3": [55, 28, 34, 91, 23, 21], "2": [11,
11, 61, 60, 61, 26, 21], "64": {"4": 255}, "65": {"1": 255}, "66": {"12": 255}},
"lists": {"11": [13839], "8": [18841, 18842, 48511], "10": [6, 13, 1768, 1769,
1770]}}, "timestamp": 1404165599401} <TAB> {"userID":71131568, "itemID":186312774,
"domainID":1677}
```

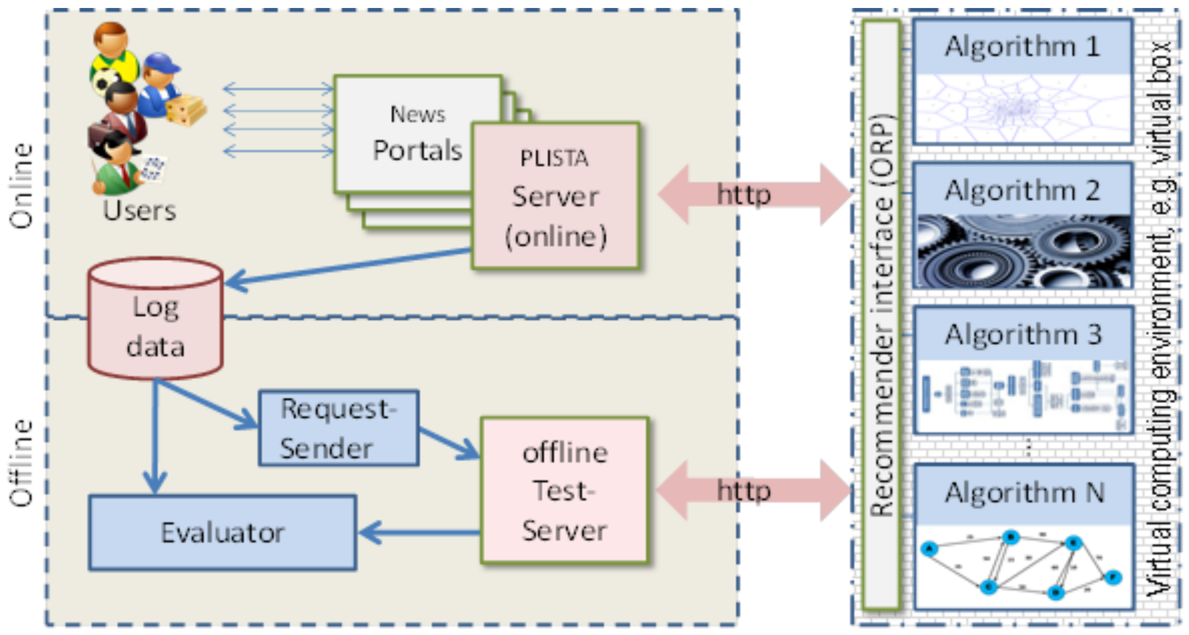
3.4.2. Discussion

In this section, we explained the format used in the NewsREEL challenge. The five-column data format allows an efficient handling of large dataset. The JSON format (used in 2 of the five columns) ensures the flexibility for storing all relevant data tracked by the news portals. The parsing of the dataset files is well supported by almost all popular programming languages since powerful libraries for parsing JSON and TSV files exist.

In order to build a recommender algorithm based on a collaborative filtering algorithm, details about news items are not needed. The required information can be extracted from the fifth column of the user-item interactions. The news item text needed when implementing a content-based recommender algorithm can be obtained from the meta-data of the news items.

3.5. The architecture

The basic architecture of NewsREEL consists of a request sender, a recommender component and an evaluator.



3.6. Evaluation

3.6.1. Benchmarking Metrics

In the NewsREEL evaluation we analyze different aspects.

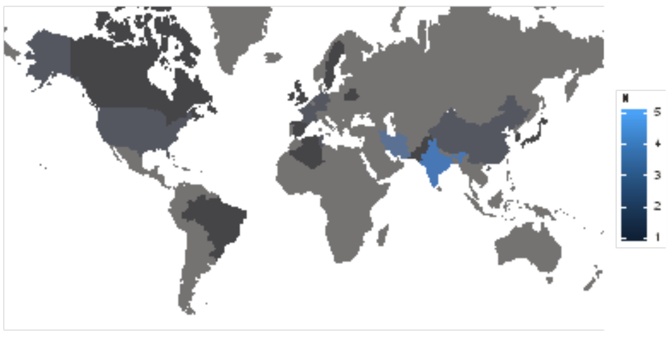
1. The recommendation precision is measured based on the Click-Through-Rate (CTR)
2. For measuring the computational complexity, the CPU-time as well as the amount of RAM is analyzed. In general, an algorithm is the more powerful the less resources are needed for completing a task. The amount of required resources is measured using an exactly defined environment defined by a virtual machine (e.g., a vagrant-build VM powered by virtual box)
3. A third important aspect for benchmarking the recommender performance is the number of requests that can be handled in a given period of time. In NewsREEL, the throughput and the number of requests completed within a given timeline are analyzed. The throughput usually indicates how well an algorithm can be processed in concurrent threads. If an algorithm does not make use of available resources (due to a lack of parallelization) additional CPUs or RAM may not increase the number of requests that can be handled in a given time slot.
4. Based on the reward for a valid recommendation and the effort required for provided the recommendation a business model can be derived. An added value for the customer should be higher than the effort needed for providing the recommendations.

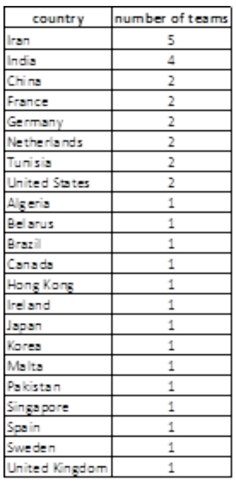
3.6.2. Context-aware evaluation

The relevance of news articles does not only depend on the users and the items, but also on the context such as the weekday, the hour of the day, and the user's device. Thus, the evaluator used in the task 2 also allows analyzing the recommender performance with respect to context parameters such as time. Knowing the strength and weaknesses of different recommender algorithms provides the basis for selecting the best matching algorithm for a specific scenario or combining different algorithms in an ensemble. Evaluation results from CLEF NewsREEL 2014 show that context-aware ensembles significantly outperform each of the single recommender algorithms..

3.7. Participants

There are 35 teams from 22 countries registered the CLEF NewsREEL lab. The following figure visualizes the statistic on a world map.





country	number of teams
Iran	5
India	4
China	2
France	2
Germany	2
Netherlands	2
Tunisia	2
United States	2
Algeria	1
Belarus	1
Brazil	1
Canada	1
Hong Kong	1
Ireland	1
Japan	1
Korea	1
Malta	1
Pakistan	1
Singapore	1
Spain	1
Sweden	1
United Kingdom	1

4. Reference framework release

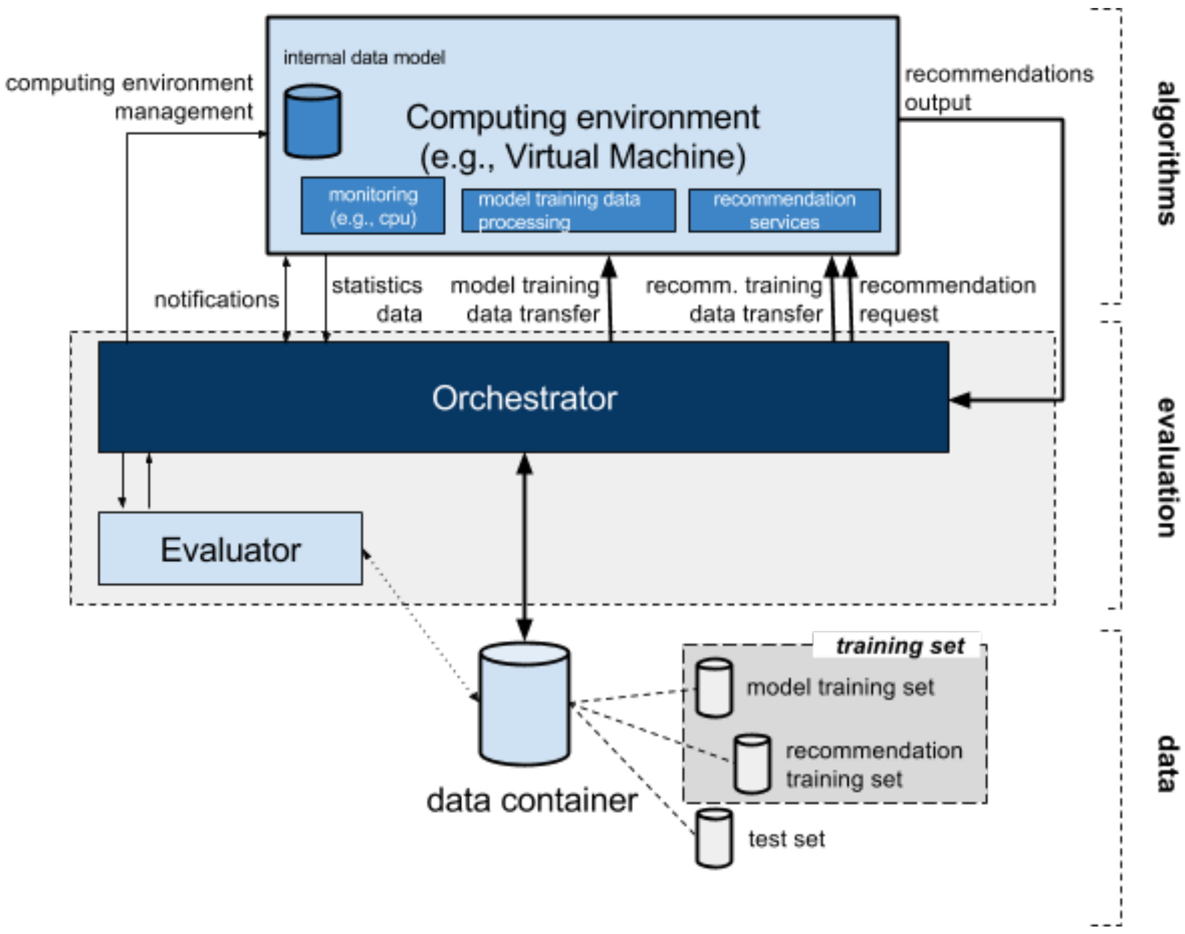
The high-level architecture of the reference framework is stable with respect of previous release, with the main components still being:

- **Data container**. The data container contains all datasets available in the reference framework. Data must be conform to the data model format specified in D2.2 “First Reference Framework Release plus Evaluation Specifications”, Section 5.1.
- **Computing environment**. The computing environment contains the environment to execute an algorithm and the implementation itself. The computing environment must be able to serve recommendation requests and to provide some system statistics (e.g., cpu times, i/o activity). The output format of the computing environment is specified in D2.2 “First Reference Framework Release plus Evaluation Specifications”, Section 5.2.
- **Orchestrator**. The orchestrator is in charge of initiating the virtual machine, providing the training data at the right time, requesting the recommendations, and eventually collecting the results to compute the evaluation metrics.
- **Evaluator**. The evaluator contains the logic to (i) split the dataset according to the evaluation strategy and (ii) compute the quality metrics on the results returned by the recommendation algorithm.

The main progress in Idomaar consists in the adoption of open source, state-of-the-art technologies to handle the management of streams of data.

The advances of each components are summarized in the following:

- **Data container.** The data format has been further validated. In addition, the dataset FilmTweetings was enhanced with IMDb data, and the CLEF NewsReel 2015 dataset was released to the NewsReel challenge participants.
- **Computing environment.** The computing environment has to expose interfaces to the innovative technologies in order to receive the stream of data. In particular, it can communicate either using the HTTP protocol or Apache Kafka.
- **Orchestrator.** The orchestrator uses advanced, scalable and distributed technologies such as Apache Flume, Apache Kafka, and Zookeeper to stream training and test data to the computing environment.
- **Evaluator.** The evaluator evolved in accordance with the integration of the new stream technologies. In particular, the evaluator was divided into two tasks, the dataset splitting and the recommendation evaluation. Both tasks consist of simple scripts using programming tools such as Python and Apache Spark.



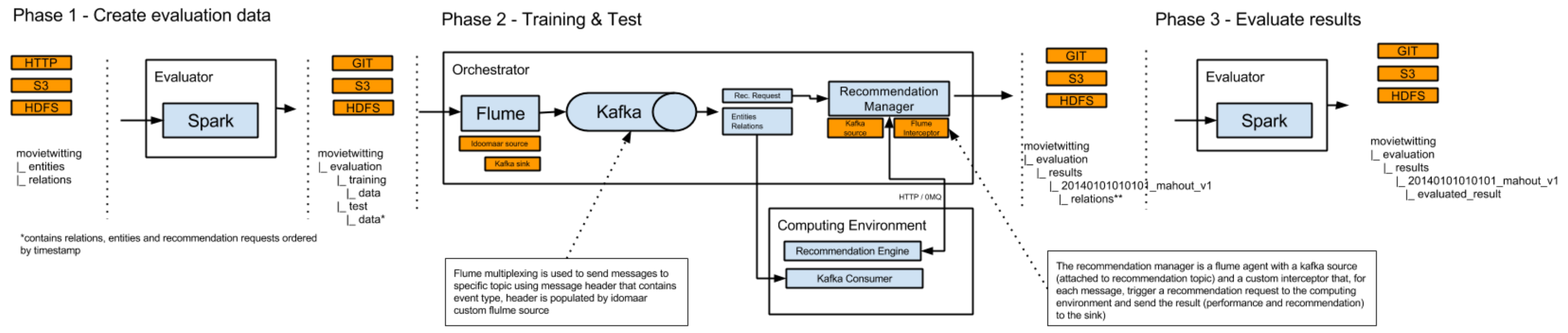
4.1. Idomaar data workflow

The following picture schematizes the full workflow of Idomaar evaluation, from the dataset preparation (i.e., the splitting into training and test sets) to the evaluation a recommendation algorithm.

The workflow can be split into three sequential, separated phases:

1. creation of evaluation data
2. training and testing
3. evaluation of recommendation results

The three phases are illustrated in the diagram below, and then each described in turn.



*contains relations, entities and recommendation requests ordered by timestamp

Flume multiplexing is used to send messages to specific topic using message header that contains event type, header is populated by idomaar custom flulme source

The recommendation manager is a flume agent with a kafka source (attached to recommendation topic) and a custom interceptor that, for each message, trigger a recommendation request to the computing environment and send the result (performance and recommendation) to the sink)

4.2. Phase 1 - creation of evaluation data

The first phase consists in reading the input data (entities and relations), split them into training and test sets, creating the recommendation requests to test the quality of the recommendation algorithm.

The splitting follows the methodology described in D2.2 “First Reference Framework Release plus Requirements Specifications”, Section 4.2: model training set (used for initial training), recommendation training set (the on-the-fly, real-time training data), and test set (the data used for the test).

Operatively, we decided to store the data accordingly to a standardized structure organized in the following folder tree:

- Dataset name
    - Split
        - splitting_filtering-Hash (e.g. random_0.75_subject_from_10_to_100)
            - Train
                - data.ido
            - Test
                - data.ido
            - GroundTruth:
                - data.ido
            - Results
                - recommendations.idop

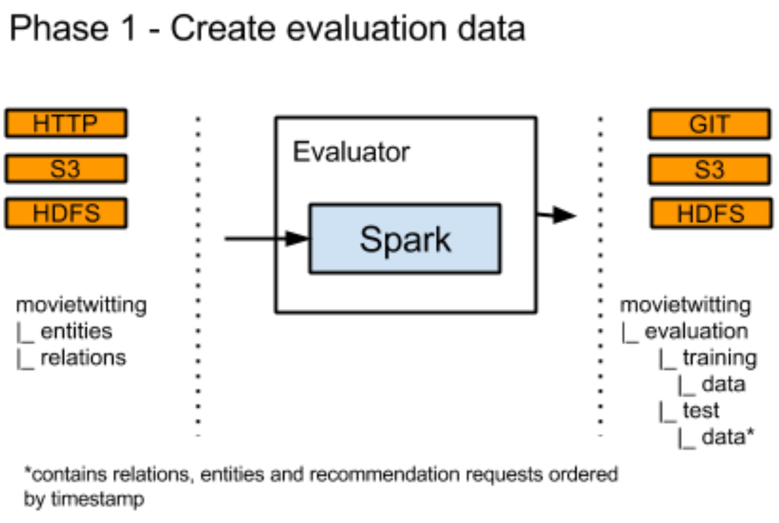
The splitter splits the data using some criteria (e.g., random split 75% train, 25% test set).

Firstly, the dataset is split in Train and not-Train. The not-train part is processed to create some recommendation requests (e.g., the user1 wants a recommendation at the ts 1929293495), and some additional data to show during the recommendation in real time (e.g., a new object is available, or the user1 has consumed item1 at ts 195959760). Every recommendation request is copied as is in the GroundTruth folder; then, every recommendation request is partially hidden and puts in the Test folder (e.g. in the GroundTruth folder the splitter puts “rating.explicit 7 1362869072 {"rating":9} {"subject":"user:5","object":"movie:1707386","recid": 2}”, while in the test folder the splitter puts “recommendation 1 1362869072 {} {"subject":"user:5"}”).

Every file in every folder must follow the Idomaar format. The id of each GroundTruth line is called “recId”. Every test entry must have a property called “recId” to allow the evaluator to map the recommendations to the GroundTruth entries.

The only file with a different format from the Idomaar one is recommendations.idop. Each line of its line must be the same as one recommendation request (blue part in the example below) from the Test folder with an additional column appended with the proposed recommendation (green part in the example below).

e.g., recommendation 2 1362946823 {} {"subject":"user:6"} [{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]



The splitting is performed by a *script* executed by the **evaluator**. The strategy can be customized by implementing an ad-hoc splitting script.

Currently, we implemented three splitting strategies as Apache Spark scripts: two non-time aware strategies (namely, random and take-n-out), and one time-aware strategy (namely, temporal splitting).

4.2.1. Random splitting

Random splitting takes a value as input: it represents the probability an entry is in the train set as a number in the interval (0,1]. Only relation lines are split, while the entities are all in the training set.

Every element of the test set is a recommendation request: the splitter does not consider the timestamp of the request, thus it merges different requests from the same subject in different timestamps as a unique request. The evaluation consider the whole recommendation versus all the objects consumed by each subject in the test set.

```
e.g.,
== RAW ==
rating.explicit     1      1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     2      1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910"}
rating.explicit     3      1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4      1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5      1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
rating.explicit     6      1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414"}
rating.explicit     7      1362869072    {"rating":9}
{"subject":"user:5","object":"movie:1707386"}
```

```
rating.explicit     8      1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389"}
rating.explicit     9      1362946823    {"rating":9}
{"subject":"user:6","object":"movie:0253474"}

#rating.explicit    8      1362093598    {"rating":9} {"subject":{"type":"user",
"id":"6"},"object":"movie:0093389"}

== TRAIN ==
rating.explicit     1      1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     3      1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4      1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5      1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
rating.explicit     9      1362946823    {"rating":9}
{"subject":"user:6","object":"movie:0253474"}

== TEST ==
recommendation      1      -1     {}     {"subject":"user:2"}
recommendation      2      -1     {}     {"subject":"user:5"}
recommendation      3      -1     {}     {"subject":"user:6"}

== RECOMMENDATION ==
recommendation      1      -1     {}     {"subject":"user:2"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2}]
recommendation      2      -1     {}     {"subject":"user:5"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2}]
recommendation      3      -1     {}     {"subject":"user:6"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2}]

== GROUNDTRUTH ==
rating.explicit     2      1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910","recid": 1}
rating.explicit     6      1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414","recid": 2}
rating.explicit     7      1362869072    {"rating":9}
{"subject":"user:5","object":"movie:1707386","recid": 2}
rating.explicit     8      1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389","recid": 3}
```

4.2.2. Take-n-out splitting

Take-n-out considers the last ‘n’ relations of a subject. It takes n as input parameter. It puts in the training set the first k-n relations, in the test set, one request for each subject, ignoring the timestamp. The evaluation considers the whole recommendation versus the last n objects consumed by each subject. Subjects with less than n+1 relations are discarded.

```
e.g.,
 == RAW ==
rating.explicit     1      1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     2      1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910"}
rating.explicit     3      1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4      1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5      1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
```

```
rating.explicit     6      1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414"}
rating.explicit     7      1362869072    {"rating":9}
{"subject":"user:5","object":"movie:1707386"}
rating.explicit     8      1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389"}
rating.explicit     9      1362946823    {"rating":9}
{"subject":"user:6","object":"movie:0253474"}

== TRAIN ==
rating.explicit     1      1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     3      1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4      1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5      1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
rating.explicit     9      1362946823    {"rating":9}
{"subject":"user:6","object":"movie:0253474"}

== TEST ==
recommendation      1      -1     {}     {"subject":"user:2"}
recommendation      2      -1     {}     {"subject":"user:5"}
recommendation      3      -1     {}     {"subject":"user:6"}

== RECOMMENDATION ==
recommendation      1      -1     {}     {"subject":"user:2"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]
recommendation      2      -1     {}     {"subject":"user:5"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]
recommendation      3      -1     {}     {"subject":"user:6"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]

== GROUNDTRUTH ==
rating.explicit     2      1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910","recid": 1}
rating.explicit     6      1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414","recid": 2}
rating.explicit     8      1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389","recid": 3}
```

4.2.3. Temporal splitting

Temporal splitting splits the dataset considering a specific timestamp: each relation and entity with a timestamp lower than the input one is put in the training set.

Each relation/entity with a higher timestamp is put in the GroundTruth one. Some of these entries are hidden and put in the test folder as recommendation requests. The data.ido file under the Test folder is strictly ordered by timestamp: this way the orchestrator can stream request in a consistent way.

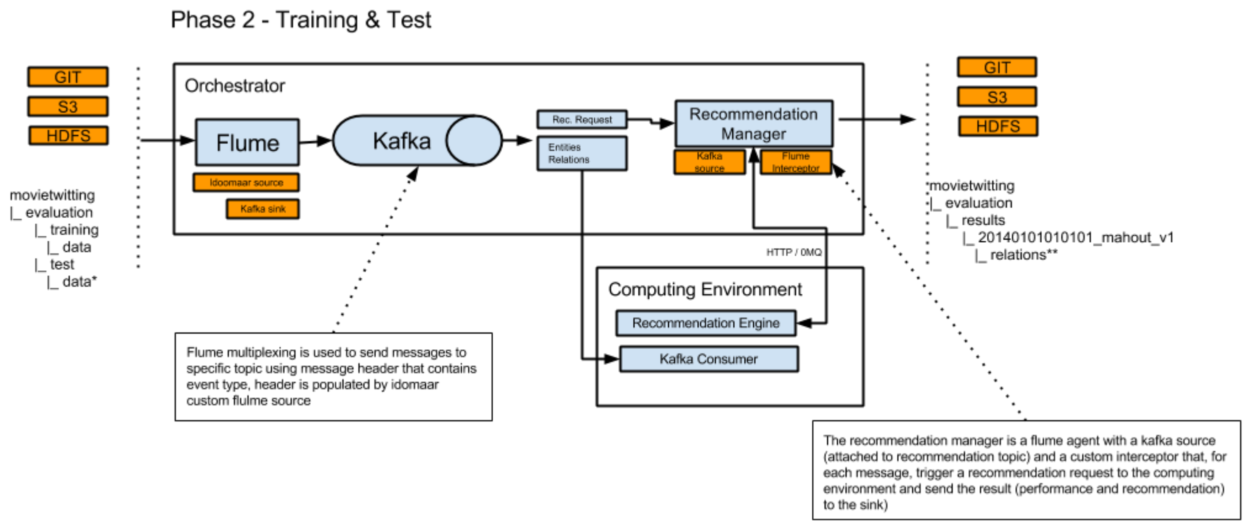
The evaluation is a bit different: every recommendation is linked to a specific request in a specific timestamp. Thus, the recommender must create one row for each request.

```
e.g.,
== RAW ==
rating.explicit     1      1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     2      1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910"}
```

```
rating.explicit     3     1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4     1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5     1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
rating.explicit     6     1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414"}
rating.explicit     7     1362869072    {"rating":9}
{"subject":"user:5","object":"movie:1707386"}
rating.explicit     8     1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389"}
rating.explicit     9     1362946823    {"rating":9}
{"subject":"user:6","object":"movie:0253474"}

== TRAIN ==
rating.explicit     1     1363245118    {"rating":9}
{"subject":"user:1","object":"movie:0120735"}
rating.explicit     2     1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910"}
rating.explicit     3     1363566189    {"rating":8}
{"subject":"user:3","object":"movie:1924396"}
rating.explicit     4     1363557326    {"rating":8}
{"subject":"user:4","object":"movie:0887912"}
rating.explicit     5     1362869000    {"rating":7}
{"subject":"user:5","object":"movie:1182350"}
rating.explicit     6     1362869039    {"rating":9}
{"subject":"user:5","object":"movie:1230414"}

== TEST ==
recommendation      1     1362869072    {}     {"subject":"user:5"}
rating.explicit     8     1362093598    {"rating":9}
{"subject":"user:6","object":"movie:0093389"}
recommendation      2     1362946823    {}     {"subject":"user:6"}

== RECOMMENDATION ==
recommendation      1     1362869072    {}     {"subject":"user:5"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]
recommendation      2     1362946823    {}     {"subject":"user:6"}
[{"object":"movie:0120735", "rank":1},{"object":"movie:0555735", "rank":2},...]


== GROUNDTRUTH ==
rating.explicit     2     1362901837    {"rating":10}
{"subject":"user:2","object":"movie:2592910","recid": 1}
rating.explicit     7     1362869072    {"rating":9}
{"subject":"user:5","object":"movie:1707386","recid": 2}
```

4.3. Phase 2 - training and testing

The training and testing phase is composed by two sequential tasks.


Phase 2 - Training & Test
GIT
S3
HDFS
movietwitting
|_ evaluation
|_ training
|_ data
|_ test
|_ data*
Orchestrator
Flume
Idoomaar source
Kafka sink
Kafka
Rec. Request
Entities Relations
Recommendation Manager
Kafka source
Flume Interceptor
HTTP / 0MQ
Computing Environment
Recommendation Engine
Kafka Consumer
Flume multiplexing is used to send messages to specific topic using message header that contains event type, header is populated by idomaar custom flulme source
The recommendation manager is a flume agent with a kafka source (attached to recommendation topic) and a custom interceptor that, for each message, trigger a recommendation request to the computing environment and send the result (performance and recommendation) to the sink)
GIT
S3
HDFS
movietwitting
|_ evaluation
|_ results
|_ 20140101010101_mahout_v1
|_ relations**

Initially, once the computing environment has booted, it is **bootstrapped** by using the *model training set* stored in the Train folder.

Once bootstrapping is completed, the orchestrator initiates a second phase where the computing environment is required to **serve recommendations** at real-time**.** The computing environment is flooded by the messages contained in the Test folder, which consists of both new training data (recommendation training data) and recommendation requests.

The second phase terminates when the computing environment has processed all messages.

The following schema specifies the communication protocol, from the boot of the computing environment to the completion of the test phase.

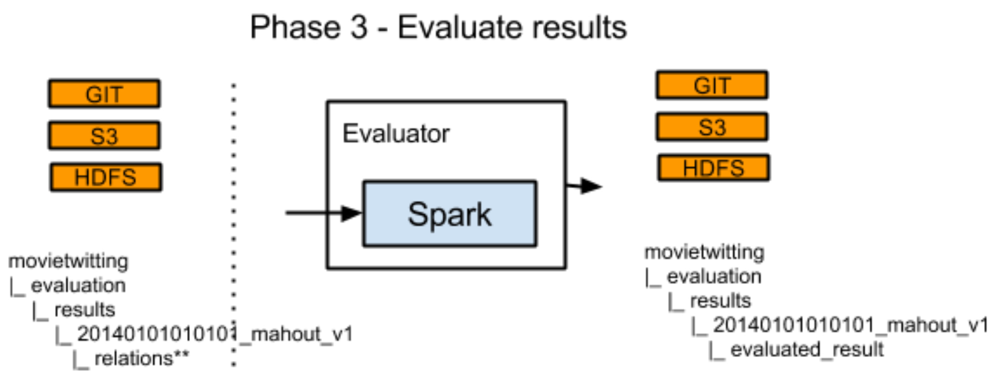
The output of the computing environment adheres the output format mentioned in D2.2 "First Reference Framework Release plus Requirements Specifications", Section 5.2.

The computing environment has to exposes suitable interfaces in order to receive messages from the orchestrator and serve recommendations. The technologies used in the communication between the orchestrator and the computing environment is described in Section 4.5.

The output of the computing environment is stored in an extended version of the Idomaar format, composed by an additional column.

4.4. Phase 3 - evaluation of recommendation results

The last phase concerns the evaluation of the results returned by the computing environment to serve the recommendation requests. This task is implemented as a script within the evaluator. Similarly to the splitting phase, the metrics to compute can be customized by implementing an ad-hoc splitting script.



Currently, we implemented the CTR (click-through-rate) as quality metric, as described in the following. The CTR is the ratio of items that have been recommended at a certain point in time and selected (i.e., *clicked*) by the user. Because it is not possible, in an offline evaluation, to have the direct user feedback (i.e., whether a user clicked on the recommended item), the CTR is evaluated indirectly using a proxy metric the ratio of items that have been recommended at a certain point in time and selected (i.e., clicked) by the user within a certain time in the future. For instance, in NewsReel such time is set to 5 minutes from the recommendation request.

CTR requires that groundtruth is not calculated during the splitting phase because it depends both on items that are really clicked by the user in the test set and the hits of recommender recommendations at each request. In fact, for any recommendation request, the groundtruth is composed by each item clicked by the user in a time-window of 5 minutes (starting from the recommendation time): however, each item already successfully recommended by any previous request is not counted again if it is recommended, otherwise it would compromise the estimate, thus it has to be filtered out. This means that the same item probably is in more than one time-window, but it counts as a “hit” only once: after it is recommended, it must be banned from the groundtruth. For this reason, we decided to compute the groundtruth dynamically during the evaluation process to reduce the overhead.

4.5. Stream management: technologies and interfaces

The orchestrator is the component responsible for driving the computing environment. That is, it transmits training and test data from their original location in the data container to the computing environment, requests recommendations (via the recommendation manager) and makes recommendation results available for evaluation.

4.5.1. Ease of use

Orchestrator application logic, in the form of Python scripts, resides on a virtual machine automatically built by Vagrant. Thus, the orchestrator depends only on Vagrant itself (and a virtual machine provider), which, in turn, are available on all commonly used platforms. Hence the orchestrator is basically platform independent and easily installed on any host. A simple shell script with a number of command line parameters starts the orchestrator.

4.5.2. Orchestrator communication

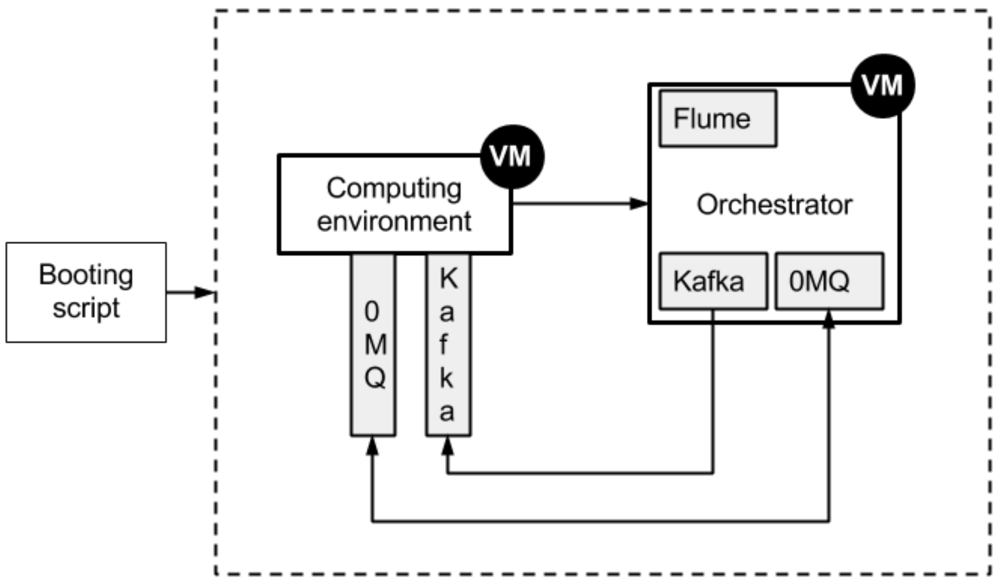
The orchestrator and the computing environment exchange short control messages via a ZeroMQ based protocol, as described in D2.2 “First Reference Framework Release and Evaluation Report”, Section 4.4.2. This serves the purpose well: it is lightweight, supports most modern environments and is straightforward to use.

For transferring large amounts of data, the D2.2 “First Reference Framework Release and Evaluation Report” report mentioned that neither FileMQ nor shared folders seemed to be an optimal choice, as the former supports only a few platforms and does not support seeking while the latter makes streaming impossible.

The orchestrator now uses a setup with two open source applications, Kafka and Flume, that together form a first-rate, versatile and performant solution to stream event data. In the following we briefly describe these two tools. All orchestrator technologies are executed within a virtual machine, as shown in the following picture. Optionally, 0MQ can be replaced with a http-based protocol.

4.5.3. Kafka

Kafka is a messaging system specifically designed to transmit logs. “Log” is meant in a general sense; apart from the human-readable text lines produced by applications for debugging purposes it includes any linear sequence of event messages. Examples are database changelogs, user activity streams or application metrics. With this principal use case in mind, Kafka does away with the complex APIs and usage patterns of general-purpose messaging systems (like ActiveMQ or RabbitMQ). At the expense of lacking strict only-once delivery guarantees, it offers superior performance. After its inception at LinkedIn, it is now an Apache Software Foundation project -- a hallmark of quality. Although the latest release version is numbered 0.8.2 (that is, the API is not yet considered stable), it is widely and successfully used in production setups at LinkedIn, Twitter, Spotify and other high-profile data-oriented companies.



Training and test data for recommender systems are actually streams of user events, so Kafka fully covers our use case. Messages in Kafka are durable, so both online and offline data processing is possible. Online processing is near real time, with latency on the magnitude of a second. Kafka scales; from one stream of log on a single host you can easily go to a few thousands of streams. (Cross-datacenter replication, although not without problems, is also being done for some deployments.)

Vagrant provisioning automatically installs Kafka on the orchestrator virtual machine. The orchestrator includes functionality to configure Kafka when needed (e.g. create multiple streams for parallel recommendation requests).

4.5.4. Flume

Another member of the Apache projects, Flume is a tool to collect and move large amounts of log data from many different sources to data stores. For Idomaar, its main strength lies in its flexibility and configurability: Flume's plugin-based architecture makes its possible to read training and test data from a variety of sources and write recommendation results to different sinks. There are a couple of built-in sources and sinks (e.g., file-based, HTTP-based, HDFS) and it is straightforward to implement and use new ones if the need arises. Notably, there is a Flume source (and a Flume sink) that reads data from Kafka (and writes data to Kafka) so Flume can serve as an integration layer between Kafka and a range of data sources.

Flume is automatically installed on the orchestrator virtual machine by Vagrant provisioning (using packages from Cloudera). At runtime, the orchestrator is able to configure and bring up Flume by generating Flume property files and starting Flume agents. For instance, the orchestrator can instruct Flume to write recommendation results to plain files or HDFS. For demo purposes, Idomaar implements a new data source in Flume to pull training and test data from GitHub.

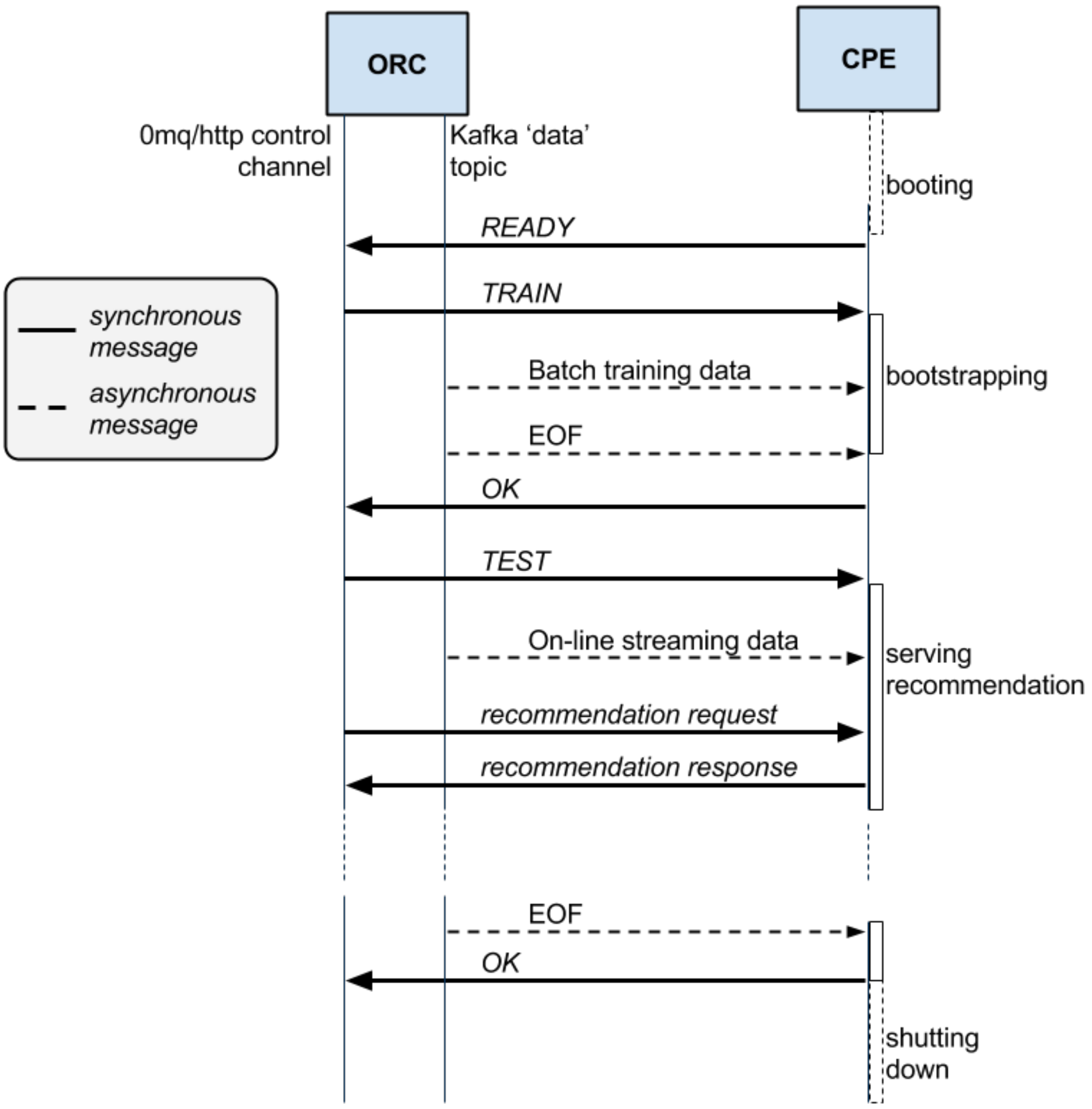
4.5.5. Computing environment interfaces

The computing environment has to expose the following interfaces in order to receive messages from the orchestrator:

- synchronous communication messages are sent to the computing environment to a 0MQ interface
- asynchronous events are sent via the Kafka queue
- synchronous recommendation requests are sent to the computing environment or via 0MQ interface or via HTTP rest interface

Communication messages

Orchestrator control the execution of the recommendation environment via 0MQ messages, the workflow and the list of messages that the computing environment has to implement are described in the following diagram.



Asynchronous events

The events, both the training and on-line streaming data, are sent to the recommendation environment using the Kafka topic “data”, the format of the data is the one specified in D2.2 "First Reference Framework Release plus Specifications", Section 5.1 "Data Model" and each streaming is concluded with a message of type “EOF”.

Synchronous recommendation requests

Recommendation requests are managed by the orchestrator using a custom Flume interceptor and currently the following interceptor has been developed:

- HTTP rest
- 0MQ

the selection of the protocol is defined at orchestrator config level and new implementation could be easily added with new flume plugins.

0MQ interface use a simple request-answer multipart message WITH the following format:

- Request: [RECOMMEND, RECLEN, SUBJECT] where RECOMMEND is the type of message, RECLEN is an integer representing the recommendation length and SUBJECT is the representation of the subject in the Idomaar standard format (e.g. {"subject":"user:27"})
- Response: has to be in the standard Idomaar format.

The HTTP interface is implemented as a Flume interceptor, which is an HTTP client connecting to the server endpoint provided by the computing environment. When a recommendation request arrives in the Flume pipeline, the interceptor implementation compiles a URL that contains the type of the message (this is “recommendation”), its properties (this includes recommendation length) and the subject representation as URL query parameters, and executes an HTTP POST request using this URL.

5. Reference framework Development and Dissemination

We maintain the web site http://rf.crowdrec.eu/ to disseminate the reference framework releases. The web site's primary aim is to describe the framework's objectives, create a community of users and explain how Idomaar can be used and integrated.

The web site refers to the resources available on the collaborative platform github, where we created the "CrowdRec" (https://github.com/crowdrec/). With respect to the first release (D2.2 "First Reference Framework Release plus Requirements Specifications", Section 8), we modified the structure of the organization by reducing it to two repositories, namely **Idomaar** and **datasets**. Practically, we moved all resources related to the computing environment, the algorithms, the evaluator, and, of course, the orchestrator to the Idomaar repository. This change aims at simplifying the management of releases and versioning.

The management of the development of the reference framework has been designed to be simple and transparent in order to facilitate the collaboration between partners. It is mainly supported by Github issues assigned to specific users and related to defined milestones.

A great deal of effort has been invested in disseminating Idomaar towards communities that will use it, and also provide feedback for further development. Two primary channels were chosen that were considered to be strategically most effective, in order to focus the dissemination effort. These are: the NewsReel challenge and the Idomaar workshop. The former, already described in Section 3, has promoted the usage of Idomaar to evaluate the participants to the challenge.

The Idomaar workshop was held on 20 March in Milan, organized by Moviri, in collaboration with Politecnico di Milano, and supported by the main partners involved in the Idomaar's development: Gravity and Technische Universitat Berlin. The timing of the workshop was planned to coordinate with the end of the third academic semester, in order to ensure the greatest possible student participation.

The main goals of the workshop were:

- disseminating Idomaar, describing its components,
- explaining the distributed and scalable technologies used to (i) provision virtual machines (e.g., Vagrant and Puppet) and (ii) manage streams of data (e.g., Apache Flume, Apache Kafka, etc.),
- proving the capabilities of Idomaar (e.g., its application to the Newsreel scenario),
- testing Idomaar with different architectures to validate the second official release,
- giving people the opportunity to practically play with the framework and the related technologies.

The workshop was a full-day event, open both to CrowdRec's members and to external participants. In particular, it welcomed members of the university community (Politecnico di Milano students and researchers). The agenda consisted of the following parts:

- **Part I - Introduction to CrowdRec and Idomaar**
    - Data stream processing and recommender system evaluation, in particular using NewsReel as practical example
    - Idomaar's architecture
    - Apache Spark: a trivial recommendation algorithm to be tested within Idomaar
- **Part II - Technologies**
    - Automatic provision: Vagrant, Puppet
    - Stream management and communication: Apache Flume, Apache Kafka
- **Part III - Practical examples**
    - Hands-on session consisting of a guided tutorial, alternating a step-by-step tutorial with some mini tasks to solve.
    - Conclusion with a practical example about the porting of an existing provisioning configuration for local execution to a cloud service (e.g., AWS)
- **Appendix - Internal workshop (only CrowdRec)**
    - **This** part is limited only to CrowdRec members and it is the opportunity to discuss what presented during the workshop, to clarify doubts about how Idomaar works and how a recommendation algorithm can be tested.

A number of master students, some researchers and some Crowdrec partners (7 people from Moviri, TU Berlin, TU Delft, and Gravity R&D) have attended the workshop. The picture at the left was taken during a presentation at the workshop, and is one of the pictures that was shared on Twitter while the workshop was running. We counted about 30 students during the morning and more than 60 students during the afternoon. During the afternoon (i.e., 'Part III – Practical examples'), the workshop became interactive: some students and some researchers attempted use the technologies to quickly solve some trivial tasks (e.g., trying to carry out some data analysis with Apache Spark).

The workshop was able to achieve its goals. In particular, it was satisfying that students appeared to be interested in the Crowdrec themes and in the technologies showed up during the workshop.

6. Reference framework evaluation

In the D2.2 report “First Reference Framework Release and Evaluation Report” (Section 9), we described how we evaluated the reference framework (that is, Idomaar) with respect to requirements set forth in D2.2 Section 3.1. In the following, we take a look at how the evolution of the reference framework contributes to the fulfillment of these criteria.

6.1. Fulfilling the requirements

6.1.1. Consistency and reproducibility

Partly as a response to the platform-dependence issues detected during an Idomaar hands-on session (mentioned in Section 9.2 of the D2.2 report), the central component of the reference framework, the orchestrator, now fully resides on one virtual machine (provisioned by Vagrant). It can be built on any platform supporting Vagrant (this includes Mac OS X, modern Windows versions and common Linuxes) and should behave identically. It can be torn down fully and be rebuilt from scratch by issuing a single command. Thus, it is perfectly reproducible, which makes testing consistency straightforward.

Note that training and test data as well as the evaluator are external to the orchestrator, so on top of orchestrator consistency, their consistency is also required. For demo purposes, Idomaar uses a GitHub repository as a data source, which, given a revision, is consistent by the very essence of Git. For future data containers to be used in production setups (Amazon S3 is currently being considered) we will also have to ensure consistency.

Generally speaking, Idomaar cannot ensure the consistency of any possible computing environment, since there is no restriction on computing environments apart from implementing the necessary communication protocols and respecting resource limitations. They are free to compute referentially opaque functions, e.g., they can look at wallclock time. However, Idomaar includes a demo computing environment, based on a recommendation algorithm from the Apache Mahout machine learning library and built by Vagrant. For this computing environment, the algorithm’s referential transparency and the reproducibility of Vagrant provisioning do yield consistency.

6.1.2. Independence of architecture

For Idomaar to gain any significant traction, it must be easy to implement computing environments on virtually any platform. As discussed in the D2.2 report, the JSON data API and ZeroMQ are lightweight and popular choices for application interfaces, and this continues to be valid. The planned inclusion of further out of the box computing environments will provide additional evidence of the APIs being adequate.

In this new release, there is a novel requirement that Idomaar must be able to handle large datasets (requirement 9 in Section 2 of this document). Furthermore, the requirement about the streaming ability (requirement 8) has been promoted to have higher priority. To address these requirements, Kafka has been introduced as a component in Idomaar (see Section 4.5.3). This, in turn, requires computing environments to be able to connect to Kafka, which makes implementing computing environments more complicated. Fortunately, Kafka consumers (i.e, libraries to read data from Kafka) abound. There is at least one for each of the major languages (C++, Java, Python, among others) and there is also a framework to expose Kafka services via HTTP REST (the latter is not integrated into Idomaar, though). The demo computing environment shows how to use a Java-based Kafka consumer.

6.1.3. Standardization

One of the goals the reference framework strives to achieve is to provide a standard environment for carrying out recommendation system evaluation. On the one hand, this enables recommender system testing under conditions which are very close to production setting, without the need to deploy these algorithms and artifacts to actual production systems. On the other hand, it provides a platform to compare recommender systems. In particular, Idomaar will be used in the NewsReel challenge to assess news recommender systems.

Integration of more recommendation frameworks (Lenskit, or MyMediaLite) besides the current Mahout example will be a crucial step towards standardization, since, once integrated with Idomaar, all the algorithms developed for the particular framework will automatically be runnable within Idomaar.

The evaluator component (performing dataset splitting and the recommendation evaluation) is now a separate scriptable part in Idomaar, instead of the earlier close integration with RiVal (see D2.2 Section 7.2). This does not harm the standardization of Idomaar. In order to introduce a new way of measurement, formerly it had to be implemented in RiVal. Now only the evaluator script has to be changed, still leaving the rest of the framework unaffected.

Another, equally relevant, facet of standardization is that Idomaar itself is built from fairly standard components. The new components in the orchestrator and the evaluator (Kafka, Flume, Spark) are all open source, widely used Apache Software Foundation projects. Apache requirements dictate that projects should have an active and diverse community and adequate support infrastructure. This increases the accessibility of the project for developers (e.g. if something goes wrong in Idomaar, it is easier to get help) and lowers the initial barrier new contributors might face.

6.2. Aspects of the 3D model

In the earlier report D2.1 “First Iteration Requirements plus Evaluation Specifications”, Section 6, CrowdRec evaluation follows the CrowdRec 3D evaluation model. Let us now take a look at how the reference framework itself aligns with the factors of the 3D evaluation model.

6.2.1. Technical constraints

With the adoption of Kafka and Flume, Idomaar can handle streams. These tools give Idomaar its unique capability of doing evaluations on streams, as opposed to currently used benchmarks, which typically operate on a fixed test set given in advance. Evaluation on streams is a feature also required by the NewsReel use case.

In terms of resource requirements, Idomaar happily runs on any current commodity hardware (e.g., on an Intel Core i5-4200 with 8GB of RAM) without delays that would hinder development. The use of Vagrant with a conventional virtual machine provider (typically, Oracle's VirtualBox) might introduce some overhead. If this becomes an issue, for Linux platforms the lightweight Docker container is available as a Vagrant VM provider without incurring the overhead of full virtualization.

6.2.2. Business requirements

There are several aspects of Idomaar that have implications for business aspects of recommender evaluation. The conclusions of D2.2, Section 9.3 about data privacy continue to hold. Orchestrator interfaces (ZeroMQ and Kafka) are available via TCP/IP, so a computing environment can reside anywhere on a suitable network. This enables testing a "black box" computing environment with a shared orchestrator and evaluator, for instance to evaluate the services of different vendors who would be unwilling to provide any insight into the mechanics of their systems. Alternatively, the whole of Idomaar can be installed on proprietary infrastructure. With the orchestrator integrated on a virtual machine, this is even easier in the new release.

An interesting possible use case for Idomaar's streaming capabilities is to do online evaluation of production recommendation algorithms *in vivo*. With Kafka and Flume it is feasible to receive a real user event stream. Flume aims to integrate with a multitude of data sources, so even in-house data pipelines can be integrated. The scalability of Kafka and Flume means that in order to keep up with a high rate data stream, all one has to do is create enough instances. Note that this functionality is not readily available in the orchestrator, but its architecture allows extensions in this direction.

6.2.3. User requirements

The potential user base for Idomaar are developers of recommendation algorithms and recommender systems. Given the abundance, variety and accessibility of free software these days, with new solutions emerging literally every day, developers have little time and patience to assess applications. Therefore, any initial trial must go very smoothly.

It is now straightforward to run Idomaar. In the GitHub repository, there is a demo script to launch Idomaar with a simple computing environment, so all that is needed to see the framework in action is to install Vagrant, check out Idomaar code from GitHub, and run a script (available for Windows and Unixes).

Besides usability issues, another common reason for instant rejection in a software evaluation session is instability (unexpected crashes). Bluntly put, it should just work. As outlined in D1.2 Project Quality Assurance Manual Section 4.6, the common way of ensuring software stability while enabling new developments is branching via version control. Idomaar already has a stable branch for its version 1.0.0 on GitHub. Some branching models suggest that the main branch (called *master* in Git) should always contain a production-ready version, which is not currently the case for Idomaar. Some way of automated testing (Idomaar quality assurance rejected the idea of developer test cases) would yield an increased level of stability.

6.3. Real world application

Apart from the general considerations above, we now provide a practical point of view on evaluation. The NewsReel challenge, as described in detail in Section 3 will use the reference framework for its Task 2. The use of the reference framework as a NewsReel execution and evaluation framework will yield important information about its applicability.

- Ease of use: Computing environments have to implement given communication interfaces in order to receive control messages, data, and recommendation requests from Idomaar. How easy is it to implement these interfaces properly? Should it be made easier? Although the demo computing environment does provide a Java implementation, using other tools and languages, as well as the sheer number of implementations, might bring new usability insights.
- Performance: The expectation is that reference framework performance should never become the bottleneck for a training-test-evaluation process. It terms of speed and memory consumption, it is usually much easier to send control messages and stream data, than to provide recommendations on a reasonably large dataset with any reasonable accuracy. However, unusual computing environments (e.g., exhibiting timeout patterns) might induce unexpected conditions in the reference framework itself that can lead to performance deterioration.

The NewsReel challenge is different from previous benchmarks (e.g., earlier KDD cups or RecSys challenges) in that instead of evaluating in an artificial static setting, participants have to predict users' clicks on recommended news articles in simulated real-time (Task 2). The competing algorithms are evaluated in terms of recommendation quality as well as other metrics (most importantly, response time).

7. Conclusions and outlook

This deliverable has presented the second release of the reference framework – Idomaar – together with the evaluation report.

The second release of Idomaar has taken into account both the outlook described at the end of the deliverable D2.1 "First Reference Framework Release and Evaluation Report" (Section 10) and the requirements collected in D2.3 "Second Iteration Requirements".

Idomaar, besides allowing the evaluation of recommendation algorithms in an architecture-independent environment that grants consistent and reproducible results, currently integrates several open-source, state-of-the-art technologies that enable the processing of streams of data in a distributed and scalable model. The effectiveness of Idomaar is currently under examination in the settings of the NewsReel challenge.

We expect to gather comments and feedback from the NewsReel challenge that will drive the next iterations of Idomaar. In addition, next releases will include supplementary computing environments compliant with the revised Idomaar architecture, allowing practitioners to rely on standard algorithms already available within the framework both to quickly "try out" the framework and to compare their custom solutions. A number of algorithms released in WP3 "Stream Recommendation Algorithms" and WP4 "Crowd Engagement Algorithms" will be included in next Idomaar releases to permit their testing and comparison in order to select the candidates for the deployment in the real-world environments depicted in WP5 "Large-scale real-world application and deployment".

Finally, the evaluator integrated in the first release has now been reviewed in the second released of Idomaar. The new implementation is based on data streaming and processing technologies (such as Apache Spark). This allows the easy integration of custom evaluation strategies and metrics, which simply require a user to write a script that interfaces with the Idomaar's orchestrator streaming components (based on Apache Flume and Apache Kafka), whose connectors are available in most popular programming languages.