COMPARATIVE EVALUATION OF RECOMMENDER SYSTEMS FOR DIGITAL MEDIA

Domonkos Tikk[1], Roberto Turrin[2], Martha Larson[3], Dávid Zibriczky[1], Davide Malagoli[2], Alan Said[3], Andreas Lommatzsch[4], Viktor Gál[1], Sándor Székely[1]

[1] Gravity R&D Zrt, Hungary
[2] Moviri, Italy
[3] Delft University of Technology, The Netherlands
[4] Technische Universität Berlin, Germany

ABSTRACT

TV operators and content providers use recommender systems to connect consumers directly with content that fits their needs, their different devices, and the context in which the content is being consumed. Choosing the right recommender algorithms is critical, and becomes more difficult as content offerings continue to radically expand. Because different algorithms respond differently depending on the use-case, including the content and the consumer base, theoretical estimates of performance are not sufficient. Rather, evaluation must be carried out in a realistic environment.

The Reference Framework described here is an evaluation platform that enables TV operators to compare impartially not just the qualitative aspects of recommendation algorithms, but also non-functional requirements of complete recommendation solutions. The Reference Framework is being created by the CrowdRec project which includes the most innovative recommendation system vendors and university researchers in the specific fields of recommendation systems and their evaluation. It provides batch-based evaluation modes and looks forward to supporting stream-based modes in the future. It is also able to encapsulate open source recommender and evaluation frameworks, making it suitable for a wide scope of evaluation needs.

INTRODUCTION

Content consumption trends have changed significantly in the past ten years with the digital revolution and the rise of the internet. The amount of content that is available for consumers to choose from has grown unprecedentedly. Additionally, a range of new devices including tablets and smartphones connected to 4G mobile networks make it possible for people to consume content in a wide range of different settings *where* they want and *when* they want. Services like Netflix and YouTube have arisen that offer a wide range of different choices. An audience that is not deeply engaged in what they are watching or listening to can easily move on to find alternative content with a greater attractive force, and better suited to when and where they are watching. Today's consumers are satisfied only with the perfect match between their present need, and the content that they are consuming.

Due to fierce competition in recent years, TV operators and content providers have expanded their media offerings by adding: more live channels, HD content, on-demand content catalogues, and premium functions like personal video recording, catch-up, pause, fast-forwarding, etc. Important trends for content brands include multi-device and multi-channel content, that is available at anytime, anywhere.[1,2] However, in order to ensure that consumers are able to find the perfect match in this broad offering, they must also provide their customers with assistive tools that allow them to navigate within the available selections. The tools must support users in finding content that is both suited for their situation at each moment, and also matches their interests and needs. Unless such services can be provided, content consumers risk drowning in the content glut, and operators' investment in content extension and services will have been in vain.

In order to address this challenge, service operators and content providers turn to recommender systems. Recommender systems are tools that filter large quantities of information in a way that identifies a personalized selection of content that is suited for a particular user. Today's recommender systems are able to select content that fits not only a user, but also the specific situation, e.g., daytime vs. evening consumption, consumption alone vs. with the family, or consumption on the go vs. while relaxing. Recommender systems are also able to process a stream of new information, such as incoming contents, or new user responses – integrating such real-time information into the recommendations on-the-fly improves the quality and the responsiveness of the system. Unfortunately, the factors that define a good recommendation, and set the perfect match apart from other possible matches, are neither consistent nor stable across use scenarios. Rather, recommendations that lead to purchases, clicks and a feeling of satisfaction on the part of consumers depend on a wide range of factors. As a result, different situations require different recommender systems.

Choosing the right recommender system can be a daunting task for a TV operator or content provider. In each case, the content and the needs of the users can vary; additionally there is a large number of alternative technologies available. Many of these have only been evaluated internally making objective comparisons across techniques and frameworks difficult, if not impossible. Further, properties of the data, such as sparsity, or the rate at which new items enter a stream can also impact the suitability of a recommender system for a particular use-case. Selecting and deploying an unsuitable recommender system can be a costly mistake. In order to select the right technology, TV operators and content providers must evaluate existing solutions within the specific conditions of their own use-case.

Currently, the evaluation of recommender systems is both complex and laborious. In order to run qualitative tests the operator has to integrate the recommender system into its IT content delivery platform—requiring significant effort and cost—or rely on others' assessment and experience, which often suffers from bias towards a context, dataset or recommendation solution. For this reason, an independent and highly customizable evaluation framework that can be used by solution seekers and vendors in different contexts is sorely needed. This is one of the major aims of the CrowdRec project consortium, a group of the most innovative recommender system vendors and academic researchers currently active in the area.

---

[1] http://socialmediatoday.com/nealschaffer/1252151/what-your-marketing-efforts-lack-leveraging-latest-content-consumption-trends

[2] http://www.imediaconnection.com/content/35476.asp

In this paper, we present the CrowdRec Reference Framework that aims to provide a transparent, objective and easy to use evaluation platform for recommendation solutions making comparisons across datasets, recommendations platforms and recommendation domains possible. Next we will discuss how the Reference Framework enables evaluation; then we describe its components and the workflow, using some typical use-cases.

CROWDREC REFERENCE FRAMEWORK AS EVALUATION PLATFORM

The CrowdRec project aims to lighten the burden in recommendation approach selection by providing a Reference Framework that impartially evaluates various aspects of recommender systems. The Reference Framework carries out objective measurement of the key properties of recommender systems: (i) functional requirements that relate to qualitative assessments of recommendations, and (ii) non-functional requirements that are specified by the technical parameters and business goals of the service, such as scalability, reactivity, adaptability and robustness [1]. In the current phase, the CrowdRec Reference Framework makes it possible to run offline tests to assess solutions. It provides communication and data interfaces to decouple the evaluation logic from the recommendation service, enabling both open source and proprietary software to be impartially compared. In the next phase, we target an extension towards online tests.

Next we list some use-cases of the CrowdRec Reference Framework. This list is non-exhaustive and serves to illustrate the potential of the framework:

1. Implementation benchmarking: An "agnostic" appraisal platform able to test both the algorithmic effectiveness and (even more importantly) the implementation efficiency. In this way, if a company is interested in introducing a new recommendation algorithm within its VoD or TV service, it will have the opportunity to validate *a priori* the different implementations of that *same* algorithm available, i.e., provided by a vendor or an open source project, in order to understand which is most appropriate for its current IT architecture (e.g., Java, GNU/Linux).

2. Tendering – external evaluation: A benchmarking tool able to evaluate and compare different recommendation solutions (RSs) for operators and assist business in selecting the one that fits best to its needs. Optionally, it is also capable of comparing the vendor's solution with the operator's internal solution (if one exists) in order to decide whether or not it is worthwhile introducing a new solution (see also Panel A of Figure 5).

3. Algorithm benchmarking – internal evaluation: A method that is capable of comparing recommendation algorithms working on either static data (e.g., in an item-to-item or personalized recommendation scenario) or stream data (e.g., tweets). This use-case enables a company to test multiple recommendation algorithms available on the market with its proprietary or with data received from external sources. If a vendor is interested in tenders, the tool assists to optimize and select algorithms based on business needs (see also Panel B of Figure 5).

4. 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. This approach differs significantly different from theoretical approaches to performance evaluation. It is widely recognized that the same recommendation algorithm described within a research paper might have several implementations with mixed results [2, 3], and

similarly each recommender framework might implement evaluation methods, or evaluation strategies, differently. Currently, the CrowdRec Reference Framework encapsulates several open source recommendation and evaluation frameworks, e.g., LensKit [4], MyMediaLite [5], and Okapi[3] for recommendation, further RiVal [3][4] and test.fm[5] for evaluation. This use-case allows a company to create a context to find a better performing recommendation algorithm, and carry out evaluation its own data set.

The Reference Framework data model (see next Section) equally allows the evaluation of algorithms using all the typical user feedback types, explicit (rating) and implicit (unary user interactions like views or buys), and can carry inputs to the algorithm using external resources such as context (users' social network, location, device, etc.).

REFERENCE FRAMEWORK – ARCHITECTURE

The Reference Framework has four components:
(i) Orchestrator (ORC), (ii) Data Container (DC), (iii) Evaluator (EVL), and (iv) Computing Environment (CPE), as shown in Figure 1.

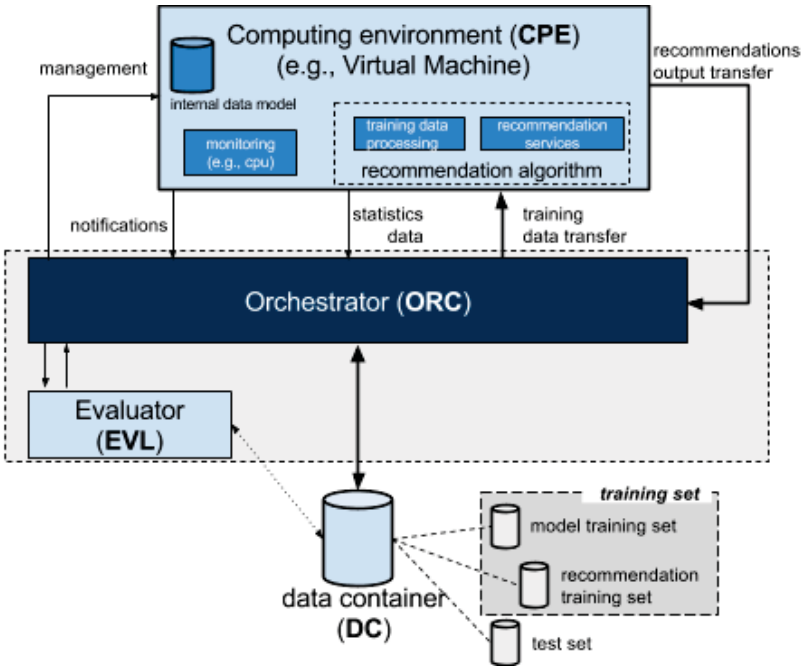

Figure 1 – CrowdRec Reference Framework Architecture

ORCHESTRATOR (ORC)

The Orchestrator is responsible for the overall coordination of the Reference Framework. It interfaces with the DL to access the available data. It cooperates with the EVL in order to prepare the data for the algorithms (e.g., splitting training/test set) and evaluate the output of the algorithms. The ORC supports the standard batch-based – when the model is trained on some historical data and evaluated on a hold-out set – as well as stream-based evaluation. Finally, the ORC is in charge of communicating with the CPE in order to run the required experiments.

Communication

The ORC controls the CPE by sending management messages (e.g. request for recommendations) and providing the training data. The CPE notifies the ORC once the request

[3] http://grafos.ml/index.html#Okapi
[4] http://rival.recommenders.net
[5] https://github.com/grafos-ml/test.fm

has been served and provides two kinds of data: statistics data (about performance metrics, such as CPU usage) and recommendation output (i.e. the result of a recommendation request). Response time can be measured at the ORC as well.

Communications from and to CPEs occur asynchronously by means of messages. Namely the communication between the CPEs and the ORC messages are implemented using the well-known client-server architecture in networking. We have chosen to implement the agreed protocol by using simple TCP sockets.

DATA CONTAINER (DC)

The Data Container contains all datasets available in the Reference Framework. A dataset is composed by two types of data: entities (e.g., users, items, etc.) and relations (e.g., ratings, views, clicks, etc.), as simplified in Figure 2.

An **entity** is a piece of information that can be modeled and recommended. An entity is represented by: a **type** (e.g., movie, user, person, genre), an **identifier**, a set of **properties** (e.g., the title of a movie), and a set of **links to** other **entities** (e.g., the actor of the movie is represented with a link to the 'person' entity "Kate Winslet"). Note that, if we want to recommend movies according to conventional, pre-assigned genres, then the genre of a movie should be an entity, and the movie–genre assignment is modeled by a link. On the other hand, the movie title can be represented by a simple property.

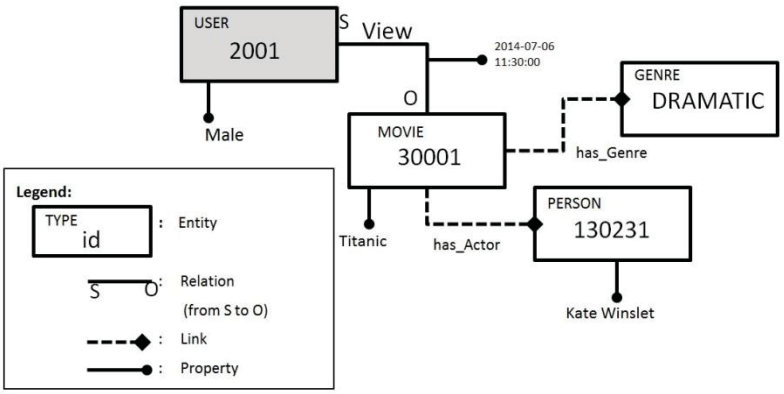

Figure 2 – Data modeling

A **relation** is defined by: a **type** (e.g., view, rating), an **identifier**, a **timestamp**, a set of **properties** (e.g., the score of a rating, the playtime), and a set of **links to entities** (e.g., the user who expressed the rating, the related item, the device, etc.). A relation (solid line) represents a connection that links multiple entities (e.g., the rating given by a *user* to a *movie* at *home* using a *smartphone*), differing from a linked entity (dashed line) that represents a fact (e.g., the *movie* has an *actor*). Multiple entities involved in the relation have been marked with different letters (e.g., 'S' for subject, 'O' for object).

All datasets to be used within the CrowdRec Reference Framework will be mapped to this generic format. Note that this format enables to encode the all typical user feedback types, as the four canonical scenarios depicted in Figure 3: modeling of explicit (user rates a movie) and implicit (user plays a track) feedbacks, context data (device and location) and social relationships (friendship, follower/follower).

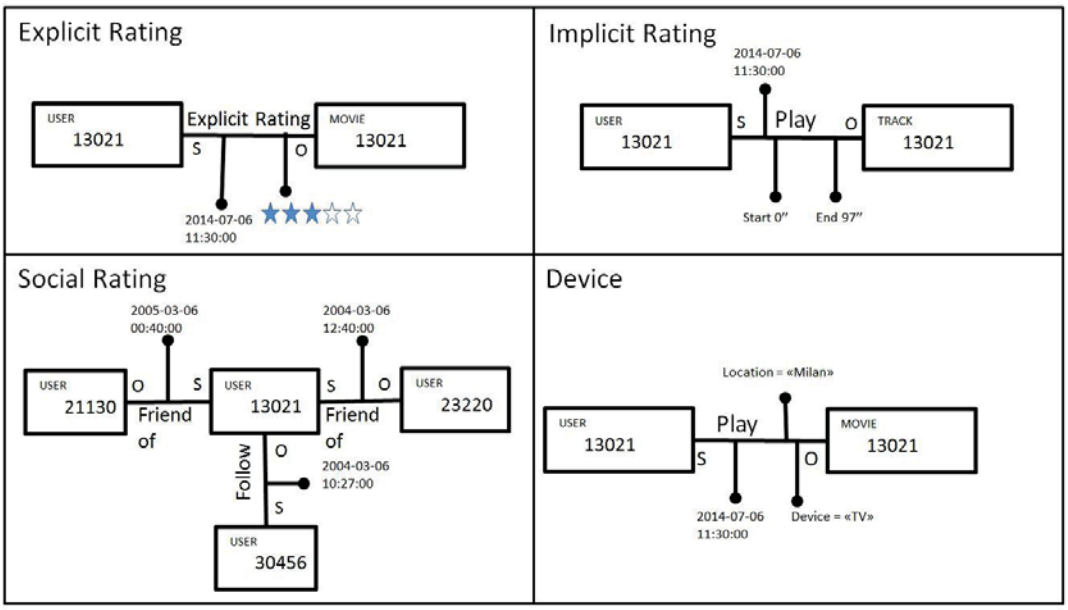

Figure 3 – Modeling of different feedback types

EVALUATOR (EVL)

The Evaluator is the component that contains the logic required to evaluate the recommendation algorithms. In particular, the EVL is able to (i) split the dataset according to a certain evaluation strategy and (ii) compute the quality metrics on the results returned by the recommendation algorithm.

The EVL divides the dataset into training and test set (Figure 4). The **test set** contains the data completely hidden to the recommendation algorithm. These data are visible only to the EVL (and ORC), while the CPE is totally unaware of such data.

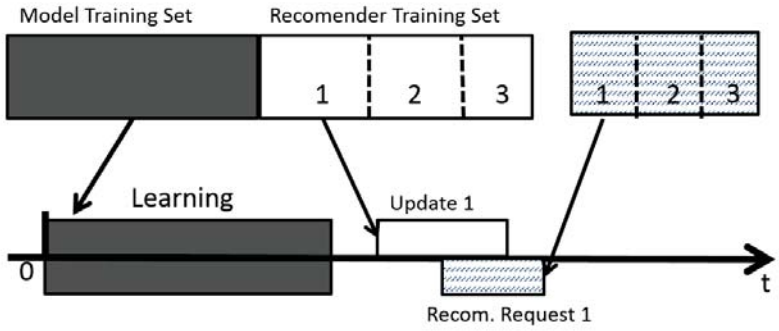

Figure 4 – Testing setup

The **training set** contains the data visible to the CPE and can be used to train the recommendation algorithms. While the training/test set splitting is a pattern commonly used overall in machine learning and in the recommender system domain as well, CrowdRec's EVL introduces a more advanced logic. Namely, we distinguish two separate training set concepts and accordingly, two separate training subsets can be created, denoted as the *model training set*

and the *recommendation training set*. These two subsets differ with respect to the point in time at which they are provided to the CPE. The *model training set* is provided to the algorithm when the CPE is started in order to bootstrap the recommendation engine (e.g., the ratings to compute the Singular Value Decomposition of a collaborative filtering algorithm). On the other hand, the *recommendation training set* is composed of all data provided to the recommendation algorithm only when a recommendation is requested. Note that some of the three subsets can be empty (e.g., the recommendation training set might be missing) and, in addition, some subsets can be overlapped.

The two main goals of the recommendation training set are:

i. to evaluate stream recommendations. For instance, in the news domain, new entities (e.g., news) and relations (e.g., a user reads an article) continuously occur. Thus, the recommendation training set allows providing the CPE with the updated data.
ii. to recommend entities on-the-fly. For instance, it is not uncommon that especially in the case of very large datasets—the algorithms are trained with a part of the users (this data refers to the model training set). Anyway, recommendations can be requested for any users, also those not included in the model training set; consequently, the recommendation training set is used to provide the user profile (e.g., the ratings) at the time we call the recommendation service, enabling the recommendation algorithm to compute a user model on-the-fly.

Worth noting is the fact that some algorithms will be able to learn incrementally, while others can only use the original dataset extended with the updates. In the latter case, either all training data are re-sent (very inefficient and costly) or the CPE is capable to store all the data (more efficient, though more sensitive to event loss).

COMPUTING ENVIRONMENT (CPE)

The Computing Environment consists of the environment where recommendation algorithms are executed, together with the related implementations. As an example, it contains the recommendation algorithm and all the required libraries and data (e.g., internal data—such as dictionaries and user models—data received by the ORC and internally cached, etc.). Among the other functionalities, the CPE must be able to serve recommendation requests and to provide some system statistics (e.g., CPU times, I/O activity).

In order to support easy use of the Reference Framework, the CPEs released with the framework will be deployed as virtual machines, automatically provisioned through provisioning tools such as Vagrant and Puppet. This option grants any practitioner the ability to experiment with the Reference Framework, also ensuring anybody to use the same identical environment producing fully comparable outcomes. Furthermore, these kinds of environments can be straightforwardly run by cloud providers.

RUNNING EXPERIMENTS WITH THE REFERENCE FRAMEWORK

We now exemplify how the Reference framework can be used for two of the formally listed use-cases: tendering and algorithm benchmarking. Panel A of Figure 5 shows the reference framework setup for tendering purposes. ORC is responsible for the overall coordina-

tion of the Reference Framework (e.g., provides data, request and process recommendations). On one hand it is able to compare internal developments (i.e., RSs) or open-source solutions, on the other hand it is capable of providing data and request recommendations from all connecting vendors that are interested in tender. On the vendor side, the tool provides the appropriate API that can be used with any available data set or stream, because it uses a standardized format defined by the Reference Framework. Note that the computing environment should be operated by the vendor, and the operator should share the data with vendors in the tender.

Panel B of Figure 1 represents the setup of reference framework for internal experimentation. ORC provides the data for the algorithms (internally developed or open source) and distributes the recommendation requests between the algorithms by 1) sending all requests for all participants (offline test) or 2) dividing requests equally (online A/B testing). The company may also use open source solutions deployed in CPEs hosted by the company itself; for standard open source solutions (Mahout, MyMediaLite), CrowdRec provides the virtual machines ready-to-be installed.

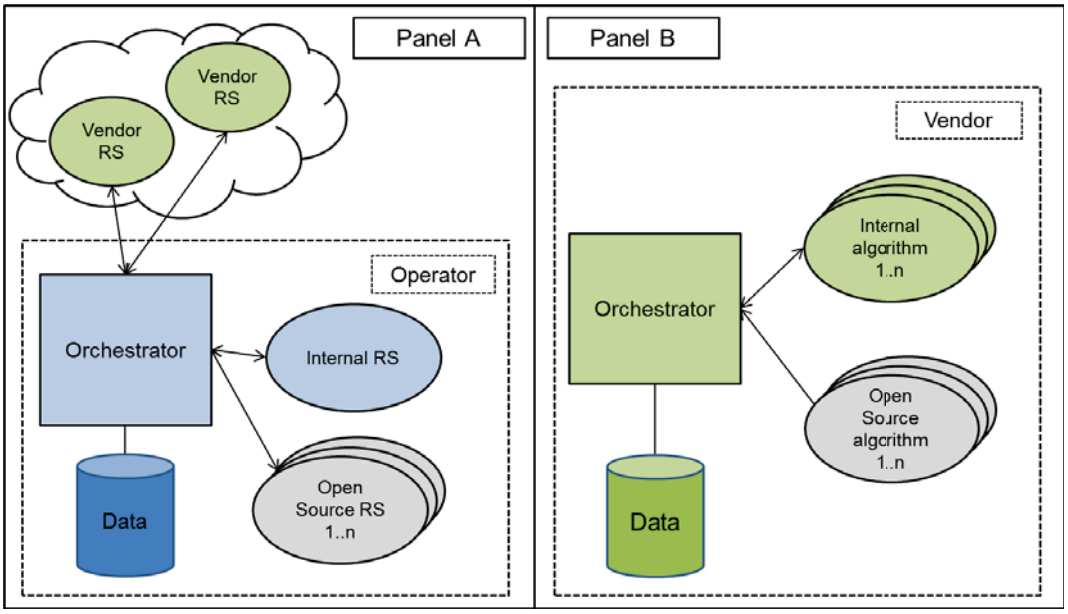

Figure 5 – Reference Framework setup for tendering and algorithm benchmarking

The Reference Framework provides an environment to execute and compare different recommendation algorithms. An experiment consists of a dataset to use, the CPE to run, and the EVL to execute. Depending on the data, *offline* or *online* experiments can be run.

Offline experiments test recommendation algorithms over existing datasets previously stored in the DL. The dataset is retrieved by the ORC that, cooperating with the EVL, splits it into three subsets (potentially, the splitting might already have been persisted). The model training set is provided to the CPE that will read it and train the configured recommendation algorithms. The ORC requests the CPE to serve recommendations for some users (and provides the related recommendation training set). The output generated by the recommendation algorithm, together with some metrics about resource consumption (e.g., CPU/memory usage), is processed by the EVL to compute some quality (e.g., root mean

square error, recall/precision) and performance (e.g., computing time) metrics [6].

Online experiments — for example, the case of **A/B testing** or **online contests**[6] — take into consideration online streams of data. In contrast to offline data sets, such data are not available in advance, but only at the time they are generated (though some "historical" data could have previously persisted in the DL to be used to bootstrap the algorithms). As soon as new data are made available, they are provided to the CPE (in the recommendation training set). The Reference Framework serves online recommendations asynchronously, as they are requested. Again, quality metrics such as conversion rate and click-through-rate (CTR) can be computed together with performance metrics such as response time. It is important to observe that in contrast to offline data sets, in online data sets the actual feedback (i.e., the test set) might not be immediately available, rather it might be collected later (or never).

Performance requires a particular attention in the case of online experiments, where the communication overhead between the components plays an important role to fulfill the requirements of a service level agreement (SLA). The most obvious set-up is that DL, ORC, and EVL are maintained by the same operator company and CPE is provided by another company or group. Hence, the bottleneck of this data flow is the communication overhead between the ORC and CPE. To overcome this, the operator may grant private access and computational resources in its own network, where the participants can establish their environment and serve recommendations. Another solution can be that computational component is provided as a compact package and deployed to the operator's environment.

Note that the evaluation of **data stream recommendations** can occur either online: at the rate that new data arrive, or offline: by splitting the training set on the basis of the timestamp and simulating the arrival of updated data. Such time-based splitting ensures a realistic simulation of the real user activity and an evaluation close to the actual quality perception. Several different test windows should be aggregated to ensure the significance of the evaluation results [7].

CONCLUSION

We have introduced a Reference Framework that addresses the needs of TV operators and content providers to carry out a realistic comparative testing that is necessary to make proper decisions regarding which recommender algorithms they should adopt for their use-cases and their data. The key aim is to ensure that the recommendations have the highest degree of relevance to the consumer. The Framework can significantly lessen the time and cost required by selecting an appropriate solution or vendor fulfilling the qualitative and technical requirements of the TV operator. The Framework allows the evaluation of both algorithmic effectiveness and implementation efficiency. Additionally, it is capable of encapsulating open source recommendation and evaluation frameworks. Currently the Framework focuses on comparison of algorithms using static data (batched-based evaluation), however in the future the focus will be more on streaming data (for stream-based

[6] e.g., http://www.clef-newsreel.org/

evaluation). The framework makes it possible to evaluate proprietary data, either in online or offline mode, and also supports end-to-end measurements for recommendation performance; going beyond the quality of recommendations, including also robustness and scalability. The inclusion of these features sets the framework apart from other tools that are currently available.

ACKNOWLEDGEMENTS

This work is funded by European Union's Seventh Frame work Programme (FP7/2007-2013) under CrowdRec Grant Agreement n° 610594.

REFERENCES

[1] A. Said, D. Tikk, K. Stumpf, Y. Shi, M. Larson, and P. Cremonesi. 2012. Recommender systems evaluation: a 3D benchmark. Proceedings of the Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE'12), Dublin, Ireland, September 9, 2012, CEUR-WS.org, online http://ceur-ws.org/Vol-910/paper4.pdf, pp. 21–23.

[2] J. A. Konstan and G. Adomavicius. 2013. Toward identification and adoption of best practices in algorithmic recommender systems research. Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys'13). ACM, New York, NY, USA, pp. 23–28.

[3] A. Said, and A. Bellogín. 2014. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. Proceedings of the 8th ACM conference on Recommender systems (RecSys'14). ACM, New York, NY, USA.

[4] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. 2011. Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. Proceedings of the 5th ACM Conference on Recommender Systems (RecSys'11). ACM, New York, NY, USA, pp. 133–140.

[5] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. Proceedings of the 5th ACM Conference on Recommender Systems (RecSys'11). ACM, New York, NY, USA. pp. 305–308.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22, 1 (Jan 2004), pp. 5–53.

[7] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. 2012. Real-time top-N recommendation in social streams. Proceedings of the 6th ACM conference on Recommender systems (RecSys'12). ACM, New York, NY, USA. pp. 59–66.