Real-time News Recommendations using Apache Spark

Jaschar Domann, Jens Meiners, Lea Helmers, and Andreas Lommatzsch

Technische Universität Berlin
Agent Technologies in Business Applications and Telecommunication Group (AOT)
Ernst-Reuter-Platz 7, D-10587 Berlin, Germany
{fistname.lastname}@campus.tu-berlin.de,
http://www.aot.tu-berlin.de

Abstract. Recommending news articles is a challenging task due to the continuous changes in the set of available news articles and the context-dependent preferences of users. Traditional recommender approaches are optimized for analyzing static data sets. In news recommendation scenarios, characterized by continuous changes, high volume of messages, and tight time constraints, alternative approaches are needed. In this work we present a highly scalable recommender system optimized for the processing of streams. We evaluate the system in the CLEF NewsREEL challenge. Our system is built on Apache Spark enabling the distributed processing of recommendation requests ensuring the scalability of our approach. The evaluation of the implemented system shows that our approach is suitable for the news recommenation scenario and provides high-quality results while satisfying the tight time constraints.

Keywords: Apache Spark, stream recommender, distributed algorithms, real-time recommendation, scalable machine learning

1 Motivation

In the last years, there has been an immense growth in the popularity of online newspapers. The huge amount of published news makes it difficult to find the relevant articles. Therefore, services are needed supporting users in browsing the news and guiding users to the most interesting news taking into account the users preferences and the specific context. In contrast to traditional "printed" newspapers, online news portals offer news much faster. Furthermore, online portal can adapt the visualization of news dynamically taking into consideration the most recent trends, the context and individual user preferences. The integrated recommender services have an important influence on the news portals. Thus, the development and optimization of recommender systems for online news portals is a very active field of research.

In order to support users in finding relevant items in large data collections, recommendation algorithms have been developed. Traditionally, these approaches are based on complex models computed on static data sets. In the field of news recommendation, the sets of items change continuously. Therefore, the real-time adaptation of the applied models must be assured.

In this work we present an approach for the dynamic real-time creation of news recommendation models. We implement a most-popular algorithm that keeps track of the currently most frequently read news articles based on the Apache Spark framework[1]. Apache Spark enables the distributed processing and assures high scalability. We evaluate the algorithm in the CLEF NewsREEL challenge. This allows us to analyze our approach in two tasks. The Living Lab evaluation (NewsREEL Task 1) enables the recommender evaluation based on live user interaction; an Evaluation Lab (NewsREEL Task 2) focuses on the evaluation based on re-playing a data stream recorded in the living lab scenario. The evaluation data set has been provided by plista[2], a company specialized in online advertisement. plista provides recommendations for most German news portals.

The remaining paper is structured as follows: Section 2 describes the scenario in detail and points out the specific challenges. Related work is discussed in Section 3. In Section 4, we present our approach and explain the applied methods. The evaluation results are discussed in Section 5. Finally, a conclusion and an outlook to future work are given in Section 6.

2 Fundamentals

In this Section we present the scenario in detail and explain the specific challenges of the Living Lab news recommendation task.

2.1 Scenario Description

The CLEF NewsREEL[3] lab gives researchers the possibility to evaluate news recommender algorithms under realistic conditions [5]. The NewsREEL challenge is co-organized by plista. plista hosts the "Open Recommendation Platform" giving researchers access to live data. Teams registered for the NewsREEL Living Lab receive live information about published news, the interactions of users with items, and recommendation requests that must be answered within 100ms. Researchers can test developed algorithms with respect to recommendation precision (measured by the Click-Through Rate) and technical aspects (measured by the response rate). In this paper we present an approach optimized for delivering news recommendations at large scale applying the Apache Spark framework[1] and its MapReduce technique for assuring high scalability.

[1] http://spark.apache.org
[2] https://www.plista.com
[3] http://www.clef-newsreel.org

Data Analysis In order to get insights into user preferences and the typical user-item interactions, we analyze the offline log files provided in the NewsREEL challenge. The dataset consists of $\approx$ 100 million interactions recorded in July and August 2014 [7]. A record refers either to a freshly published news article, an article update, an "impression" (a user reads an article), a recommendation request, or to a click event (a user clicks on a recommendation). The dataset contains three different news portals from different domains (sport, local news, tech news). Each record is represented as a vector with several features encoding information such as *browser brand*, *article category*, *user location*, *time*, etc. The features *age*, *gender*, and *income* however, had the same default value for all users as the information is not tracked by plista. A detailed description of the dataset is given by Kille et al. [6].

The analysis of the dataset shows that the number of interactions per user is small. This makes it difficult to adapt to individual user preferences since too little information about individual users is available. In addition, the user preferences are highly volatile and strongly influenced by recent events and the time of the day. Based on this observation we focus on algorithms that analyze recent trends and the taste of the "crowd" for computing recommendations.

Evaluation metrics The performance of a news recommendation system can be measured based on the Click-Through Rate (CTR) which is the ratio of users who click on a recommendation to the total number of offered recommendations.

$$CTR = \frac{\#\text{clicks}}{\#\text{recommendations}}$$

The evaluation scores are computed by plista. The results are visualized in an online portal for all participating recommendation systems on a weekly basis.

2.2 Challenges

The provision of high quality news recommendations is a challenging task. Compared with traditional scenarios (focusing on recommending books or movies), the continuous changes in the set of users and items as well as the context-dependent relevance of items are difficult to handle. The short lifecycle of news items and fast changes in the user interest require a continuous adaption of the recommender models. Furthermore, the Living Lab evaluation in the CLEF NewsREEL defines tight time constraints that must be fulfilled also in load peak. This requires highly optimized data structures and algorithms scaling well with the number of requests per second.

3 Related Work

In this Section we discuss existing recommender algorithms as well as news recommender systems and relevant frameworks.

3.1 Existing Recommender Algorithms

The task of recommending is crucial in many different application scenarios. Over the last decades several approaches have been suggested for computing recommendations. We review the most commonly used approaches and discuss the specific strengths and weaknesses. We focus on collaborative filtering (CF) algorithms, most-popular items methods, content-based algorithms as well as on hybrid approaches combining two basic approaches [1, 11].

Most Popular Items Most-popular Items algorithms suggest the most popular item in the community. The idea behind this popularity-based method is that items rated as relevant by a huge number of users are relevant for new users. The strengths of this approach are that no detailed knowledge about individual user preferences is required. Furthermore, these algorithms are able to adapt to new trends quickly. In addition, recommendations can be efficiently pre-computed since individual user preferences are not taken into account. The weaknesses of the most-popular items paradigm are that neither context nor individual user preferences are considered. This may lead to a reduced recommendation precision. Moreover, there is a high probability that the most popular items are already known to users.

User-based Collaborative Filtering User-based CF algorithms recommend items to users by determining users having a similar taste to the active user. The similarity is computed by analyzing the user behavior in the past. I.e., if user A and user B both assigned equal ratings to items c and d, users A and B are likely to have a similar taste. If user A rated e positively, a user-based CF recommender algorithm concludes that item e is a good suggestion for user B.

Item-based Collaborative Filtering Item-based CF algorithms compute the similarity between items based on user feedback. Two items a and b are considered as similar if users tend to assign the same rating for a and b ("users who liked a also liked b"). If a user U rated item a positively, the system suggests an item similar but still unknown to U. The strength of this approach is that anonymous user feedback can be used and valuable explanation can be provided.

Content-based Recommender Methods Content-based methods determine the items that should be recommended to the user by considering their ratings or consumption history. They try to find the commonalities between the items in the user's history. Then these methods search for other items with similar characteristics. A requirement for this approach is that a certain amount of information on the user is indispensable.

Hybrid Recommender Approaches Hybrid approaches combine several of the presented recommender techniques. The combination helps to overcome the weaknesses of single algorithms and usually improves the recommendation precision in most scenarios.

3.2 Frameworks for Processing Large Datasets

Since news recommender system must be able to handle huge amount of messages, the scalability of online news recommender systems is an important requirement. In the NewsREEL challenge recommendation requests must be answered within 100 *ms* in all contexts and all load levels. This implies that a large amount of data must be processed within a short time period. Several frameworks have been developed tailored for the efficient processing of big data relying on the distributed processing, such as Apache Flink and Apache Spark. Spark seems to have a larger number of users and supporters as well as a more detailed documentation. Therefore, we decided to use the Apache Spark framework for our implementation.

3.3 Existing Recommender Systems

Most existing recommender systems use either content-based or CF-based algorithms (cf. Sec. 2).

The content-based approach has been applied in a range of recommender systems. Unlike CF methods, in its basic implementation it cannot take variability between users into account. Carreira et al. [2], e.g., developed a news reader for wireless devices that creates user profiles based on the users reading behavior. The created profiles incorporate user preferences and variability between users. When users decide to use the system, they first have to provide ratings for different categories describing the level of interest in these categories. Afterwards, the system does not require addition explicit feedback. The user profiles are kept up-to-date by evaluating the reading behavior taking into account how much time users spend on an article. Stating that Carreira et al. are able to infer the user interests in articles with high accuracy, the approach relies on information not available in the analyzed NewsREEL scenario. Thus, this approach does not seem to be applicable in our case.

Kim and Chan [9] describe an algorithm for creating user profiles that neither depends on initial ratings nor on the time users spent on an article. Their approach consists in constructing a cluster hierarchy from general to specific interests for each user. Therefore, merely the set of articles visited by a user is needed. Our data analysis has revealed that a large number of persons reads only one article such that we have too little information at hand for this approach.

CF is the other approach that has been applied in a wide range of news recommender systems. Konstan et al. [10] provide news recommendations based on explicit user ratings. With this method, the user interests can be captured reliably. The additional effort for the users can be seen as critical disadvantage of that approach. In the algorithm designed for Google News, Das et al. [3] avoid this inconvenience by relying only on the click history of registered users. These implicit ratings improve the usability; but implicit ratings are less reliable and do not allow users giving negative feedback.

An example of a hybrid approach of content-based methods and CF is described in [12]. The authors developed an algorithm for providing recommendations on Google News. The system calculates a content-based score for each article based on the user's genuine news interests. Therefore, the system analyzes the user's click history taking into account the influence of current news trends at his location. This content-based score is then multiplied with the CF score that is calculated as described in [3].

3.4 Discussion

Most of the described content-based and CF algorithms depend on detailed user and item data not available in the analyzed NewsREEL scenario. The analysis of the NewsREEL data has shown that a large fraction of users has read only one article. The user data provided in NewsREEL seems to be noisy due to privacy issues. Given these restrictions, we decided to use a most-popular approach for recommending the news articles. We contribute to the research field of news recommendation by developing a highly scalable recommendation system using a state-of-the-art parallelization framework. Our system yields accurate recommendations depending on very little user information.

4 Approach

Given the very sparse user information provided by our data set, we refrained from applying a CF algorithm. In order to avoid computational overhead, we decide implementing a simple but effective most-popular approach. The recommendation strategy is applied separately for each of the three news domains and can optionally be refined for the provided categories ("sub-domains") in each domain. In the NewsREEL setting, users are not asked to explicitly rate articles. We treat each user-item interaction ("impression", "click") as an indicator that the user is interested in the news item since the user proactively clicked on the news article. In the most-popular approach, the challenge is to determine the most read news items at high scale. As the amount of articles steadily increases, the model must be updated continuously. In our system, we make use of the MapReduce mechanism provided in Spark. MapReduce is an appropriate method for quickly determining the most popular articles as it allows us to count the number of clicks each article in the stream received in a highly parallelized way. We store the incoming user interactions in a ring buffer of a predefined size. If an element (i.e. a user-article interaction) is added to the buffer, a MapReduce algorithm is run using Spark to count the number of received clicks for each article in order to update the current model. Due to concurrency issues, only one MapReduce phase runs on a buffer at a time. If the data stream is fast there might be interactions that are overwritten without being incorporated into the model. Hence, choosing a good buffer size is highly relevant for ensuring high-quality recommendations.

4.1 Implementation

In this section, we present the architecture of the implemented system and discuss technical issues.

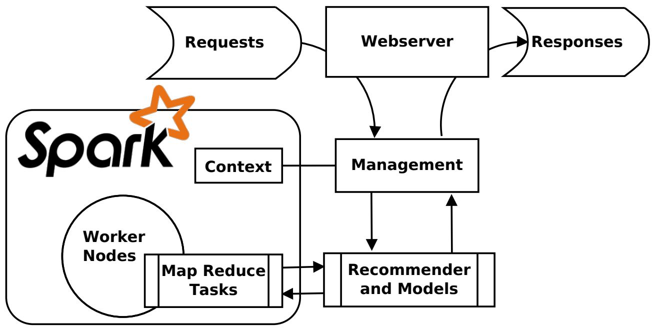

Fig. 1. Integration of the web server and Spark. The web server delegates recommendation requests to the recommender in order to retrieve results. The web server, management component, and the Spark-context are started in the same JVM to ensure fast communication.

The API defined in the NewsREEL Living Lab scenario defines that a web server must be used for handling messages and requests. The integration of Spark into a web server is a challenging requirement, since the Spark Streaming API does not provide a native consumer. In order to not depend on third party software, we decided to handle the requests ourselves as shown in Fig. 1. This has the benefit that the implemented approach allows us to pre-process the requests right away rather than sending them through a third-party messaging system like Apache Kafka. The management component handles pre-processed requests from the web server, delegating the requests to the respective recommender. The recommender handles several MapReduce tasks that run through Spark. If the system runs on a standalone Spark cluster, the web server is available on the same machine as the cluster master while the MapReduce jobs are parallelized on the slave nodes.

To further scale with the high demand while having a stable system, incoming impressions are stored in an optimized queue of fixed size following the first-in-first-out scheme. For each domain and category a separate queue exists containing the most recent items. The MapReduce jobs compiling the recommendation results are scheduled such that there is only one active job at a time for each queue. Every time the queues are altered in some way, the system makes sure that another job will be started to update the results. Hence, in times of high throughput, a small queue size can force some items to be overridden without being processed by a MapReduce job. The loss of precision that this behavior might cause is acceptable if more popular articles still dominate the sample of impressions contained in the queues. Another possibility of bypassing this problem is to provide a bigger cluster such that the MapReduce jobs finish in time to incorporate new results. If a smaller queue size is chosen the adaptation to new trends is enhanced while a bigger queue size can store more details of the stream and reduces precision loss.

5 Evaluation

We evaluate the recommender system both in the Living Lab and in the Evaluation Lab scenario. First we analyze the performance of our component in the Evaluation Lab scenario. This allows us simulating high message load on our system to test scalability and stability. Second we analyze the response rate of our system in the Living Lab scenario.

5.1 Scalability Analysis in the Evaluation Lab

In this section we evaluate the system with the focus on scalability. The scalability is an important aspect in news recommender systems due to the huge number of messages and the high variance in the messages volume induced by specific events. Figure 2 shows that the number of impressions recorded by our

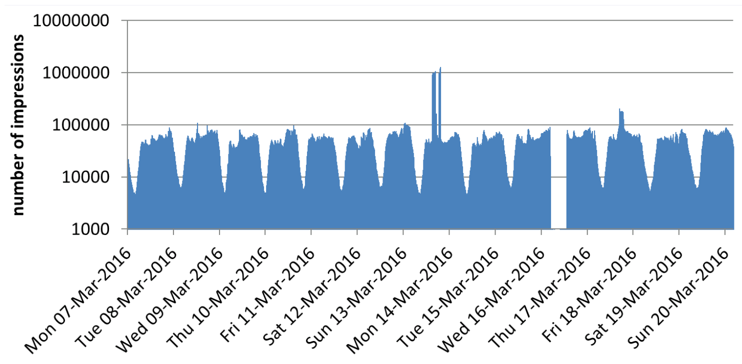

Fig. 2. The graph shows the number of impressions that where recorded within two weeks in March. The bin size for this plot is on a 15 minutes basis.

system can change significantly over time. The high amplitude in Figure 2 relates to several big transfers of soccer players. As the articles in the presented news domain are dominated by sports news, the increase in impressions related to this kind of special events seems reasonable.

To further assess the capabilities of the system, we evaluated the throughput under extreme circumstances. Figure 3 shows the response time histogram on a single machine with 1000 threads simultaneously requesting recommendations and sending impressions.

It can be seen that there are three peaks. The first peak marks requests that are answered immediately. This case occurs if the incoming JSON-string is corrupted or required information is missing. The second peak represents requests that are cached while the third are normal responses that are handled by the management and MapReduce components. The mean response time is about

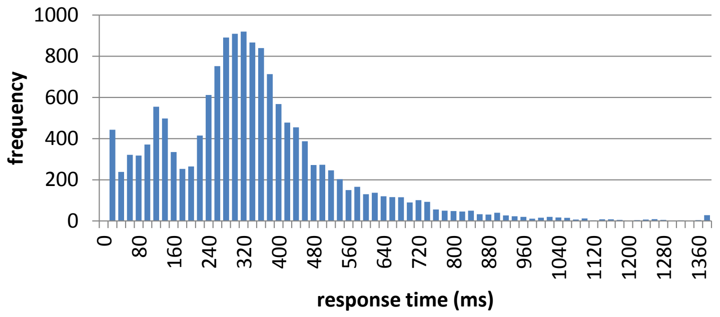

Fig. 3. The graph shows the frequency of response times for 15k recommendation requests with 1k parallel request threads.

330 ms and considerably larger than the minimum response time constraint of 100 ms. This can be explained by the fact that the environment resided on a single machine with maximized throughput. In a production system, the expected mean response time is narrowed accordingly because of the distribution of the system and better scalability with high parallelism and throughput.

We analyze how distributing the system over several machines (using the Spark framework mechanism) influences the performance. We built a cluster that consists of a master and two slave nodes as shown in Figure 4. The master resides on a virtual machine with 2 CPU's and 2.9 GB RAM. The machine is located outside of the local network and also hosts the application with the web server. The two slaves are located within the local network and have 8 CPU's each and 16 or 32 GB RAM. On each slave node, we deployed 4 worker instances that occupied 2 CPU's and 1.7 GM RAM each. We strongly recommend deploying the cluster locally for a production use case. For research purposes, we wanted to evaluate the impact of higher network latencies within the cluster. This scenario may also be of interest if the cluster must be available locally with a low risk of failure while a third party hosts the virtual machines for computations.

For the evaluation we recorded the NewsREEL Living Lab messages in May 2016. The messages in this data set are sent from the local network to the remote master to incorporate network latencies.

In the first evaluation only one slave node with four workers has been used. In this scenario we test the current setup and determine the response time distribution of the system. In the second evaluation both slave nodes and eight workers have been used. This scenario determines the influence more nodes and workers have on the results. With more computational power in the scenario with more workers, the quality of the results and the number of jobs that finished increased.

Summarizing the evaluation results, we state that the API limits us to use a single webserver as an entry point for messages from the service. In order to respond in time in the Living Lab scenario, the webserver must reside on a machine with many CPU cores to compensate for higher demand. Using more

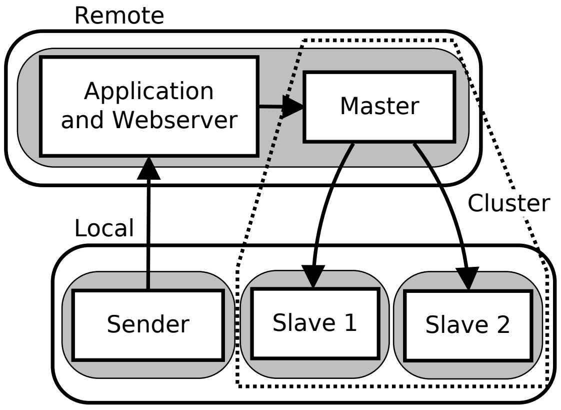

Fig. 4. The figure shows the distribution of the system. A gray background denotes a distinct virtual machine. The arrows show the path a request takes along the system and the dotted area encloses the cluster components.

Table 1. Comparison of offline CTR from various recommending algorithms

Algorithm	**Description**	**CTR**
MostPopular300	Uses a most-popular algorithm, operates on a queue of items with size 300	2.93 %
CategoryMP100	Uses a most-popular algorithm considering categories (with separate queues for each category), operates on queues of items with size 100	2.84 %
MostPopular100	Uses a most-popular algorithm, operates on a queue of items with size 100	2.83 %
Baseline	Recommends most recent items	0.59 %

slave nodes via the Spark frameworks enables us to improve recommendation quality while ensuring a reliable response rate of our system.

5.2 Click-Through Rate Analysis in the Evaluation Lab

For the Evaluation Lab a data record of one week in the period between 7 March and 13 March 2016 is used. This time period has been selected because in this timeframe the Living Lab data stream provided a reliably high volume of data.

Kille et al. [8] introduced an offline evaluation approach that we applied for the presented system. A stream was replayed which was recorded in a Living Lab setting. The message order is preserved. In contrast to the Living Lab setting the evaluation metric is based on future user impressions. An impression represents the evens that a user accesses a news article, not necessarily by clicking on a

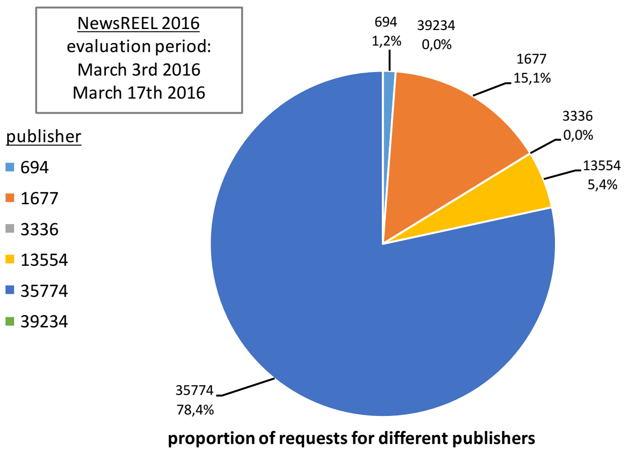

Fig. 5. The figure indicates that requests for the publisher `35774` dominate the NewsREEL challenge.

recommendation. A recommendation for a user is handled as correct in case the user reads the recommended article within a 5 minutes time window after the request. The impression based the offline CLICK-THROUGH RATE (CTR_{offline}) is calculated in the following way.

$$CTR_{\text{offline}} = \frac{\#\text{A}}{\#\text{S}}$$

Multiset A represents the recommendations handled as correct and multiset S represents all recommendations, which were given by the algorithm, in the evaluation period.

In the evaluation process three most popular algorithms, labeled with MOSTPOPULAR300, MOSTPOPULAR100 and CATEGORYMP100, have been deployed on a standalone local SPARK cluster running on a single machine. The number in the label describes the size of the operation queue which is used by the algorithm. The replay of the data record is done on the same system. In reference to Figure 5 a data record analysis shows that the domain with `ID 35774` is dominating the number of recommendation requests. Hence, the evaluation results are based on an analysis of the impression and requests related to domain `35774`. The results are shown in Table 1 and indicate that all three algorithms significantly exceed the CTR_{offline} performance of the baseline. This verifies that the evaluated algorithms generate adequate recommendations.

5.3 Response Rate Analysis in the Living Lab Evaluation

The implemented system has been evaluated on a single machine in the Living Lab scenario [4]. The performance of the system has been computed based on log files provided by PLISTA. The log files include information about the CTR and the response rate for each algorithm on a daily basis. The latter rate only incorporates responses that did not break any constraints (such as the response time limit of 100 ms). The response rate is of special interest because the CTR was not very consistent, partly because the total number of requests sent by PLISTA per day may change by a three digit factor. We show the response time for different time periods in Table 2. The Table shows that the system outperforms most other algorithms with respects to the total number of days that the algorithms has been active. Overall, with a response rate of over 99 %, we clearly achieved the goal of high availability.

Table 2. Comparison of response rates from the Living Lab scenario from the 2nd of April to the 1st of June in 2016. The last column shows the number of days the algorithm was active. The table entries are sorted by the minimal response rate column in descending order. The presented system is labeled *Spark* while other participant names are anonymized.

Algorithm label	**Mean response rate**	**Minimal response rate**	**days**
3	99.88 %	99.62 %	30
Spark	99.76 %	99.43 %	30
6	99.87 %	99.43 %	28
5	99.87 %	99.24 %	27
1	98.46 %	84.10 %	29
7	98.62 %	79.82 %	30
2	97.40 %	75.11 %	28
4	95.95 %	52.43 %	26

5.4 Discussion

The presented system fulfills the real-time requirements of handling streams with strict time constraints. The Evaluation Lab results ensured the significance of the measured results. The evaluation shows the high quality of the recommendations and the scalability of the system. Furthermore, the system can easily be applied to more demanding scenarios by distributing the system among several machines of CPU cores.

6 Conclusion

In this paper we present a recommender system implementing a most popular strategy. Our system counts the number of impressions for the most recently read articles and suggests the currently most read news articles to the users. The recommender is based on the idea that articles interesting for many readers are also relevant for new users. The high requirements with respect to scalability and response time we address by building our recommender system on Apache Spark. The use of Apache Spark enables us running the recommender as a distributed system in a cluster ensuring the scalability of the approach. We have evaluated the system in a Living Lab and Evaluation Lab scenario. The Living Lab evaluation shows that our recommender system reliably reaches a good CTR. In the Evaluation Lab we showed that the system can handle huge data streams efficiently.

Future Work In the future we plan to evaluate additional recommendation models in order to further improve the CTR. Scaling the cluster horizontally makes it possible to use more elaborate algorithms from the Spark machine learning library while corresponding to the strict time constraints.

Acknowledgments

The research leading to these results was performed in the CrowdRec project, which has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement Nr. 610594.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.
2. R. Carreira, J. M. Crato, D. Gonçalves, and J. A. Jorge. Evaluating adaptive user profiles for news classification. In *Proceedings of the 9th Intl. Conf. on Intelligent User Interfaces*, IUI '04, pages 206–212, New York, NY, USA, 2004. ACM.
3. A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th Intl. Conf. on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.
4. F. Hopfgartner, B. Kille, A. Lommatzsch, T. Plumbaum, T. Brodt, and T. Heintz. *Information Access Evaluation. Multilinguality, Multimodality, and Interaction: 5th International Conference of the CLEF Initiative, CLEF 2014, Sheffield, UK, September 15-18, 2014. Proceedings*, chapter Benchmarking News Recommendations in a Living Lab, pages 250–267. Springer International Publishing, 2014.
5. F. Hopfgartner, T. Brodt, J. Seiler, B. Kille, A. Lommatzsch, M. Larson, R. Turrin, and A. Serény. Benchmarking news recommendations: The clef newsreel use case. *SIGIR Forum*, 49(2):129–136, Jan. 2016. ISSN 0163-5840. doi: 10.1145/2888422.2888443. URL `http://doi.acm.org/10.1145/2888422.2888443`.
6. B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *NRS13: Proceedings of the International Workshop and Challenge on News Recommender Systems*, ICPS, page 1422. ACM, 10 2013.
7. B. Kille, A. Lommatzsch, R. Turrin, A. Sereny, M. Larson, T. Brodt, J. Seiler, and F. Hopfgartner. Stream-based recommendations: Online and offline evaluation as a service. In *CLEF'15: Proceedings of the 6th International Conference of the CLEF Initiative*, LNCS, vol. 9283, pages 497–517. Springer Verlag, 2015. ISBN 978-3-319-24026-8.

8. B. Kille, A. Lommatzsch, G. Gebremeskel, F. Hopfgartner, M. Larson, J. Seiler, D. Malagoli, A. Sereny, T. Brodt, and A. de Vries. Overview of NewsREEL'16: Multi-dimensional Evaluation of Real-Time Stream- Recommendation Algorithms. In N. Fuhr, P. Quaresma, B. Larsen, T. Goncalves, K. Balog, C. Macdonald, L. Cappellato, and N. Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction 7th Intl. Conf. of the CLEF Association, CLEF 2016, Evora, Portugal, September 5-8, 2016.*, LNCS 9822. Springer, 2016.
9. H. R. Kim and P. K. Chan. Learning implicit user interest hierarchy for context in personalization. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 101–108, New York, NY, USA, 2003. ACM Press.
10. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
11. L. Li, D.-D. Wang, S.-Z. Zhu, and T. Li. Personalized news recommendation: A review and an experimental investigation. *Journal of Computer Science and Technology*, 26(5):754–766, 2011.
12. J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pages 31–40, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-515-4.