Development of a News Recommender System based on Apache Flink

Alexandru Ciobanu[1] and Andreas Lommatzsch[2]

[1] Technische Universität Berlin Straße des 17. Juni, D-10625 Berlin, Germany
alexandru.ciobanu@campus.tu-berlin.de
[2] DAI-Labor, Technische Universität Berlin, Ernst-Reuter-Platz 7, D-10587 Berlin, Germany
andreas@dai-lab.de

Abstract. The amount of data on the web is constantly growing. The separation of relevant from less important information is a challenging task. Due to the huge amount of data available in the World Wide Web, the processing cannot be done manually. Software components are needed that learn the user preferences and support users in finding the relevant information. In this work we present our recommender system tailored for recommending news articles. The developed recommender system continuously analyzes a data-stream using the APACHE FLINK framework, computes recommender models and provides real-time recommendations. The recommendations are optimized on specific news portals and consider the user session. The recommender system analyzes the user-item interactions in real-time and continuously updates the recommender models ensuring that only fresh articles are recommended. We explain the developed architecture of the system and discuss the specific challenges of processing continuous streams. The scalability and the methods for optimizing the parameter configuration are explained. The evaluation in the NEWSREEL Living Lab scenario as well as in the offline evaluation shows that our recommender fulfills the requirements and reaches a good recommendation performance.

Keywords: Apache Flink, stream analysis, recommender system, scalibility, news recommender

1 Introduction

The demand for always being up to date with current events and developments is addressed by the media offering instant publications of the most recent news. The amount of published news articles has steadily grown making it almost impossible for users to read all the published news. Furthermore readers are often interested in a limited set topics or categories. Finding the relevant items in the huge mass of existing items is an issue addressed by recommender systems [6]. Recommending news articles is a challenging task due to the specific properties of news items: News articles tend to be short-living and expire after few days or weeks. The cost for creating and consuming news is relatively low leading to a high volume of items and a big diversity in consuming news [7]. Compared to movies or online shops, most news web sites do not require a login, making the exact identification of users almost impossible. The diversity of topics, the variety of usage scenarios, the limited user tracking capabilities and short life cycle of news item are the major challenges that must be addressed when developing powerful algorithms for recommending news.

News recommender components must be able to handle a huge amount of messages describing the creation and deletion of news articles as well as the interaction between users and items. The amount of data is often represented as a continuous data streams. Due to the steady changes in the user preferences and in the item set, models relying on static sets cannot cover the dynamics of the scenario. Thus recommending suitable articles and analyzing streams are tightly coupled tasks.

Requirements for News Recommenders A news recommender is a piece of software that helps users finding relevant news articles in the huge amount of available news. Recommender algorithms predict what articles (potentially unknown to a user) match the individual user preferences. The user interests are derived from user ratings and the user behavior, e.g. the interactions of users with items. The decision whether an article is relevant or not is made by an algorithm that suggests the most interesting items to the user.

Highlighting the most relevant news articles helps users coping with the huge amount of available items. The recommender component computes the potentially most relevant articles. Based on the analysis of the user behavior the recommender adapts to the user preferences and supports users in selecting the relevant items.

Our Contribution In this paper, we present a recommender system focusing on the efficient handling of data streams. The APACHE FLINK framework is applied for observes the NewsREEL stream and for creating statistics in real-time. The data is aggregated in a model used for the fast provisioning of recommendations (based on a most-popular-algorithm). The paper researches the influences of parameter configurations on the recommendation quality. Variables (such as considered time frame length) are optimized in the offline evaluation.

Structure of the Work The remaining work is structured as follows. Section 2 describes the analyzed scenario and the NEWSREEL Challenge in detail. In the third Section, we discuss related work and already existing solutions. Section 4 explains our approach and explains the design of the developed recommender system. The implementation of the system and the influence of different parameter settings on the system's performance are presented in Section 5. Finally, a conclusion and an outlook on future work are given in Section 6.

2 Problem Description

The NEWSREEL [4] challenge gives researches the possibility to evaluate developed news recommender systems based on real-life data. In the Living Lab scenario (Task 1) recommendations for news articles must be computed for different news portals. In the Offline Scenario (Task 2) the participating teams must provide recommendations for a simulated stream of messages. The structure of the contest is shown in Figure 1.

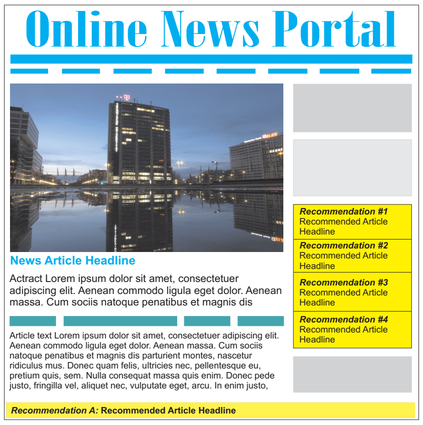
Online News Portal
News Article Headline
Actract Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis
Article text Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo,
Recommendation #1 Recommended Article Headline
Recommendation #2 Recommended Article Headline
Recommendation #3 Recommended Article Headline
Recommendation #4 Recommended Article Headline
Recommendation A: Recommended Article Headline

Fig. 1: The figure visualizes the placement of recommendations on the online news portals.

The communication between news portals and participants of the competition is coordinated by company PLISTA. PLISTA hosts the API ("ORP") that provides all relevant data and assigns the recommendation requests to the participating teams. Furthermore, PLISTA creates statistics, aggregates the log data, and computes the performance of the team in the Living Lab scenario. In the challenge the participating teams receive information about freshly published news articles and the user-item interactions. In addition, teams receive recommendation requests that must be completed within a tight time limit. The quality of the recommendations is measured based on the Click-Through-Rate (CTR). The CTR describes how many of the provided recommendations have been clicked by users (cf. Figure 2) [5].

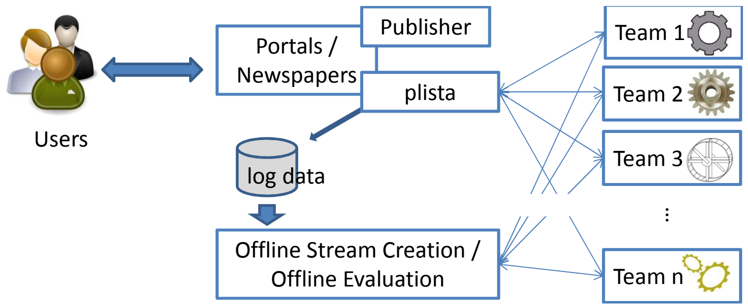

Fig. 2: The figure visualizes the architecture of the NEWSREEL challenge. When a user visits a web portal participating in the challenge, an recommendation request is sent to a registered team. The recommendations are embedded into the web page. If the user clicks on the recommendation, the team is rewarded. In addition to the online evaluation NEWSREEL offers an offline task based on a simulated message stream.

The developed recommender components must be able handle high request volumes. An answer is expected within 100ms which is characteristic for NEWSREEL as stated by Brodt et al. These requirements introduce limitations that need to be considered in the design of recommender systems. Special attention must be put on efficiency and scalability in order to assure high quality recommendation and to fulfill the technical requirements. Due to the data available in the scenario not all types of recommender algorithms can be applied.

3 Related Work

We review existing recommender approaches and discuss how these strategies could be used in the analyzed news recommendation scenario.

3.1 News Recommender Algorithms

Recommender algorithms are software components supporting the users to find items matching the user preferences. Recommender algorithm became popular with growing relevance of e-commerce shops as they help finding interesting products in the long-tail. These recommendation systems can be also applied to news portals. Predicting what news item is interesting for which user is a complex task. Different types of algorithm algorithms have been created to fit the specific characteristics of different scenarios.

News item rated as interesting by a huge number of users are potentially also relevant to new users. Algorithms focusing on recommending the most popular items have a low computational complexity. This enables the fast and efficient processing of a large number of requests. The weakness of the most popular approach is that the recommendations are not personalized. Since the visitors of news websites might are interested in a wide spectrum of topics, most popular recommender may perform poorly since individual preferences are not taken into account. In many analyzed scenarios, personalized recommender algorithms provide more relevant suggestions optimized to the individual user preferences [3]. The weakness of personalized recommender algorithms is that they require comprehensive training data. Furthermore, these algorithms usually have a high computational complexity. This can be problematic in web-based scenarios in that the exact identification of users is impossible and strict response time limits for the recommender services exist.

Recommending news articles requires an approach optimized to the specific requirements of the scenario. The news recommendation scenario differs from scenarios focusing on recommending movies, books or general shopping products regarding the dynamics of the set of items and users and the technical constraints. Said et al. state that characteristic properties of news articles are short relevance period, low consuming costs and wide range of used devices [7]. Most news websites do not require a user authentication which limits the precision of tracking and creation of preferences for users.

3.2 CF-Based Approaches

Collaborative filtering (CF) is the most frequently used approach for providing personalized recommendations. The idea of CF is that users, who showed a similar taste in the past, will like the same items in the future. The similarity between users is computed based on the user behavior and on ratings. In order to recommend an item to a user A, the system determines the most similar users to A and suggests items the similar users liked. These CF-based algorithms perform well on websites with a big amount of different visitors and many interactions. Big companies such asNETFLIX or AMAZON successfully run CF-based recommender systems.

Collaborative filtering is usually applied in scenarios characterized by a static set of items. In the analyzed news recommendation scenario the set of items changes continuously requiring frequent model updates. In addition, the computation of one or more entities (neighborhood) with equal preferences is a time consuming task that requires optimized parameter configuration for the similarity function and the size of the analyzed neighborhood. Lommatzsch and Werner [9] have presented an implementation of item-based collaborative filtering for the NEWSREEL challenge. Their evaluation has shown significant performance differences that depend on news portal and context.

3.3 Stream processing Frameworks

Data on the web is created continuously. In order to capture the most recent events and trends, the data must be processed in real-time. This task can be accomplished by using stream processing frameworks. These libraries provide scalable, distributed architecture and can be accessed using an API. A level of abstraction helps integrating stream analysis into existing applications.

APACHE STORM is a common open source framework which brings the MapReduce paradigm to streams. STORM has been developed since 2011 and has reached a mature stage providing several components and extensions. The technical documentation and the tutorials help new users to integrate the framework in new applications.

APACHE SPARK provides a higher-level API and can be used for both stream and batch-based analysis. The stream component is optional; internally a micro-batching approach is used.

APACHE FLINK provides functions of both worlds regarding to [8]. This new framework is optimized for real-time applications. It combines a high-level API for JAVA, SCALA and PYTHON with highly expressive syntax and can be run in cluster or in local mode.

3.4 Discussion

In our scenario the use of APACHE FLINK seems to be most promising since the features of APACHE FLINK match our requirements best. The framework is used to handle the large NEWSREEL message streams to create a model for our recommender system. Most-popular algorithms have been implemented successfully by other participants. These algorithms can be used for efficiently computing predictions relevant to most users. In our work we combine APACHE FLINK and Most-popular algorithms. We study how a FLINK-based recommender system applying most-popular algorithms performs in the NEWSREEL scenario.

4 Approach

We develop a recommender system tailored to the specific requirements of the NEWSREEL challenge. Our system architecture is optimized for the efficient handling of huge message streams and the continuous adaptation of recommender models. The continuous model updates ensure that only fresh news articles, requested most in the last minutes are recommended. We implement a most popular algorithm; the implementation is build based on APACHE FLINK in order to ensure that huge message streams are efficiently processed. The use of FLINK ensures the scalability and simplifies the distribution of the system over several machines. We use a highly modular system architecture (Figure 3). The system consists of four components:

1. The `HTTP` endpoint receives the NEWSREEL messages. The impression messages are forwarded to the FLINK-based analysis component. The recommendation requests are dispatched to the recommendation request handler. Furthermore the `http` endpoint converts the recommendation into valid `JSON` messages and provides valid answers as defined in the NEWSREEL protocol.
2. The APACHE FLINK-based component analyzes the impression messages and computes the statistic.
3. The models build based on the impression statistics are stored in a database.
4. The models are used by the Request Handler for computing the recommendations for incoming requests.

We discuss the algorithms and data structures used by the different components in the next sections.

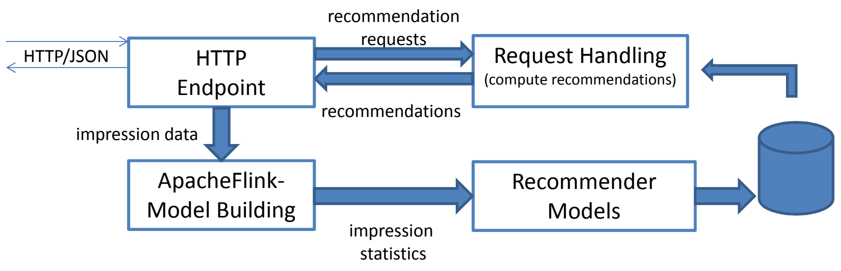

Fig. 3: The figure visualizes the system architecture.

4.1 Endpoint

The `http` endpoint receives the NEWSREEL messages from PLISTA. Messages describe user-item interactions and provide data about freshly published news. The messages are formatted in JSON; the data is sent via HTTP post messages [2]. A Java-based webserver handles the incoming messages. Based on the message type the received data is either forwarded to the FLINK-based component (that keeps the models up-to-date) or to the component that computes the recommendation results.

4.2 Flink Processing

This component is responsible for reading and analyzing the data stream. It is designed to be efficient and scalable through load distribution on multiple cores or machines. FLINK observes the stream and aggregates the information applying a window-based approach. The model only incorporates the most recent items; old items are discarded and treated as outdated. This ensures that the models always describe the most recent items and interactions on the relevant news websites.

APACHE FLINK is used to aggregate the data of every domain ("publisher") and category, which are then transformed into descriptive models (used by the component that provides the results). The stream processing runs completely decoupled from all other processes. This design pattern allows us running time consuming operations without violating the 100ms response time constraint for requests. The separate handling of requests and the real-time analysis of the impression processing ensures the scalability as well as continuous updates of the recommender models.

4.3 Recommender Models

The separation of recommender algorithms and model creation requires communication between the two components. We implement the data exchange based on a data pool used for storing the commonly needed data. Our model stores the statistics as well-structured tuples optimized for relational databases. The database is connected to the FLINK output stream and stores all the aggregated information. For our recommendation algorithms, the portals, the categories, the articles as well as the number of views within the current time frame are stored. This model can be used to answer the following question: "What are the most popular articles within the last minutes in a specific category of a given website?".

For the recommender system, we decided to use a MYSQL database. MYSQL is an open source database server, supporting indexing and the concurrent access (which is required due to concurrency in the system). MYSQL has a big potential of horizontal scaling using master-slave replication. It is widely integrated in most common programming languages such as JAVA [1]. It runs on a big variety of platforms and answers all required queries within a small time frame. The recommender can also benefit from the transparent query cache. Since we do not provide personalized recommendations, the cache speeds up the query handling a lot; most requests can be answered based on cached results.

4.4 Result Creation

The recommendations are computed by a separate component. This component communicates with the database and loads batches of statistics. Due to the distributed writes of APACHE FLINK the upper and lower bounds of a time window need to be detected. A system-wide identifier for every period cannot be set since no system-wide clock is available. The detection of the intervals can be done by comparing timestamps of two successive rows. Due to the fact that writes are very fast, the differences are a few seconds in maximum. Whenever a bigger gap is detected, a new time window is assumed. The decoupling of stream processing and the creation of recommendation results raises another problem: It is impossible to determine whether the currently running write operations have been finished and whether the statistics in the database are complete. We solve this issue by using the previously mentioned interval detection. The implemented strategy ensures that always the most recent, completely written model is used. This solution introduces a delay which is acceptable in our scenario.

The central step for computing the recommendations is the computation of the most popular news items based on the statistics (created by the FLINK-based component). The required sorting and filtering of the data is efficiently done by the database, since databases are optimized for these operations. SQL queries allow us to write compact, human readable code that is fast and reliably execute by the database server.

5 Evaluation

The implemented recommender has been evaluated in Task 2 (offline evaluation) of the NEWSREEL challenge. Our analysis focuses on the recommendation accuracy as well as on technical aspects, such as response time and scalability. We study how different parameter configurations influence the evaluation results as well as discuss strengths and weaknesses of our approach.

5.1 Efficient, Reproducible CTR Optimization

We optimize the parameter configuration of our recommender the NEWSREEL offline evaluation scenario. The offline evaluation environment allows us to analyze different parameter settings concurrently in a reproducible way. In contrast to the online evaluation environment characterized by a high variance in the number of messages, the offline scenario offers a reliably, high volume message stream. The offline evaluation components allow us re-playing the data stream previously recorded in the online scenario. The re-played streams contain exactly the same messages (as the stream at the recorded day); the order of messages is preserved; but the stream is re-played faster in order to speed-up the evaluation. In our evaluation, we use the data collected at May 12th, 2016. Several of the system's configurations have been additionally tested using the data collected at May 15th. In the evaluation we consider only messages from the **sport1** domain due to the very small number of requests for the other publishers.

The Impact of Time Window Size We analyze the influence of the window size used by APACHE FLINK for building the recommendation model. We evaluate the recommendation performance for the following window sizes `t = 600s, 300s, 180s` and `60s`. In the evaluation we simulate a load level of 1,000 concurrent requests. In the online scenario the amount of information is lower because the NEWSREEL participants receive only a small fraction of the traffic. However, this setup represents the productive system load better.

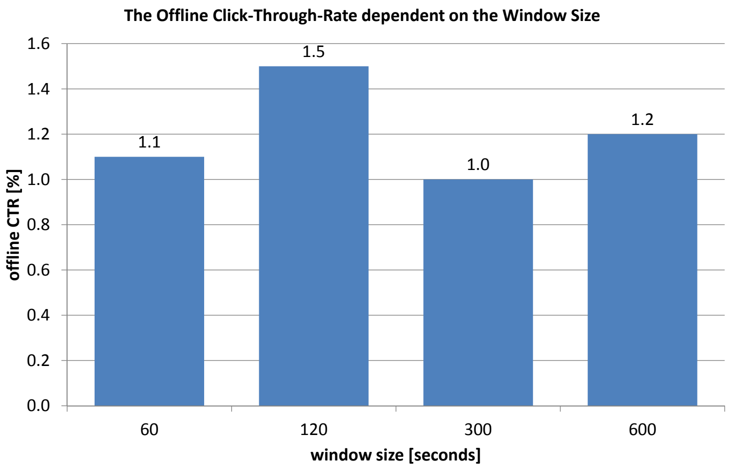

Fig. 4: The figure visualizes the offline CTR dependent on the window size. The graph shows that there is clear relation between the window size and CTR.

The measured CTR dependent from the window size (used by APACHE FLINK) is shown in Figure 4). The Figure shows, that no direct correlation exists. This indicates that the impact of window size on the prediction quality is low. The system reaches an offline CTR of 1.3% using a window size as short as 30 seconds.

Impact of the Re-calculation Interval A challenge in the NEWSREEL scenario is the continuous changes in the user preferences and in the set news items. Thus, the recommender must continuously discard outdated items and compute the relevance of freshly added items. In order to address this challenge, we periodically re-compute our recommender model using APACHE FLINK. We study the impact of the re-calculating interval on the model. Short re-calculation intervals allow the recommender to following trends quickly keeping the model very close to the most recent data in the stream. The disadvantage of short re-calculation intervals consists in a big number write operations (in the database) resulting in a high load on the database. Long re-calculation intervals cause a smaller database load due to smaller number of write operations and smaller updates in the database caches.

We evaluate the influence of different model update intervals p on the recommender performance.

The evaluation (cf. Figure 5 shows that refreshing the model every couple of minutes works well for ensuring a good recommendation quality (in high load scenarios). Shorter re-calculation periods reduce the recommendation precision and increase the system load.

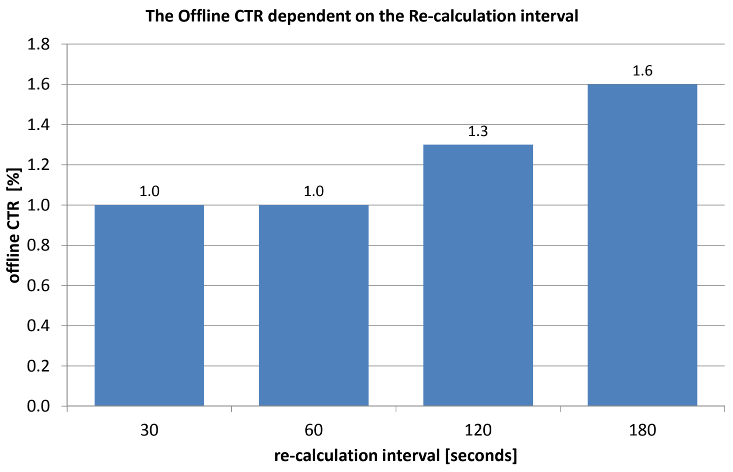

Fig. 5: The figure visualizes the impact of different recalculation intervals. Higher intervals lead to a better CTR.

5.2 Evaluation of Technical Aspects

In the offline evaluation we analyzed the handling of 1,000 concurrent requests. The number of concurrent requests is much higher than the typical number of concurrent requests in the Living Lab ("online") scenario but the high load allows us to maximize the throughput of the recommender. Due to the high number of concurrent requests in the offline evaluation, only a small fraction of requests is handled within the 100ms limit as shown in Figure 6.

The architecture of our system decouples the computation of recommendations and the building of recommender models. This has several advantages. In extreme load peaks, we can apply sub-sampling in order to reduce the effort for building the model. Since the number of requests is small compared with the number of impression messages, a sub-sampling based on the stream of impression data allows us to use the available resources for handling the recommendation requests. Recommendation requests are typically processed very fast since the results have been pre-computed in the model.

Based on our experiences, the bottlenecks in the systems seem to be the used webserver and the database server. In the online evaluation our recommender reached a low error rate. This shows that our recommender reliably handles the number of messages in the online evaluation scenario. In extensive tests we observed on several days that APACHE FLINK stopped writing to the output stream. In order to handle this case, we implemented an observer component that restarts the component in the case of an error.

5.3 Discussion

The evaluation results show that our system reaches the best prediction performance when it updates the recommendation model every few minutes (in the offline scenario).

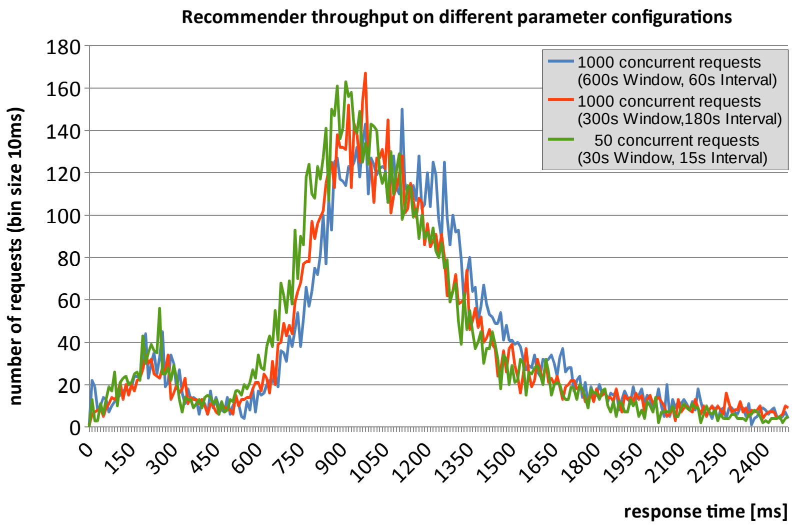

Fig. 6: The chart shows the processing time for different recommender configurations.

Using time windows of less than half of a minute does not result in a significant CTR improvement. The architecture of our recommender systems ensures that huge message streams can be efficiently handled. This is shown by a low error rate in the online evaluation.

6 Conclusion and Future Work

In this work we presented a recommender system implemented based on APACHE FLINK tailored to the news recommendation scenario. The evaluation results show that our system performs well in the contest. The implemented system continuously updating the recommender models is a suitable approach for the efficient processing of message streams. The APACHE FLINK API provides a good abstraction and simplifies the development and adaptation of the recommender algorithms. The decoupling of the model building and the provisioning of recommendations ensures that sophisticated data analysis algorithms can be implemented ensuring that the tight response time constraints are reliably fulfilled.

Our recommender system uses a most popular item model. The model is robust against noisy userIDs and fast changes in the set of news items. The evaluation shows that the recommender reaches a competitive CTR. Our system computes the popularity separately for every domain. The approach can be refined by calculating the most popular models for every category (using the categorization provided in the message meta-data). Further optimization can be reached by adapting the size of the window used for re-calculating the model and by learning the best fitting interval for retraining.

The evaluation with respect to technical aspects showed that our system is highly scalable. The use of APACHE FLINK allows us to distribute the system over multiple machines. The integration of additional machines enables us to concurrently compute the model on several different machines ensuring the scalability of the system.

As future work, we plan to analyze trend extrapolation approaches. Based on the most recent most popular items and the trends, we want to predict the items most popular in the near future. We plan to examine the influence of context parameters for evolution of trending items. In addition we plan to investigate approaches for dynamically identifying topics and analyze recommender algorithms based on trending topics.

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 No. 610594.

References

1. D. J. Balling and J. Zawodny. *High Performance MySQL*. Safari Tech Books Online, 2004.
2. T. Brodt and F. Hopfgartner. Shedding light on a living lab: the CLEF NEWSREEL open recommendation platform. In *Proceedings of the 5th Information Interaction in Context Symposium*, pages 223–226. ACM, 2014.
3. S. Hohfeld and M. Kwiatkowski. Empfehlungssysteme aus informationswissenschaftlicher Sicht – State of the Art. *Information Wissenschaft und Praxis*, 58(5):265, 2007.
4. 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.
5. B. Kille, T. Brodt, T. Heintz, F. Hopfgartner, A. Lommatzsch, and J. Seiler. NEWSREEL 2014: Summary of the news recommendation evaluation lab. In *Working Notes for CLEF 2014 Conference*, pages 790–801, 2014. urn:nbn:de:0074-1180-0.
6. P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
7. A. Said, A. Bellogín, J. Lin, and A. de Vries. Do recommendations matter?: news recommendation in real life. In *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 237–240. ACM, 2014.
8. J. Traub, T. Rabl, F. Hueske, T. Rohrmann, and V. Markl. Die Apache Flink Plattform zur parallelen Analyse von Datenströmen und Stapeldaten. 2015.
9. S. Werner and A. Lommatzsch. Optimizing and Evaluating Stream-based News Recommendation Algorithms. In *Working Notes for CLEF 2014 Conference*, pages 813–824, 2014. urn:nbn:de:0074-1180-0.