Real-Time Recommendations in a Multi-Domain Environment

Emanuel Lacic
KTI
Graz University of Technology
Graz, Austria
elacic@know-center.at

ABSTRACT

Recommender systems are acknowledged as an essential instrument to support users in finding relevant information. However, adapting to different domain specific data models is a challenge, which many recommender frameworks neglect. Moreover, the advent of the big data era has posed the need for high scalability and real-time processing of frequent data updates, and thus, has brought new challenges for the recommender systems' research community. In this work, we show how different item, social and location data features can be utilized and supported to provide real-time recommendations. We further show how to process data updates online and capture user's real-time interest without recalculating recommendations. The presented recommendation framework provides a scalable and customizable architecture suited for providing real-time recommendations to multiple domains. We further investigate the impact of an increasing request load and show how the runtime can be decreased by scaling the framework.

Keywords

scalability; real-time recommendations; Apache Solr; multi-domain;

1. MOTIVATION

In the past decade, there has been a vast amount of research in the field of recommender systems. Most of that work focuses on developing novel approaches [22] and improving accuracy [16]. Thus, many well known methods are available, such as Content-Based Filtering [15], Collaborative Filtering [21] or Matrix Factorization [13], all having their unique strengths and weaknesses. These approaches are traditionally adapted and applied with the focus on a single domain model (e.g., marketplace, hotel, conference, etc.). However, to support a diverse set of domains is becoming an important issue for modern recommender systems [12].

In most domains, the prediction task is usually viewed as a two-dimensional problem which one needs to solve (e.g., utilizing user-item interactions). But nowadays it is not enough to support multiple domains on the basis of only one common data feature. With the arrival of the big data era, recommender systems are expected to analyze a lot of data, to support various data types and to handle streams of new data (i.e., volume, variety and velocity defining the Big Data problem). In such large-scale settings, traditional recommender systems usually analyze the data offline and update the generated model in regular time intervals. However, in many domains, choices made by users depend on factors which are susceptible to change anytime. Lets take a shopping mall for example, where a user triggers frequent indoor location updates via a smartphone application while moving through the mall. Employing an offline model update strategy that lasts hours or days may potentially miss the current location context of the user and fail to provide the right recommendations to match user's real-time demand. As a consequence, being able to capture user's real-time interests is gaining momentum and is currently of high demand [5, 16, 4].

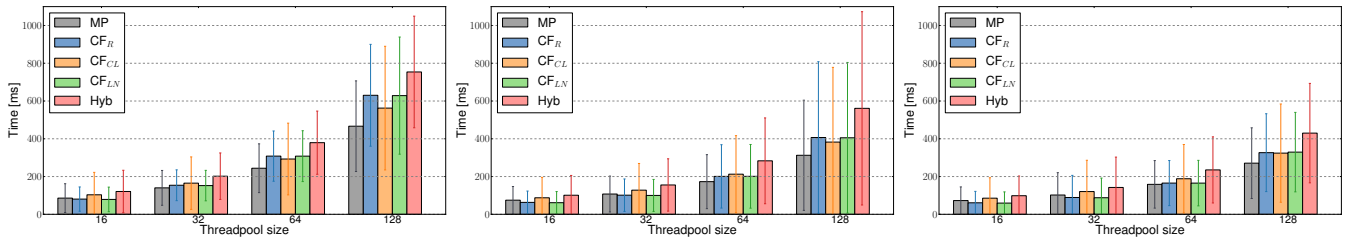
2. BACKGROUND

Most existing work, which focuses on real-time recommendations (e.g., Netflix [1], Microsoft [17], among others, e.g., [3, 23]), use offline batch processing frameworks like Apache Hadoop, Mahout or Spark. Other approaches use a relational database system to provide near real-time recommendations by querying the recommendations from a generated data model [19]. However, to capture user's real-time interest, streaming data needs to be processed online, thus needing to tackle the conflicting accuracy, real-time and big data requirements. For example, recent research from Huang et al. [5] and Chandramouli et al. [4] goes into that direction by utilizing a scalable Item-based Collaborative Filtering approach to provide real-time recommendations.

But, by focusing on the common user-item interactions, additional contextual information is usually neglected. As such, the research community has also looked into exploiting social or location data (e.g., [2, 13]). In doing so, personalized recommendations using Matrix Factorization dominate the literature. Jamali et al. [6] predicted ratings using a Matrix Factorization model that incorporates social relations. Ma et al. [13] improved both Mean Absolute Error and Root-Mean-Square Error by incorporating social information, using social regularization in two Matrix Factorization models. In general, Matrix Factorization based approaches need to be retrained, when the data changes. This tends to be time-consuming, especially in case of frequent data updates where it fails to capture user's real-time demand. Furthermore, empirical studies showed that a large number of factors are needed so that Matrix Factorization based approaches can deal with sparse data [18].

3. APPROACH AND METHODS

In this work, we are interested in finding out to what extent different data features (i.e., item, social or location) can be utilized or even be combined for real-time recommendation. To perform



(a) 1 processing node (b) 2 processing nodes (c) 4 processing nodes

Figure 1: Scalability experiment with five recommendation approaches (having the hybrid run the four approaches in parallel), making 325,005 independent recommendation requests to process. The exponentially increasing request loads (simulated by threadpools that continuously fire requests) are handled in three scenarios: (a) localy with only 1 processing node being deployed, (b) scaling the framework with 2 distributed nodes, and (c) having 4 distributed nodes to process the incoming recommendation requests.

this task, we rely on data crawled from the virtual world of SecondLife [1] and perform an extensive evaluation in terms of *nDCG* and *User Coverage* [7] of the different content- and network-based data features. The main reason for choosing SecondLife data over other sources are manifold, but mainly due to the fact that currently there are no other datasets available that comprise extensive item, social and location data of users at the same time. Building up on these results, the aim is to provide a general framework which can (1) process streaming data online while providing real-time recommendations, (2) support a multi-domain environment and the corresponding data features, and (3) provide a scalable architecture to cope with increasing request loads.

Recently, search engines have gained attention in the context of recommender systems [14]. While the results are promising, they do not provide explanations and evaluations of how such an approach would perform in a big data, nor in a real-time multi-domain environment. As such, the aim of this work is to proof the benefits of using search engines to support different data features while providing real-time recommendations. One issue, however, is that in this way scalability problems are only tackled on the data side of the domain. In order to truly be able to support multiple domains, a recommender framework is needed which can additionally (1) be customized with domain specific models and approaches, and (2) cope with an increasing request load a domain could experience. Using the already mentioned SecondLife dataset, but also a much larger Foursquare dataset [20], we simulate an increasing recommendation request load which such a framework needs to handle.

4. OUTCOMES

In [7] we showed to what extent different data features (derived from item, social and location data) can be utilized for recommending items, low-level and top-level categories. In our results, we showed that approaches which utilize social data features can outperform the ones based on item or location features in case of recommending items. In case of recommending categories, these differences get substantially smaller and even change in favor to item and location data. Moreover, our results suggests that combining the data sources should result into more robust recommendations, especially in cases of recommendation tasks on different levels of specialization (i.e., categories). In a similar fashion, we also showed in [10] that location data can especially be helpful in tackling cold-start users which have no interaction data whatsoever.

In [8], we proofed the benefits of using the search engine Apache Solr [2] to provide real-time recommendations. We showed that a recommender system is able to process data updates in real-time and immediately consider these updates (i.e., user's real-time interest) in the recommendation process without the need for recalculations. In [9] we also presented the first open-source recommender framework based on the Apache Solr search engine. But as previously mentioned, we considered the scalability issues only on the data side of a domain, and not within the framework (e.g., handling an increased request rate). For that purpose, we recently presented *ScaR* [11]. *ScaR* adopts the microservices architecture and was built with the focus on providing a scalable and customizable architecture suited for providing real-time recommendations to multiple domains. Different domains can run (and scale) the framework in isolated environments. The domain specific data features and recommendation approaches can be dynamically customized using a dedicated microservice which synchronizes the change to all domain-relevant nodes.

To demonstrate *ScaR's* scalability performance, Figure 1 reports a runtime experiment on the Foursquare dataset with an increasing number of request loads. As described in [11], we requested five different recommendation approaches for 65,001 users, making it 325,005 independent recommendations requests to process. We performed this experiment by simulating an increasing request rate (load) to the system, having 16, 32, 64 and 128 threads simultaneously requesting recommendations. These experiments were then repeated three times: first having 1 local processing node and then, scaling it to 2 and 4 distributed nodes. As seen, the local deployment has an exponential increase in the runtime as the load grows. Such behaviour is somewhat expected as the number of incoming recommendation requests cause a load spike and the processing threads consequently cause to much context switching. But, as we deploy additional nodes, we can see a significant decrease in the growth of the mean processing runtime when compared to the local deployment, which is crucial in cases when a maximal runtime needs to be guarantied.

5. PLAN AND TIMELINE

With respect to the future research workplan, the aim is to further look into feasible strategies to balance the trade-off between accuracy and runtime in a multi-domain environment. For a thesis conclusion, the idea is to find out how recent the utilized history data and the candidate recommendations need to be (i.e., by considering the exact time or a sliding window approach) in order to even better recommend user's real-time interest. This would not only lead to better accuracy but also to a better performance, as less data will need to be processed.

[1] http://secondlife.com/
[2] http://lucene.apache.org/solr/

6. REFERENCES

[1] X. Amatriain. Big & personal: Data and models behind netflix recommendations. In *Proc. of BigMine '13*.

[2] K. Bischoff. We love rock 'n' roll: Analyzing and predicting friendship links in last.fm. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, pages 47–56. ACM, 2012.

[3] S. Chan, T. Stone, K. P. Szeto, and K. H. Chan. Predictionio: a distributed machine learning server for practical software development. In *Proc. of CIKM '13*.

[4] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Streamrec: A real-time recommender system. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1243–1246, 2011.

[5] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 227–238, 2015.

[6] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 135–142. ACM, 2010.

[7] E. Lacic, D. Kowald, L. Eberhard, C. Trattner, D. Parra, and L. Marinho. Utilizing online social network and location-based data to recommend products and categories in online marketplaces. In *Mining, Modeling, and Recommending 'Things' in Social Media*, pages 96–115. Springer, 2015.

[8] E. Lacic, D. Kowald, D. Parra, M. Kahr, and C. Trattner. Towards a scalable social recommender engine for online marketplaces: The case of apache solr. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 817–822. International World Wide Web Conferences Steering Committee, 2014.

[9] E. Lacic, D. Kowald, and C. Trattner. Socrecm: A scalable social recommender engine for online marketplaces. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, HT '14, pages 308–310, 2014.

[10] E. Lacic, D. Kowald, M. Traub, G. Luzhnica, J. Simon, and E. Lex. Tackling cold-start users in recommender systems with indoor positioning systems.

[11] E. Lacic, M. Traub, D. Kowald, and E. Lex. Scar: Towards a real-time recommender framework following the microservices architecture.

[12] Q. Liu and D. R. Karger. Kibitz: End-to-end recommendation system builder. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015.

[13] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 287–296. ACM, 2011.

[14] D. Parra, P. Brusilovsky, and C. Trattner. User controllability in an hybrid talk recommender system. In *Proceedings of the ACM 2014 International Conference on Intelligent User Interfaces*, IUI '14, pages 305–308. ACM, 2014.

[15] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[16] C. Rana and S. K. Jain. A study of the dynamic features of recommender systems. *Artificial Intelligence Review*, 43(1):141–153, 2015.

[17] R. Ronen, N. Koenigstein, E. Ziklik, M. Sitruk, R. Yaari, and N. Haiby-Weiss. Sage: Recommender engine as a cloud service. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 475–476, 2013.

[18] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887. ACM, 2008.

[19] M. Sarwat, J. Avery, and M. F. Mokbel. Recdb in action: Recommendation made easy in relational databases. *Proc. VLDB Endow.*, 6(12):1242–1245, Aug. 2013.

[20] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Lars*: An efficient and scalable location-aware recommender system. *IEEE Trans. on Knowl. and Data Eng.*, 26(6):1384–1399, June 2014.

[21] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. The adaptive web. chapter Collaborative Filtering Recommender Systems, pages 291–324. Springer-Verlag, 2007.

[22] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[23] S. G. Walunj and K. Sadafale. An online recommendation system for e-commerce based on apache mahout framework. In *Proceedings of the 2013 annual conference on Computers and people research*, pages 153–158. ACM, 2013.