Scalable Personalized Recommender System

Adam LIESKOVSKÝ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
adamliesko@gmail.com

Abstract. Requirements and priorities for modern recommender systems grow and change hand in hand with the continuous growth of Internet users. In this paper we focus on aspects such as scalability, flexibility and speed of recommender systems, which nowadays play an important role. We propose novel hybrid recommender system, which utilizes user behaviour and context, items' content and items' popularity and recency. We evaluate our approach in the domain of news articles, processing streaming data and computing recommendations online, in real-time. Conducted evaluations showed the benefits of using article trendiness as a strong component in the process of computing recommendations.

1 Introduction and related work

Information overload is nowadays a serious issue on Web. Users don't have the time, neither the resources or skills to manually search and filter for items which they seek. Personalized recommendations are great way to reduce and eliminate this problem. Furthermore, it helps users to discover new items, which they wouldn't be able to find by the means of standard navigation. Other than that, personalized recommendations help with increasing the commercial revenue as well. Therefore, it is a fairly popular topic not only in the academic world, but also between the tech giants like LinkedIn [1], Netflix [2] or Amazon [3], where recommendations play a key role in the user experience.

Another prevailing problem, which appears with the enormous traffic and number of users the popular sites have to serve, is the aspect of scalability. Recently, methods like distributed systems, GPU matrix computations or client-server computation distribution [4] have enabled for real-time processing of streaming data on a large scale.

Majority of established approaches for personalized recommendations fall into the categories of collaborative filtering and content based recommendations. Each of these approaches comes with it's downsides (e.g., cold start problem of a new item or new user, long tail problem), which hybrid recommender systems usually help to eliminate [5]. For long, data processing and computations at large scale have been limited by existing technology or software. With the introduction of map-reduce computation model, now also available through in-memory

* Master degree study programme in field: Software Engineering
Supervisor: Dr. Michal Kompan, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

computation, or the massive growth of NoSQL databases, recommender systems are nowadays able to achieve the required scalability and efficiency needed for the Web scale.

Nowadays, collaborative filtering approaches, for example using a scalable k-NN algorithm [6] or matrix factorization [7, 8], present the state of the art method for recommender systems. Most successful and widely used are matrix factorization processing user-item association matrix [9].

In [9] Shi et al. present a comprehensive survey of the state-of-the-art collaborative filtering and latest challenges in the domain of recommender systems. They identified the incorporation of social recommendations and additional user interaction data (e.g., context), cross domain and group collaborative filtering as the arising key challenges to the future of collaborative filtering.

2 Scalable and flexible recommendation approach

In this paper, we propose a scalable hybrid recommender system, with focus on a fast response time, scalability and flexibility in regard to the system workload and available resources. Abstractly, it can be separated into two subparts, first consisting of a real-time online computation of recommendations, the second of computing offline user models. Our hybrid approach can be classified as a meta-level, mixed or weighted, depending on the form of final recommendation aggregation. In the following sections, we describe respective recommender modules for hybrid recommender system in the domain of news articles, as outlined on the Figure 1.

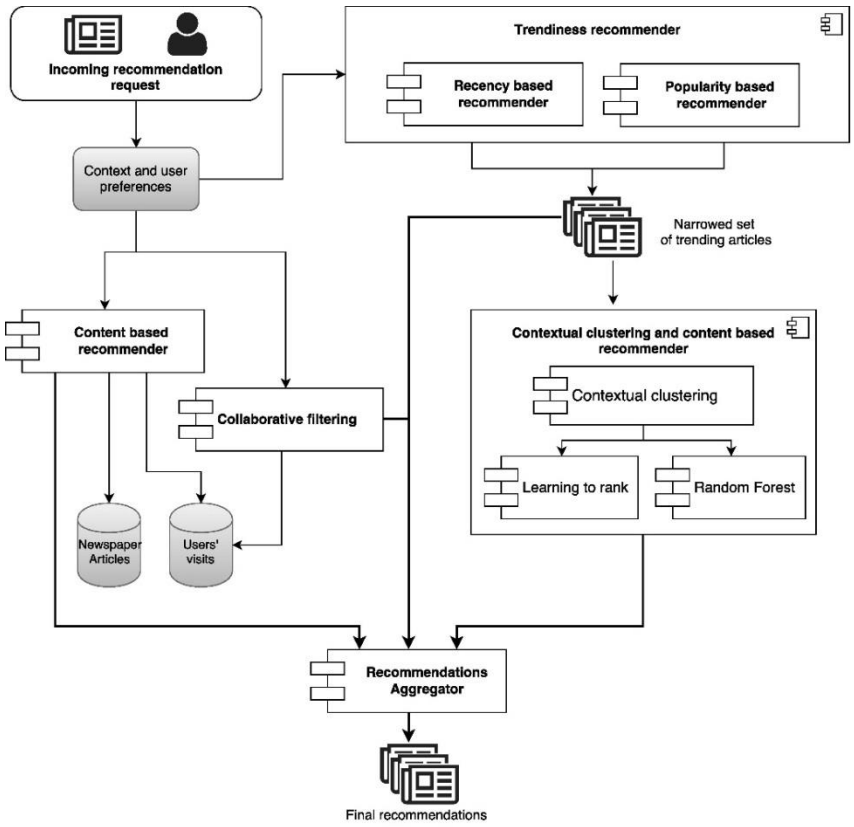

Figure 1. Proposed scalable hybrid recommender approach.

2.1 Content based recommendation

Research in natural language processing (NLP) and its approaches to content similarity estimation using *LDA* (*Latent Dirichlet allocation*) [10] or *LSA* (*Latent semantic analysis*) [11] have recently become fairly popular and successful. If combined with SVD (Singular value decomposition), these algorithms provide a scalable way to cluster items and infer topics based on their content. In our work we opted for a simpler and faster approach, utilizing direct content similarity estimation between item pairs.

On top of static item content comparison, we incorporate sentiment and relevance of extracted keywords or entities into the recommendation process. This information can be extracted from the content of items, e.g., in the domain of news articles we use article title and text as source of keyword extraction. This way, we can identify user's probable attitude towards certain topics (election parties, war etc.), entities (e.g., Steve Jobs, England) or extracted keywords. Given this information, we can narrow the recommendations to the users with items containing similar sentiment or attitude in general towards certain entities or topics, in which the user has previously shown interest.

2.2 Collaborative filtering

Since Netflix prize [8], matrix factorization has become a state-of-the-art method for computing recommendations with collaborative filtering. We chose alternating least squares (*ALS*) algorithm, which can compute latent factors describing users and items from user-item association matrix. ALS algorithm provides us with an option to specify a weight to a user-item interaction, if it turns out that it is desirable to differentiate between user clicking on a certain recommendation or a user naturally visiting an item. More specifically we are using *ALS-WR* approach, as described in [12]. *ALS-WR* uses a normalization parameter $lambda$, which tackles scalability and sparsity issues in matrix factorization. One of the nice properties of *ALS-WR* is that the accuracy of the method monotonically improves with the number of algorithm iterations.

ALS model is computed periodically in the background with latest subset of user-item interactions narrowed by their occurrence time. Computation of a model is distributable and we can easily set number of iterations based on the current system workload. The periods between model re-computation can be also adjusted according to the workload, which also adds to the flexibility property of our proposed approach.

2.3 Popularity and recency of items

Popularity and recency are two aspects of items, that we combined together to form a so-called trendiness recommender. We model popular and recently created or updated items in certain time intervals. In addition to global counters, we monitor the popularity and recency of articles in the subsets of items, as per their categorization by content. For popularity, we use integer counters of item visits occurrences in the respective time interval. For recency aspect, we propose a more sophisticated approach. Our goal is to be able to assign the same value of recency to an article published 10 minutes ago, as to an article published 1 hour ago (example thresholds). We are able to achieve it, by using Equation 1[1], which creates a gauss like figure of recency in respect to the times of item creation and their last update. $Decay$ and $scale$ parameters enables the fine grained time decay property of recency recommender module.

$$recency(article) = w_1 * timePenalty(\text{p}ublished_{at}) - w_2 * timePenalty(updated_{at}) \quad (1)$$

$$timePenalty(time) = exp(-\frac{max(0,\ |time - \ published_{at}| - penaltyStart)^2}{2(-\frac{scale^{\ 2}}{2} * \log(decay))}$$

[1] https://www.elastic.co/guide/en/elasticsearch/guide/current/decay-functions.html

We either use the subsets of our so-called trending articles directly as top-n recommendations, or push them further down our method, where they serve as an input to some of the more computational expensive methods for recommendation generation.

Our reasoning behind this approach is strongly tied with the current trends of navigation on the Web. On majority of sites you can find lists of most popular items, new and current items, that narrows the information and navigation space for users. This especially applies to the domain of newspaper articles, where users are visiting news portals in order to find new and current information. These items could stand a higher chance of actually being visited by user, because on average, majority of users tend to be interested in some of the most popular and recent topics. We proved this assumption correct in our evaluations.

2.4 Contextual recommendations and clustering

Nowadays, context is getting ubiquitous and as seen in other works [13], it is a great source of information, which we can use for clustering user recommendation requests. In our method, we use contextual data like geo-location, gender, estimated age or salary, ISP and similar properties. By using streaming *k-means* algorithm, specifically the *kmeans++* implementation [14], we are able to perform clustering in real-time, while maintaining up-to-date representation of the clusters. Each recommendation request is assigned to a cluster, based on the properties of the request context. Inside of each of these cluster, we hold machine learning models used for classification and ranking items of items. Classification outcome and rank of items depend on the relevance and confidence of respective items being clicked or visited by user.

For classifying and ranking items we use two scalable and parallelizable approaches: decision trees (*Random forest*) and learning to rank (*Listnet*). Learning to rank and *Listnet* algorithm specifically, have already been proved [15, 16] to be a suitable choice for large scale machine learning model construction. In our approach we are using them as estimators of user's probability of actually clicking on a recommendation. Input to machine learning models is a narrowed subset of trending articles and outcome is a set of top-n recommendations with respective confidence weights, based on the learned model for a specific cluster, to which a recommendation request was assigned depending on contextual data. These models are updated and reconstructed periodically in the background with the latest data.

2.5 Aggregation of recommendations

Our proposed approach provides us with variety of possible combinations and aggregation of recommender modules within our hybrid system. The aggregation process is exactly the place, where the flexibility and adaptive properties of our approach can show off. Under ideal conditions and state of a recommender system (low workload), our approach uses all of the described recommender modules. Trendiness module acts as a meta-level recommender, which generates input subsets for other recommender modules (e.g., contextual). Collaborative filtering and content based recommendations, which work with constructed user models provides recommendations based upon the user's behaviour and user models. Recommendations from the respective recommender are then grouped and their confidence is summed up, along with the weight of each of recommender modules.

Our approach eliminates cold start issue of a new user in the system. Even for an unknown or new user, we still have on-line contextual data and information about the currently visited item. Altogether with popularity and recency information, we can identify and recommend relevant articles to the user. Incoming request is assigned to a cluster by the on-line information and a machine learning model (learning to rank, random forest) ranks subset of trending articles according to the predicted cluster and user. As a response to the recommendation request, we present top-n recommendations to the user.

3 Evaluation

As explained in the previous section, one of the key points of our proposed approach is the focus on popularity and recentness of articles. In order to prove this hypothesis, we evaluated the effect of adding popularity and recency aspects to a simple k-nn recommender. In the k-nn algorithm, nearest neighbors were chosen by the co-occurrence factor based on item to item similarity [3]. Similarities were computed using log likelihood ratio of impression occurrences. Evaluations were performed on an offline dataset, consisting of ~12 million impressions gathered from slovak online newspaper publisher SME.SK during a week in October 2009.

As shown in the Table 1, we were able to improve precision@10 and NDCG@10 metrics by over 100%, when we added popularity and recency aspects into the recommendations process. With items impressions, we have only binary scale (seen/not seen) which is unable to fully utilize NDCG properties. This effect can be seen also by the similar improvement rate of both of the used metrics.

Table 1. Evaluation of article trendiness increase in precision and NDCG metrics.

No. of nn	Without popularity and recency		With popularity and recency (exp. boosting)			
	precision @10	NDCG@10	precision @10		NDCG@10	
2	0.0086	0.0410	0.0179	**+ 108%**	0.0803	**+ 96%**
3	0.0093	0.0411	0.0190	**+ 104%**	0.0792	**+ 93%**
5	0.0114	0.0495	0.0222	**+ 95%**	0.0906	**+ 83%**
7	0.0104	0.0453	0.0228	**+ 19%**	0.0913	**+ 102%**
10	0.0108	0.0446	0.0243	**+ 125%**	0.0952	**+ 113%**
15	0.0107	0.0414	0.0229	**+ 114%**	0.0860	**+ 108%**
20	0.0109	0.0418	0.0229	**+ 110%**	0.0840	**+ 101%**
30	0.0103	0.0412	0.0225	**+ 118%**	0.0809	**+ 96%**

4 Conclusions and future work

In this paper, we have proposed hybrid recommender system with focus on the scalable and extensible architecture, flexibility of the system and ability to adapt to the system workload. Generated recommendations are computed with the use of up-to-date information, whether it is users behaviour and context or item content. As a unique feature of our approach, we highlight recommendation requests clustering based on the user's context, which reduces information space and enables for frequent and continuous updating of underlying machine learning models used for generating recommendations.

So far, we have performed only a basic and partial evaluation of our proposed hybrid recommender approach. We yet have to evaluate usage of clustering of incoming requests, scalability and flexibility of our approach. We plan to perform online evaluations with streaming data, where we will focus on CTR metric. Furthermore, we will perform rigorous and repeatable evaluations concerning scalability and accuracy of our proposed recommender system with prepared datasets created from acquired streaming data.

In terms of extending our proposed method, there are several opportunities for improvement and experimenting. Inclusion of information gathered from social data could serve as a great addition to the user model. For example, if we had performed latent topic analysis (LDA/LSA), we could map these identified topics to not only topics identified inside the items content, but also to the user's interests, as expressed on their social network profiles.

Acknowledgment: This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0774/16.

References

[1] R. Sumbaly, J. Kreps, and S. Shah, "The 'Big Data' Ecosystem at LinkedIn," *Int. Conf. Manag. Data (SIGMOD 2013)*, pp. 1–10, 2013.

[2] X. Amatriain, "Big & Personal: data and models behind Netflix recommendations," *Proc. 2nd Int. Work. Big Data*, pp. 1–6, 2013.

[3] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations Item-to-Item Collaborative Filtering," *IEEE Internet Computing.*, vol. 7, no. 1, pp. 76–80, 2003.

[4] A. Boutet, D. Frey, and A. Kermarrec, "HyRec: Leveraging Browsers for Scalable Recommenders Categories and Subject Descriptors," pp. 85–96, 2014.

[5] V. Vekariya and G. R. Kulkarni, "Hybrid recommender systems: Survey and experiments," *2012 2nd Int. Conf. Digit. Inf. Commun. Technol. its Appl. DICTAP 2012*, pp. 469–473, 2012.

[6] X. Yang, Z. Zhang, and K. Wang, "Scalable collaborative filtering using incremental update and local link prediction," *Proc. 21st ACM Int. Conf. Inf. Knowl. Manag. - CIKM '12*, p. 2371, 2012.

[7] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *J. Mach. Learn. Res.*, vol. 10, no. 6 , pp. 623–656, 2009.

[8] R. M. Bell and Y. Koren, "Lessons from the Netflix prize challenge," *ACM SIGKDD Explor. Newsl.*, vol. 9, no. 2, p. 75, 2007.

[9] Y. U. Shi, M. Larson, and A. Hanjalic, "Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–45, 2014.

[10] B. Chen, "fLDA : Matrix Factorization through Latent Dirichlet Allocation," *New York*, pp. 91–100, 2010.

[11] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Boston, MA: Springer US, 2011.

[12] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Algorithmic Aspects in Information and Management," Proc. of the 4th Int. Conf., AAIM 2008, Shanghai, China,R. Fleischer and J. Xu, Eds. Berlin, Springer Berlin Heidelberg, 2008, pp. 337–348.

[13] C. Palmisano, A. Tuzhilin, and M. Gorgoglione, "Using context to improve predictive modeling of customers in personalization applications," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1535–1549, 2008.

[14] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable K-Means ++," *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 622–633, 2012.

[15] S. Shukla, M. Lease, and A. Tewari, "Parallelizing ListNet training using spark," *Proc. 35th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '12*, p. 1127, 2012.

[16] T. Niek, "Scaling Learning to Rank to Big Data Using MapReduce to Parallelise Learning to Rank," University of Twente, Twente, Netherlands, 2014.