BBC

Research & Development

White Paper

WHP 274

January 2014

Design and Evaluation of a Client-side Recommender System

Chris Newell and Libby Miller

BRITISH BROADCASTING CORPORATION

White Paper WHP 274

Design and Evaluation of a Client-side Recommender System

Chris Newell and Libby Miller

Abstract

Most recommender systems found on the web are server-based and centralised. However, it can be difficult to maintain the responsiveness with this approach when there are large numbers of concurrent users. This paper describes an alternative distributed approach where major parts of the recommender system are implemented in scripts run by the user's client system. This client side approach ensures consistent responsiveness and supports the possibility of privacy-preserving recommender systems. The advantages and limitations of the approach are discussed, suitable algorithms are proposed and an open source implementation is described with example applications.

This document is an extended version of a demonstration paper presented at RecSys 2013 [1].

Additional key words: k-NN

White Papers are distributed freely on request.

Authorisation of the Chief Scientist or General Manager is required for publication.

White Paper WHP 274

Design and Evaluation of a Client-side Recommender System

Chris Newell and Libby Miller

1 Introduction

Our previous work involving a user centric study of recommender systems [2] indicated that users valued the ability to explicitly influence the behaviour of recommender systems. This encouraged us to pursue research on conversational recommender systems where users are able to steer the recommendations to suit their current mood and interests. Supporting this degree of interaction increases the work required by the recommender system. However, to provide a satisfactory user experience it needs to remain responsive. This led us to explore the feasibility of a client-side approach which could provide low latency whilst supporting many concurrent users.

Previous work adopting a client-side approach has primarily employed content-based filtering using locally recorded user preference data [3, 4]. In our alternative approach the client utilizes a downloaded recommender model which allows the use of either content-based or collaborative filtering.

The flowchart in Figure 1 shows the sequence of tasks required in a typical recommender system. In most web-based systems all tasks except the presentation task are implemented by the server.

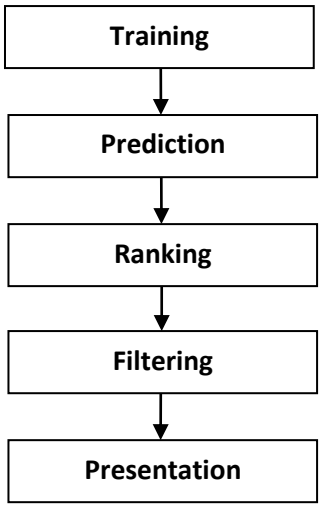

Figure 1. Recommender system tasks

This has two consequences:

1. The processing load on the server rises with the number of concurrent users.
2. The responsiveness of the system is limited by the round-trip delay between the client and the server.

In the proposed client-side approach all tasks except the training task are implemented on the client system using two components downloaded from the server:

1. A recommender model, which is trained and rebuilt at periodic intervals by the server.
2. A recommender engine which implements prediction, ranking and filtering.

Since both of these components are static files, which can be cached, the approach is easily scalable to large numbers of users. The recommender system is also easier to manage and deploy as the complexity and processing requirements of the server are reduced. The responsiveness becomes independent of the load on the server and the round-trip delay to the server is eliminated.

This client-side approach raises the possibility of privacy-preserving recommender systems where the user can choose to retain all personal data on the client. It also allows personalised recommendations to be provided in unidirectional broadcast systems, where no return path is available.

In the remainder of this white paper we discuss the potential scope of the client-side approach and suggest suitable algorithms. We then present an example implementation and some applications with quantitative and qualitative evaluations.

2 Potential Scope

Although it is possible to conceive of client-side recommender systems where some personal elements are delivered by the server (e.g. user preference data or user factors) this paper will consider the case where only non-personalised data is required. However, this doesn't preclude the possibility that some user data is stored on the server and used for training purposes.

Theoretically, any algorithm which supports the incremental addition of new users and their feedback would be suitable. However, the client-side approach is only practical if the recommender model is of a suitable size for clients to download. This may prohibit the use of some algorithms or prevent their use in applications where there are a large number of recommendable items, since models generally scale with the number of items.

Consideration must also be given to the fact that the recommender model would be in the public domain and whether it contains any sensitive data.

2.1 Memory-based Algorithms

Memory-based algorithms such as Item-based k-nearest neighbor (k-NN) algorithms [5] are practical candidates for this approach as the required client-side components are simple to implement.

Prediction can be done on the client-system using the user's locally recorded user feedback data and a downloaded k-NN model. The justification for not retraining the model, in collaborative filtering variants, is that the absence of a single user's feedback is unlikely to have a significant effect on a model built using a large number of users.

2.2 Model-based Algorithms

Model-based algorithms which support the incremental addition of new users and feedback are also suitable candidates for the client-side approach (e.g. BPR-MF [6]). In this case the server need only deliver the item factors from the model and a set of user factors can be created on the client system by incremental training using locally recorded feedback. The factors can then be used for prediction in the normal way.

3 Example Implementation

The context of our work is an online TV and radio catch-up service where programmes are available on demand for 7 days after broadcast. In this application the number of programmes available at any one time is relatively small so the client-side approach is very practical.

For our initial implementation we have focused on Item-based k-NN algorithms since these provide reasonable performance even for users with low amounts of feedback. The recommender model in these algorithms is a sparse correlation matrix which lists the nearest neighbors for each item (and their weights for some algorithms) and the overall size of the model scales linearly with the number of items.

The theoretical performance of a number of these algorithms was evaluated in our application context using the MyMediaLite Recommender System Library [7]. Positive-only training data in the form of user-programme viewings was obtained from our media server logs for all the programmes available at a specific time. All viewings that followed, up to the end of each programme's 7-day availability window, were used as test data. To avoid false positives, short viewings lasting less than a pre-set threshold duration were ignored, based on the observation that the drop-off rate of viewings is initially high but then falls to a lower, more consistent rate. The characteristics of the datasets are listed in Table 1.

	Training data	Test data
Users	3,056,160	1,289,779
Programmes	729	711
Viewings	14,113,357	4,048,830

Table 1. Characteristics of the evaluation data

Predicted rankings derived using the training data were compared against the test data to determine the Area under the ROC (Receiver Operating Characteristic) curve (AUC) ranking accuracy [8] for several collaborative filtering and content-based algorithms. All the algorithms used Cosine Similarity for correlation.

The collaborative filtering algorithms included Item k-NN with (WI) and without (I) using the nearest neighbor weights for prediction. The content-based algorithms included Item Attribute k-NN with (WIA) and without (IA) using the nearest neighbor weights for prediction. The content-based algorithms used binary programme attributes such as genre, format, people, locations and subjects as used in our online programme guide.

The evaluation results are shown in Figure 2. It can be seen that the weighted versions of the algorithms achieve better performance particularly for high values of k. However, this is at the expense of increased model size, since they require a weight value for each nearest neighbor.

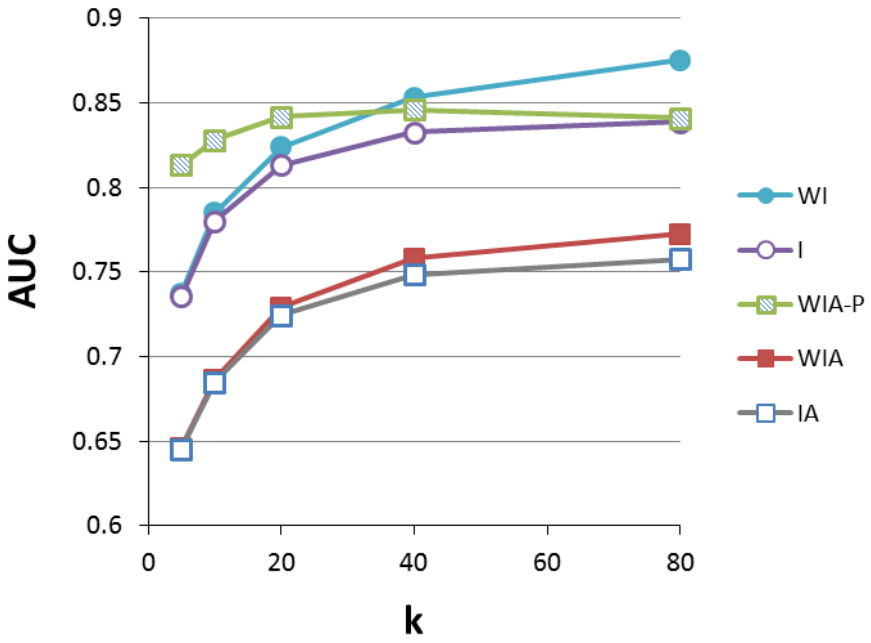

Figure 2. AUC against k value for a selection of Item based k-NN algorithms

When using the content-based algorithms it was observed that many programmes would share the same discrete prediction values because of the limited set of binary attributes available to compare between programmes. There was therefore no way to rank these programmes with respect to each other and the qualitative and subjective performance of these algorithms was found to be poor. An effective solution was to rank each subset of the programmes, which shared the same prediction value, by programme popularity (i.e. total number of viewings). The resulting hybrid Weighted Item Attribute k-NN and Popularity (WIA-P) algorithm achieved a significantly better theoretical and subjective performance.

On the basis of these results we selected the WI algorithm and the hybrid WIA-P algorithm for evaluation in our prototype implementation. We choose a k value of 20 as this gave a good compromise between performance and the model size, which rises linearly with k.

In our prototype implementation the recommender model is embedded in a JavaScript file. This file also carries some basic metadata for each programme (e.g. title, locator and genre) which is provided for filtering and presentation purposes. The weighted model size is typically around 120 KB including the metadata. However the size could be reduced substantially by using a binary format.

The recommender engine is also implemented in JavaScript. It accepts positive and negative binary preferences expressed by the user, as this matches the requirements of our applications but it could be adapted to other rating schemes. When no user feedback is available the rankings of programmes is determined by a pre-set order in the model file. In our implementation this order is determined by programme popularity but it could be set editorially to promote specific programmes. The recommendations can be filtered by genre or media type (i.e. TV or radio) and are typically produced in around 20 milliseconds.

In subjective evaluations the collaborative filtering WI algorithm appeared to produce more interesting recommendations with greater perceived variety than the content-based WIA-P algorithm, which has a tendency to recommend programmes in one or two predominant genres. For this reason we introduced a genre-diversity post-filter for this algorithm which adjusts the rankings to ensure that the same genre does not occur more than once in three consecutive recommendations.

An open source implementation of this recommender engine is available with guidance and tools for producing the required model files[1].

4 Example Applications

Two prototype applications have been developed using the client-side recommender. In both cases all user feedback is captured during each individual user session. However, locally stored user feedback accumulated over multiple sessions could also be used.

4.1 Drag and Drop Recommender

The first application is a TV and radio programme recommender system designed for tablets and personal computers[2]. It provides a simple drag-and-drop user interface as shown in Figure 3.

Users are presented with a scrollable list of programme recommendations which they can filter by genre and media type. The initial ordering of the list is determined by the overall popularity of each programme. The user is able to express explicit preferences by dragging individual programmes into “like” and “dislike” boxes positioned below. These preferences are immediately used by the client-side recommender to re-rank the programmes and refresh the recommendation list.

Initially, the application was implemented using a server-based recommender system and there was a noticeable and variable lag between each user input and the refresh process. With the client-side approach the delay is imperceptible.

[1] http://github.com/bbcrd/clientside-recommender
[2] http://sibyl.prototyping.bbc.co.uk/

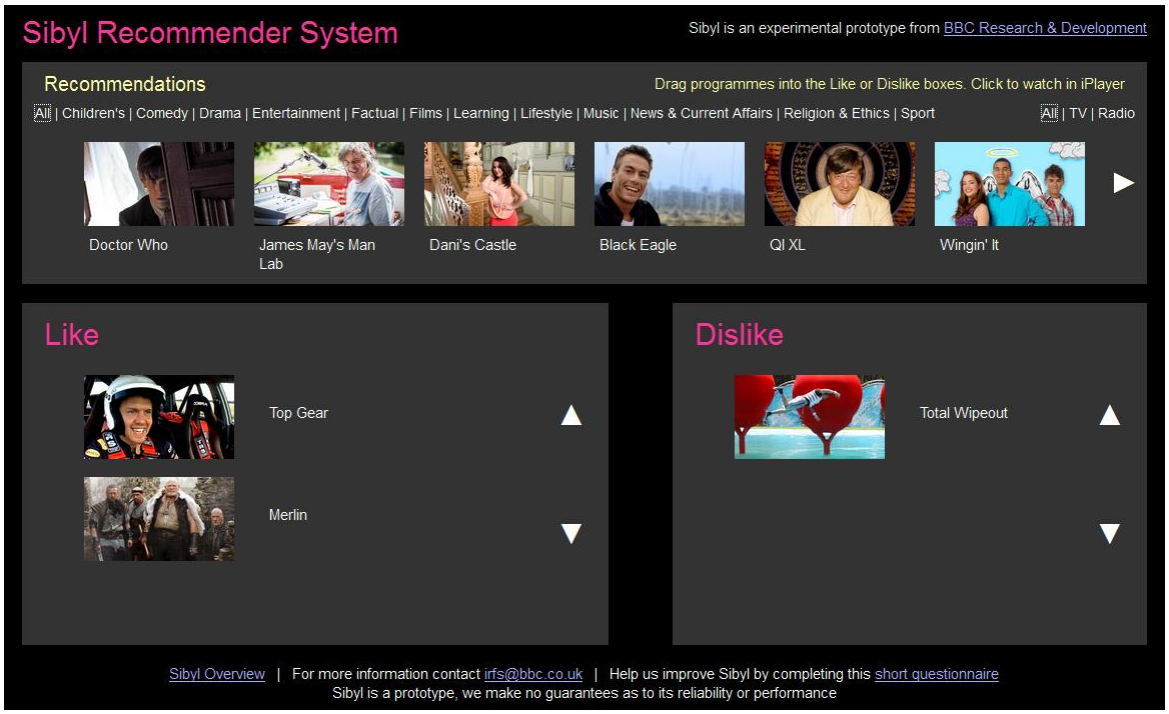

Figure 3. Drag-and-drop recommender system

The application was evaluated in user trials initially using the hybrid WIA-P algorithm (40 users) and more recently using the WI algorithm (22 users). The users were asked whether the recommender provides good recommendations using a 5-level Likert scale. The percentage of users giving each response is shown in Figure 4 for both algorithms. It can be seen that there is a reasonable consensus that the recommender is producing good recommendations for both algorithms but this consensus is stronger for the WI algorithm, with no users disagreeing. We considered the performance of both algorithms to be satisfactory for this application. When asked 80% of the users agreed that the recommender system provided a useful service.

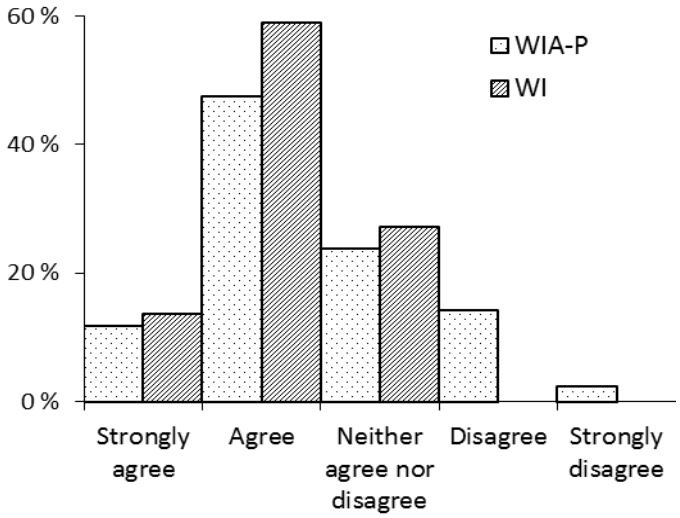

Figure 4. User responses to the statement that the recommender provides good recommendations

4.2 ViSTA-TV Clip-based Recommender

The second application is a TV programme recommender system which provides a continuous sequence of programme clips. The user can either skip or add each programme to their playlist for later consumption. These two user actions are interpreted as negative and positive user feedback respectively. The recommender engine uses this implicit user feedback to update the ranking of the clips and select the next clip to be presented. The mobile version of this application is shown in Figure 5 and will be evaluated in a future user trial.

Figure 5. Clip-based recommender system

In the present implementation the programme clips are streamed by the server. However, the clips could also be downloaded, allowing the application to function when the user is offline. This offline capability would only be possible with a client-side recommender.

This application was developed within the ViSTA-TV EU-funded collaborative research project[3]. The main goal of the project is to develop a stream processing engine which is able to complete a real-time analysis of IPTV viewing data and media content. The statistical viewing data and extracted media features will be used to provide recommender models at an earlier and more timely stage than is feasible with a non-real time approach.

5 Conclusions

An alternative, client-side approach to the implementation of web-based recommender systems has been presented which can be used to create highly responsive and easily scalable systems. The approach has been shown to be effective in conversational recommender systems, which require a high degree of interaction with the user and enables the possibility of privacy-preserving recommender systems. The approach has been shown to work effectively using Item-based k-NN algorithms but theoretically could also be applied using model-based algorithms.

6 Acknowledgements

The authors would like to thank Anthony Onumonu, Dan Nuttall and Andrew Wood for their design and development work and gratefully acknowledge the co-funding of the most recent part of this work through the European Commission FP7 project ViSTV-TV under grant agreement no. 269126.

[3] http://vista-tv.eu

7 References

[1] Newell, C and Miller, E. 2013. Design and Evaluation of a Client-side Recommender System. *Proceedings of the 7th ACM conference on Recommender Systems*, RecSys 2013. 473-474. DOI= http://dx.doi.org/10.1145/2507157.2508220

[2] Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H. and Newell, C. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*. 22, 4-5 (Oct. 2012), 441-504. DOI= http://dx.doi.org/10.1007/s11257-011-9118-4

[3] Bradley, K., Rafter, R. and Smyth, B. 2000. Case-Based User Profiling for Content Personalisation. *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, AH 2000. 62-72.

[4] Lu, H., Luo, Q. and Shun, Y. K. 2003. Extending a Web Browser with Client-Side Mining. *Proceedings of Web Technologies and Applications*, APWeb 2003. 166-177.

[5] Dietmar, J., Zanker, M., Felfernig, A. and Friedrich, G. 2011. *Recommender Systems: An Introduction*. Cambridge University Press, Cambridge, UK, 18-20.

[6] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intell*igence, UAI 2009.

[7] Gantner, Z., Rendle, S., Freudenthaler, C. and Schmidt-Thieme, L. 2011. MyMediaLite: a free recommender system library. *Proceedings of the 5th ACM conference on Recommender Systems*, RecSys 2011. 305-308. DOI= http://dx.doi.org/10.1145/2043932.2043989

[8] Manning, C. D., Raghavan, P. and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 154-163.