An Adaptive Hybrid Recommender System that Learns Domain Dynamics

Fatih Aksel and Ayşenur Birtürk

Department of Computer Engineering, METU
{fatih.aksel,birturk}@ceng.metu.edu.tr

Abstract. Traditional hybrid recommender systems typically follow a manually created fixed prediction strategy in their decision making process. Experts usually design these static strategies as fixed combinations of different techniques. However, people's tastes and desires are temporary and the gradually evolve. Moreover, each domain has unique characteristics, trends and unique user interests. Recent research has mostly focused on static hybridization schemes which do not change at runtime. In this paper, we describe an adaptive hybrid recommender system, called AdaRec that modifies its attached prediction strategy at runtime according to the performance of prediction techniques. Our approach to this problem is to use adaptive prediction strategies. Experiment results with datasets show that our system outperforms naive hybrid recommender.

1 Introduction

With the rapid development of the recommender system technology, a number of recommender engines have been developed for various application domains. Content-based and collaborative filtering are the two major recommendation techniques that have come to dominate the current recommender system. Content-based recommender system uses the descriptions about the content of the items (such as meta-data of the item), whereas collaborative filtering system tries to identify users whose tastes are similar and recommends items they like. Every recommendation approach has its own strengths and weaknesses. Hybrid recommender systems have been proposed to gain better results with fewer drawbacks [3].

Most of the recommender system implementations focuses on hybrid systems that use mixture of recommendation approaches [3]. This helps to avoid certain limitations of content-based and collaborative filtering systems. Previous research on hybrid recommender system has mostly focused on static hybridization approaches (strategy) that do not change their hybridization behavior at runtime. Fixed strategy may be suboptimal for dynamic domains&user behaviors. Moreover they are unable to adapt to domain drifts. Since people's tastes and desires are transient and subject to change, a good recommender engine should deal with changing consumer preferences.

In this paper, we describe an *Adaptive Hybrid Recommender System*, called AdaRec, that modifies its switching strategy according to the performance of

prediction techniques. Our hybrid recommender approach uses adaptive *prediction strategies* that determine which *prediction techniques* (algorithms) should be used at the moment an actual prediction is required. Initially we used manually created rule-based strategies which are static. These static hybridization schemes have drawbacks. They require expert knowledge and they are unable to adapt to emerging trends in the domain. We now focus on prediction strategies that learn by themselves.

The paper is organized as follows. Related work is described in Sec. 2. In Sec. 3, we present the adaptive prediction strategy model for hybrid recommenders. We then describe our experimental recommender systems' architecture & learning module that dynamically adjusts recommendation strategy in response to the changes in domain. An initial evaluation of our approach, based on MovieLens dataset, is presented in Sec. 5.

2 Related Work

There have been several research efforts to combine different recommendation technologies. The BellKor system [2], statically combines weighted linear combination of more than a hundred collaborative filtering engines. The system uses the model based approach that first learns a statistical model in an offline fashion, and then uses it to make predictions and generate recommendations. The weights are learned by using a linear regression on outputs of the engine. The STREAM [1] recommender system, which can be thought of as a special case of the BellKor system, classifies the recommender engines in two levels: called level-1 and level-2 predictors. The hybrid STREAM system uses runtime metrics to learn next level predictors by linear regression. However combining many engines in to levels cause performance problems at run-time. Our appraoch combines different algorithms on a single hybrid engine with an adaptive strategy.

In our system, we chose the Duine Framework[1], which is an open-source hybrid recommendation system [6]. The Duine framework allows users to develop their own prediction engines. The framework contains a set of recommendation techniques, ways to combine these techniques into recommendation strategies, a profile manager, and it allows users to add their own recommender algorithm to the system. It uses switching hybridization method in the selection of prediction techniques.

3 Our Approach

Hybrid recommendation systems combine multiple algorithms and define a *switching behavior* (strategy) among them. This strategy decides which technique to choose under what circumstances for a given prediction request. Recommender system's behavior is directly influenced by the prediction strategy.

[1] `http://duineframework.org/`

Most of the currently available personalized information systems and research into these systems focus on the use of a single selection technique or a fixed combination of techniques [6],[4]. However, domains are dynamic environments. Users are continuously interacting with domain, new concepts and trends emerge each day. Therefore, user interests change dynamically over time. It does not seem possible to adapt trends by using a static approach. Instead of using predefined static methods for hybridization, it may be more effective to use adaptive methods in dynamic domains.

In our approach we apply an instance based, supervised learning scheme, which learns through predictions performance results. The main idea of our system is to re-design the hybrid system's switching behavior at runtime according to the performance results of the algorithms. In AdaRec System, we implement prediction strategy learning module, which produces a strategy by using its attached learning technique. Learning module initializes prediction engine according to the specified Machine Learning (ML) technique. Different instance-based ML algorithms could be attached to our experimental design.

A prediction strategy uses attribute-value pairs called *threshold values* in order to make decisions about which algorithm (case-based reasoning, collaborative filtering, content-based etc.) to use and how to combine them. However in naive hybrid recommender systems, the supplied threshold values are static and do not change during the execution of recommender engine. For example, as shown in the Fig. 1, in the Duine Framework the knowledge that designs the selection strategy is provided manually by experts. The combination of techniques should not be fixed within a system and that the combination ought to be based on knowledge about strengths and weaknesses of each technique and that the choice of techniques should be made at the moment a prediction is required.

Our system, AdaRec, employs switching hybridization by deciding which algorithm is most suitable to provide a recommendation. The decision is based on the most up-to-date knowledge about the current user, other users, the information for which a prediction is requested, other information items and the system itself. In AdaRec we propose the use of ML techniques to analyze and learn which user and item features in a personalized recommender system are more adequate for correct recommendations.

The main purpose of a prediction strategy is to use the most appropriate prediction technique in a particular context. Domain experts usually design the static decision rules in a generic way. But each domain has unique characteristics including user behaviors, emerging trends etc. Adaptive prediction strategy frequently re-design the selection rules of prediction techniques according to the domain drifts. Fig. 1 shows a sample prediction strategy that decides when to use which predictor (gray nodes) by using the threshold values (arrows).

To make decisions about which algorithm is suitable for the current context, threshold values, algorithm's state and users' feedbacks are used by the adaptive prediction strategy. In the AdaRec, initial strategy is defined by using the expert knowledge like the naive hybrid recommender systems. System starts with the

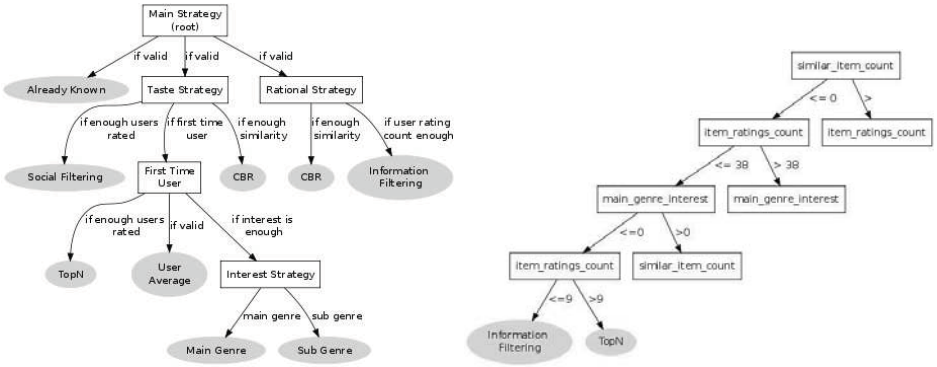

Fig. 1. The default Duine Framework's static prediction strategy is shown on the left. A subset of a sample adaptive prediction strategy that uses domain attributes is shown on the right.

defined initial prediction strategy. Later on the Learning Module adjusts the prediction strategy to the current systems domain.

Depending on the nature of the domains (movie, music, book etc.) different attribute-value combinations can be used for prediction strategy design. In our proposed system, since we tested on MovieLens dataset, we chose these specific attributes that have meaningful correlations between movie domain and prediction techniques. We believe that, by measuring the changes on these attributes, we can capture the domain drifts and trends.

1. *item ratings count*, the number of ratings that the current item has.
2. *item similar user count*, similar users count that have already rated the current item.
3. *similar item count*, the number of similar items count according to similarity measures.
4. *main genre interest*, main genre interest certainty[2] of the current item among the users items.
5. *sub genre interest*, sub genre interest certainty[2] of the current item among the users items.

Decision trees/decision rules are constructed using the combination of the above five attributes. As shown in the Fig. 1 at each node of the decision tree an attribute is compared with a value, which is called *threshold value*. These five attributes are used to classify the prediction techniques. In our system, we used and tested different prediction techniques. These are; *topNDeviation, userAverage, socialFiltering, CBR, mainGenreLMS, subGenreLMS, informationFiltering*. Because of the dynamic nature of the domain, these attributes create different forms of decision trees.

[2] Sub-genre and main-genre are domain-dependent attributes. Genre certainty is measured by using the movies metadata information which is interpreted according to MovieLens and IMDB (Internet Movie Database) datasets

4 Overview of the AdaRec System

Fig. 2 depicts the architectural overview of the proposed AdaRec system. Our experimental framework is an extension of the open-source Duine Framework. System consists of two core parts, *Recommender Engine* and *Learning Module*. Recommender Engine is responsible for generating the predictions of items based

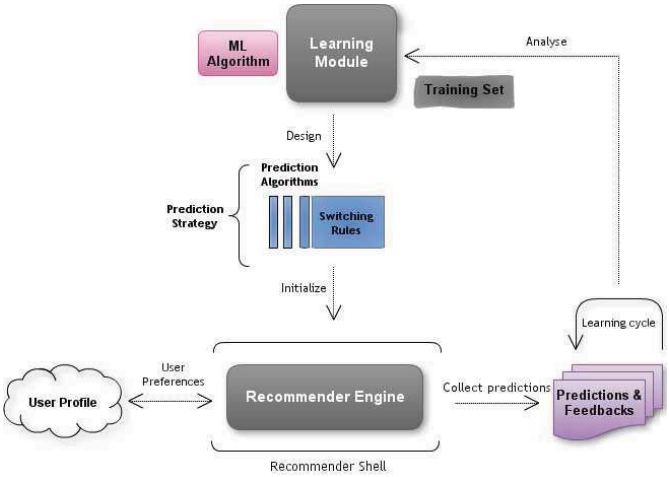

Fig. 2. Overall architecture of the AdaRec System.

on the previous user profiles and item contents. It attempts to recommend information items, such as movies, music, books, that are likely to be of interest to the user. Learning Module handles the new prediction strategy creation upon the previous instances and performance results of the prediction techniques on each learning cycle. It allows the building of new decision trees/decision rules based on the previous recorded instances. *Learning cycle* is a logical concept that represents the re-design frequency of the prediction strategy. A learning cycle indicates the instance count. Each instance, based on the indicated count, and prediction algorithms performance results are collected between two learning cycles.

The learning module first tests the accuracy of each algorithm in the system. Then the prediction strategy is re-designed by the learning module in order to improve proper use of algorithms. Adaptive prediction strategy improves its prediction accuracy by learning when to use which algorithm. The learning module adapts the hybrid recommender system to the current dynamics of the domain.

A learning algorithm is attached to the framework that analysis previously stored instances. The algorithm determines the attribute-value pairs and builds a model with respect to previously recorded instances called *training set*. Previous predictions and user feedbacks are fed to the training set of the next

learning cycle. User feedbacks and MAE (Mean Absolute Error) are the main concepts, which describe the trends in the domain. Adaptive prediction strategy learns its domain trends over time via unobtrusive monitoring and relevance feedback. The prediction strategy is adapted to the current context by analyzing the previous performance results of the techniques. Different ML algorithms that induce decision trees or decision rule sets could be attached to our experimental design.The architecture is open and flexible enough to attach different ML algorithms. Learning module adjusts the prediction strategy and re-initializes the recommender system with modified strategy.

5 Experiments

In our experiments MovieLens[3] one million ratings dataset is used, with 6040 users and 3900 movies pertaining to 19 genres [5]. MovieLens dataset contains explicit ratings about movies and has a very high density. In order to train the recommender system, the MovieLens dataset is divided in to different logical time periods (subsets) based on their distribution in time (*time-stamp*).

5.1 Experimental Setup

For our experiments, we use a widely popular statistical accuracy metric named MAE, which is a measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $< p_i, q_i >$, this metric treats the absolute error between them i.e., $|p_i, q_i|$ equally. The MAE is computed by first summing over these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. Formally;

$$MAE = \frac{\sum_{i=1}^{N} |p_i, q_i|}{N} \tag{1}$$

The Duine generates prediction ratings in unipolar range goes from 0 (not interesting) to +1 (interesting) with 0.5 representing a neutral interest. This original state of the framework is referred as *baseline*. We want to compare the prediction quality obtained from the framework's baseline (non adaptive) to the quality obtained by AdaRec (adaptive). The approach will be considered useful if the prediction accuracy is equal or better than the baseline.

The validation process is handled using the following procedure: The ratings provided by the users are fed to the system one by one, in the logical (time-stamp) order of the system; when a rating is provided during validation, prediction strategy is invoked to provide a prediction for the current user and the current item. The adaptive system collects the feedback and current attributes of the system as instances. After the MAE has been calculated, the collected instances with the best performed algorithms are fed as feedback to the learning module. Whenever the collected instances reached the learning cycle's instance count (1K

[3] `http://www.movielens.umn.edu`

for example), the prediction strategy of the system will be redesigned by the adaptive system according to the instances. This way, when the next learning cycle is processed, the adaptive system knows and has learned from all the previously processed ratings.

5.2 Results and Discussion

In the experiments, different number of instances, such as 1K, 2K and 3K, are used. The purpose of different number of instances was to compare the influence of the instance size on algorithms at the same domain. In the adaptive system, C4.5, BF-Tree and Conjunctive Rules classifier algorithms are attached to the learning module and its results are recorded. We tuned the algorithms to optimize and configured to deliver the highest quality prediction without concern for performance. We also plot the result of the best MAE (less is better) of the hybrid recommender in the current context at each run. The best MAE is referred to *best* in the charts. Therefore it is possible to compare the performance of the algorithms and the best possible result. The left part of the Fig. 3 shows

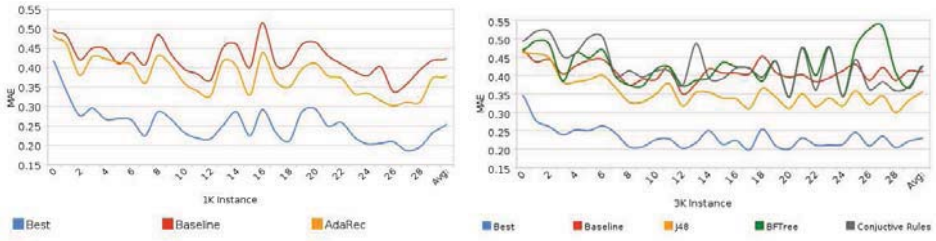

Fig. 3. Quality of prediction (MAE) using AdaRec with J48 (pruned C4.5, BF-Tree and Conjunctive Rules) vs Baseline & Best.

the prediction quality (average MAE) results of our experiments, which used 1K instances as learning cycle count, for the AdaRec (with attached J48 ML algorithm) as well as the original system referred as baseline. In this chart, average MAE is plotted for each of the runs. On each run adaptive system re-designs it's prediction strategy according to the previous runs instances. We make two important observations from this chart. First, the prediction quality of adaptive system performs better than the baseline. It can also be observed from the chart that the the adaptive systems MAE that changes overtime shows parallel variation. The adaptive system shows similar trends with the best results. It can be inferred that the adaptive system adapts itself to the current trends. Second, it can also be observed from the chart that as we increase the instance size of algorithms the quality tends to be superior (decreased MAE). The same trend is observed in the case of 2K and 3K instances. The right part of the Fig. 3 presents the performance results of our experiments for both of the learning algorithms

and baseline. From the plot we see that using the *J48* as the plugged ML, the MAE is substantially better than the baseline. This is due to the reason that with the adaptive approach the recommender system adapts itself to the current context.

6 Conclusion & Future Work

In this paper, we introduced an adaptive hybrid recommender system that re-designs its prediction (switching) strategy according to the performance of prediction algorithms. The learning module adjusts and re-designs the switching rules based on the feedbacks gathered from users. As a result, the system becomes adaptive to the application domain, and the accuracy of recommendation increases as more data are accumulated. In the MovieLens dataset, the adaptive system adapts an adequate to good prediction strategy as it is sometimes capable of providing more accurate predictions than the baseline (naive hybrid system).

Initial experimental results show its potential impacts. Therefore, for the next step this learning module will be deeply tested with various heterogeneous datasets. Also, our future work will explore the effectiveness of other ML techniques for use in the learning module. We believe that with this adaptive learning module, a naive hybrid recommender should have higher chance to allow its users to efficiently obtain an accurate and confident decision.

References

1. Bao, X., Bergman, L., Thompson, R.: Stacking recommendation engines with additional meta-features. In: Proceedings of the third ACM conference on Recommender systems. pp. 109–116. ACM (2009)
2. Bell, R., Koren, Y., Volinsky, C.: The bellkor solution to the netflix prize. KorBell Team's Report to Netflix (2007)
3. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)
4. Middleton, S.E.: Capturing knowledge of user preferences with recommender systems. Ph.D. thesis, University of Southampton (May 2003)
5. Movielens dataset. http://www.grouplens.org/node/73
6. Setten, M.V.: Supporting People in Finding Information- Hybrid Recommender Systems and Goal Based Structuring. Telematica instituut fundamental research series no:016, Telematica Instituut (November 2005)