Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends

Fatih Aksel
Department of Computer Engineering
METU
fatih.aksel@ceng.metu.edu.tr

Ayşenur Birtürk
Department of Computer Engineering
METU
birturk@ceng.metu.edu.tr

ABSTRACT

Hybrid recommender systems combine several algorithms based on their hybridization strategy. Prediction algorithm selection strategy directly influence the accuracy of the hybrid recommenders. Recent research has mostly focused on static hybridization schemes which are designed as fixed combinations of prediction algorithms and do not change at run-time. However, people's tastes and desires are temporary and gradually evolve. Moreover, each domain has unique characteristics, trends and unique user interests. In this paper, we propose an adaptive method for hybrid recommender systems, in which the combination of algorithms are learned and dynamically updated from the results of previous predictions. We describe our hybrid recommender system, called AdaRec, that uses domain attributes to understand the domain drifts and trends, and user feedback in order to change it's prediction strategy at run-time, and adapt the combination of content-based and collaborative algorithms to have better results. Experiment results with datasets show that our system outperforms naive hybrid recommender.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *Information filtering, Retrieval models, Selection process*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Design, Experimentation, Algorithms

Keywords

Hybrid Recommender Systems, Switching Hybridization, Decision Tree Induction, Hybrid Recommender Framework, Adaptive Recommender Systems

1. INTRODUCTION

Content-based and collaborative filtering are the two major recommendation techniques that have come to dominate the current recommender system area. 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. Other recommendation technologies include hybrid techniques, knowledge-based approaches etc [6]. The popular Amazon item-to-item system [7] is one of the well-known recommender system that uses collaborative filtering techniques. NewsWeeder [12] and InfoFinder [11] are the pure content-based recommender systems that analyze the content of items, in their recommendation process.

Previous research in this area, has shown that these techniques suffer from various potential problems-such as, sparsity, reduced coverage, scalability, and cold-start problems [1, 6, 23]. For example; collaborative filtering techniques depend on historical ratings of across users that have the drawback, called cold start problem - an item cannot be recommended until it has been rated by a number of existing users. The technique tends to offer poor results when there are not enough user ratings. Content-based techniques can overcome the cold start problem, because new items can be recommended based on item features about the content of the item with existing items. Unfortunately, content-based approaches require additional information about the content of item, which may be hard to extract (such as, movies, music, restaurants). Every recommendation approach has its own strengths and weaknesses. Hybrid recommender systems have been proposed to gain better results with fewer drawbacks.

Most of the recommender system implementations focuses on hybrid systems that use mixture of recommendation approaches [6]. 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 run-time. 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 Section 2. In Section 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 Section 6.

2. RELATED WORK

Personalization techniques have been investigated extensively in several areas of computer science. Especially in the domains of recommender systems, personalization algorithms have been developed and deployed on various types of systems [1].

There have been several research efforts to combine different recommendation technologies. The BellKor system [4], 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 [3] 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 run-time metrics to learn next level predictors by linear regression. However combining many engines level by level results performance problems at run-time. Our approach combines different algorithms on a single hybrid engine with an adaptive strategy.

Some hybrid recommenders choice the best suited recommender engine for a specific case (user, item, input etc.). For example, the Daily Learner system [5], which is a personal web-based agent, selects the best recommender engine according to the confidence levels. But in order to handle different engines in a common point, confidence scores should be comparable.

The use of machine learning algorithms for user modeling purposes has recently attracted much attention. In [13], the authors proposed a hybrid recommender framework to recommend movies to users. The system uses a content-based predictor to enhance existing user data, and then provides personalized suggestions through collaborative filtering. In the content-based filtering part of the system, they get extra information about movies from the IMDB[1] and handle each movie as a text document. User and item profiles are built by using a Naive Bayes classifier that can handle vectors of bags of words; where each 'bag-of-words' corresponds to a movie-feature (e.g. title, cast, etc.). The Naive Bayes classifier is used to approximate the missing entries in the user-item rating matrix, and a user-based collaborative filtering is applied over this dense matrix.

In our system, we choice the Duine Framework[2] for our recommendation engine component, which is an open-source hybrid recommendation system [20]. The Duine framework allows users to develop their own prediction engines for recommender systems. 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. The result of a Duine prediction engine is the retrieved set of information with added data about how interesting each piece of information is for the user [20, 17].

3. ADAREC: AN ADAPTIVE HYBRID RECOMMENDER SYSTEM

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. The construction of accurate strategy that suits in all circumstances is a difficult process. A well-designed adaptive prediction strategy offers advantages over the traditional static one.

In our approach we use the switching hybridization in order to decide which prediction technique is most suitable to provide a prediction. Prediction techniques, also called the predictors are combined into an upper prediction model that is called prediction strategy. The central concept in combining multiple predictors using the switching hybridization method is the prediction strategy. Prediction Strategy, which defines the algorithm selection strategy, changes the behavior of the recommender engines at run-time.

Most of the currently available personalized information systems focus on the use of a single selection technique or a fixed combination of techniques [20, 14]. However, application domains are dynamic environments. Users are continuously interacting with domain, new concepts and trends emerge each day. Therefore, user interests might change dynamically over time. It does not seem possible to adapt trends by using a static approach (static prediction strategy). Instead of static methods dynamic methods that can adapt to change on domains, could be more effective.

Different design approaches might be used for the prediction strategy adaptation. Rule based, case based, artificial neural networks or Bayesian are some of the learning techniques. Each technique has its own strengths and weaknesses. In this paper, we introduce self adaptive prediction strategy learning module which employs a strategy based on

[1]`http://www.imdb.com/`

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

its attached learning technique. Learning module initializes prediction engine according to the specified machine learning technique. This prediction strategy adapts itself to the current context by using the previous performance results of the techniques. Different machine learning algorithms that induce decision trees or decision rules could be attached to our experimental design.

4. PREDICTION STRATEGY LEARNING

Duine Recommender offers extensive options for configuring various recommender algorithms. It provides a sample of most common recommendation algorithms that can be combined in algorithm strategies. In the Duine Recommender the knowledge that designs the selection strategy is provided manually by experts [8, 21]. However, the combination of different techniques in a dynamic, intelligent and adaptive way can provide better prediction results. 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.

Hybridization of a recommender system employs using the best prediction technique from the available ones. The main purpose of a prediction strategy is to use the most appropriate prediction technique in a particular context. Adaptive prediction strategy depicts the selection rules of prediction techniques. Figure 1 shows a sample prediction strategy that decides when to use which predictors (gray nodes) by using the threshold values (arrows).

Prediction strategy employs the selection rules of the available prediction techniques. Our experimental hybrid recommender system has plenty of pre-defined prediction techniques. These prediction techniques are implemented by using different paradigms. Content-based and collaborative filtering are the two principal paradigms for computing recommendations [23]. In our system we have used content-based, collaborative filtering, knowledge based and case-based prediction techniques.

To make decisions about which predictor is suitable for the current context, threshold values, predictors' state and users feedback are used by the adaptive prediction strategy. The state of a predictor is described by the amount and at the quality of knowledge that is available to the predictor. In other words, the knowledge that is used by the prediction technique is the basis of its predictions. One of the objectives of the prediction strategy is to select the right prediction technique according to the current states of the predictors. The initial strategy is defined by using the expert knowledge. System starts with an initial prediction strategy. Later on the Learning Module adjusts the prediction strategy to the current systems' domain.

Decision trees and decision rules are model based strategies. Each node in a decision tree represents some attributes and each branch from a node corresponds to a possible value for that attribute. When trying to classify a certain instance, as seen in the Figure 1, one starts at the root of the decision tree and move down the branches of the tree according to the values of the attributes until a leaf node is reached. The leaf nodes represent the decisions of the tree [20].

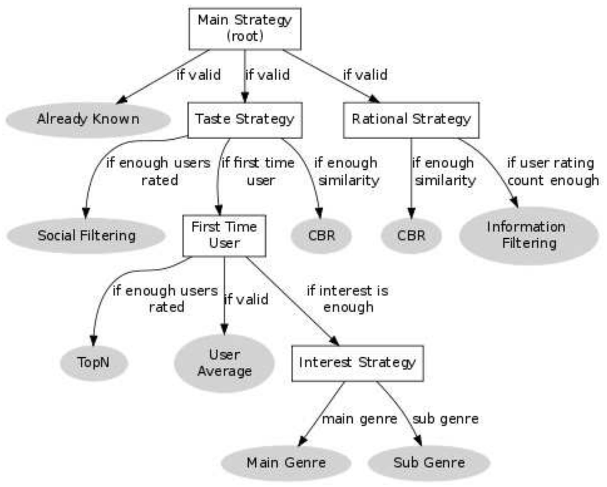

Figure 1: The Duine Recommender's prediction strategy depicted as decision tree. Grey nodes represent prediction algorithms and arrows represent the attributes&threshold values.

Adaptive strategy can be designed using by rule sets or trees that contain the knowledge on decisions. Decision rules can also be expressed as decision trees. Experts often describe their knowledge about a process using decision trees and decision rules as they are easy to interpret by other people. The decision trees are a good interface between experts and the recommender systems.

Adaptive prediction strategy is in the form of a decision tree. Another way to represent decision tree is using decision rules. Decision rules generally take the form of IF ... THEN ... rules, i.e. `IF` $\texttt{attribute}_1$ `=` $\texttt{value}_1$ `AND` $\texttt{attribute}_2$ `=` $\texttt{value}_2$ `THEN` $\texttt{result}_2$.

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 choice 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 of the current item among the users items.

5. *sub genre interest*, sub genre interest certainty of the current item among the users items.

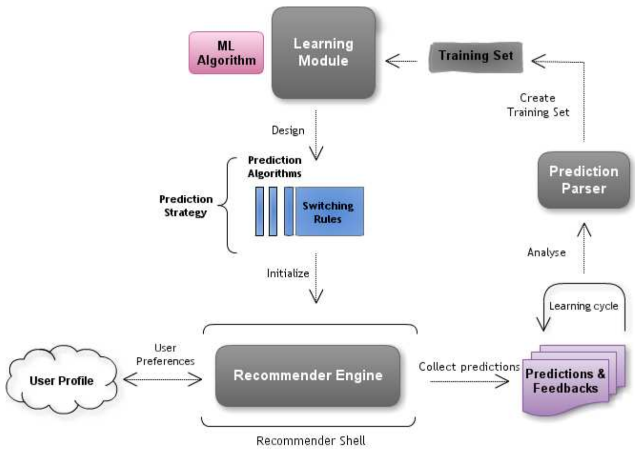

Figure 2: Overall architecture of the AdaRec System.

Decision trees/decision rules are constructed using the combination of the above five attributes. As shown in Figure 1 at each node of the decision tree a attribute is compared with a value, which is called *threshold value*. These five attributes are used to classify the prediction algorithms.

The Recommender System needs to work with the best suited prediction technique for its domain and users. In our system, we used and tested different prediction techniques. These are; *topNDeviation, userAverage, socialFiltering, CBR (Case Based Reasoning), mainGenreLMS, subGenreLMS, informationFiltering*. Because of the dynamic nature of the domain, these attributes create different forms of decision trees.

Each domain has unique characteristics including user behaviors, emerging trends etc. Recommender engine able to adapt itself to the changes in the domain by analyzing the changes. Also the system can capture the trends in the domain able to re-design its' attached prediction strategy. In our system, we used and tested different prediction techniques.

The quality of a decision tree depends on both the classification accuracy and the size of the tree[10]. After reviewing and testing many of the options, we decided to use two decision tree classifiers; *J48 (pruned C4.5 decision tree)*[18], which is the WEKA's[3] implementation of the decision tree learner based on the C4.5 decision tree algorithm, and *BF-Tree (best first-decision tree)*, which is a decision tree learner that uses a best first method of determining its branches. Also in order to compare the rule-based and tree induction methods we plugged and tested the *Conjunctive Rules* classifier, which is a simple rule learner that learns a set of simple conjunctive rules.

5. OVERVIEW OF THE ADAREC SYSTEM

Figure 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*.

[3]Waikato Environment for Knowledge Analysis: `http://www.cs.waikato.ac.nz/ml/weka/`

Recommender Engine is responsible for generating the predictions of items based 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. The recommender generate the predictions by using its attached prediction strategy. The implementation here uses the open source Duine Framework for the recommender engine.

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 redesign frequency of the prediction strategy. Each instance, based on the indicated count, and prediction algorithm's performance results are collected between two learning cycles. The learning module modifies the values of attributes in decision rules, which is also called threshold values according to the gathered results of the prediction techniques performance. The old prediction strategy is modified by using recommender engines' machine learning algorithm (rule tuning, rule adaptation, decision tree induction etc.). The modification of the threshold values allows recommender system to analyze&adapt the nature of the users and the domain.

The learning module first tests the accuracy of the each predictor in the system. Than the prediction strategy is redesigned by the learning module in order to improve proper use of predictors. Adaptive prediction strategy improves its' prediction accuracy by learning better when to use which predictors. The learning module adapts the hybrid recommender system to the current characteristics of domain.

Previous predictions and user feedbacks are fed to the training set of the next learning cycle. Inductive learning is used in learning from the training set. In our experiments we tested different (1K, 1.5K, 2K and 3K) instance sizes for training sets. The training set contains the instances from the previous learning cycle results. There are quite a few inductive learning techniques to choose from, including information theoretic ones (e.g. Rocchio classifier), neural networks (e.g. back-propagation), instance-based methods (e.g. nearest neighbour), rule learners (e.g. RIPPER), decision trees (e.g. C4.5) and probabilistic classifiers (e.g. naive Bayes) [14].

The User Profile, is the representation of the user in the system. For each active user a user model is stored in the user profile. User profile holds the knowledge (such as preferences, feedbacks, feature weights etc.) about users in a structured way. The Recommender Shell, encapsulates the Recommender Engine's interaction with other modules. The shell serves the created prediction strategies to the engine. The Prediction Parser, produces the performance results of the prediction algorithms based on the analyzing of the collected predictions & feedbacks. This module handles the decomposition of the prediction results and generates the training set of sample instances with current attributes.

User feedbacks and MAE (Mean Absolute Error) are the main criteria, which describe the trends in the domain. Adaptive prediction strategy learns its domain trends over time via unobtrusive monitoring and relevance feedback. In our proposed system, we focused self adaptive prediction strategy that classifies according to its' attached machine learning technique. This prediction strategy adapts itself to the current context by using the previous performance results of the techniques. Different machine learning 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 machine learning algorithms.

6. EXPERIMENTS

In this section we present a brief discussion of our experimental dataset, evaluation metric followed by the experimental results and discussion.

In order to assess the impact of our proposed adaptive recommender and different machine learning algorithms, we calculated prediction accuracy (MAE) of the system using different configurations of the machine learning schemes. Different MovieLens datasets are examined during the experiments.

The datasets are divided into temporal subsets according to their *time-stamp* values. Natural domain trends and changes in user interests are handled by using the subsets of the dataset.

6.1 Datasets

We used data for our recommender system from MovieLens[4], which is a web-based research recommender system that debuted in Fall 1997 [15].

In our experiments MovieLens one million ratings dataset is used, with 6040 users and 3900 movies pertaining to 19 genres. 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 temporal sets based on their distribution in time. When testing the ratings of first sets are used for recommender engine training [15, 9, 16].

6.2 Experimental Setup

Recommender systems researchers use a number of different measures for evaluating the success of the recommendation or prediction algorithms [19, 22]. For our experiments, we use a widely popular statistical accuracy metric named Global Mean Absolute Error (MAE), which is a measure of the deviation of recommendations from their true user-specified values. The MAE is defined as the average difference between the predicted ratings and the real user ratings, as defined within the test sets. Formally, MAE can be defined as:

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

where p_i is the predicted value for item i and r_i is the user's rating.

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

The aim of the experiments is to examine how the recommendation quality is affected by our proposed learning module. The present model of the Duine Framework is non adaptive but it supports predictor level learning. This original state of the framework is referred to *baseline*. As shown in the Figure 1, Duine recommender uses a static prediction strategy as its' hybridization scheme, which does not change at run-time. We want to compare the prediction quality obtained from the framework's baseline (non adaptive) to the quality obtained by our proposed experimental framework (adaptive). The approach will be considered useful if the prediction accuracy is better than the baseline. At the first iterations both systems are initialized with the same strategy, which is the default strategy of the Duine Framework.

The validation process is handled using the following procedure:

1. The ratings provided by the dataset are fed to the system one by one, in the logical order of the system (ordered by timestamps).

2. When a rating is provided during validation, prediction strategy is invoked to provide a prediction for the current user and the current item. The average prediction error can be used a performance indicator of the attached prediction strategy.

3. After the error has been calculated, the given rating is provided as feedback to the recommender system. The adaptive system collects the feedback as well as the current attributes of the system as instances.

4. Whenever the collected instances reached the learning cycle's instance count (1000 instances 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 has learned from all the previously processed ratings. This process is repeated for all ratings at both adaptive and non-adaptive (baseline) systems in the test set. At the end MAE is calculated by averaging absolute errors within the baseline and the adaptive system, as described above.

6.3 Results & Discussion

Each prediction provided by the two different systems are examined. The prediction accuracy and the prediction error (MAE) are recorded. In experimenting with the MovieLens dataset, we considered both the proposed method, called *adaptive system*, and the existing method, called *baseline*.

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.

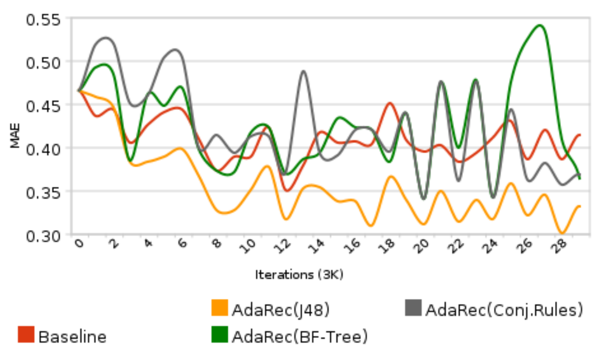

Figure 3: Quality of prediction (MAE) using AdaRec (attached J48, BF-Tree and Conjunctive Rules with 1000 instances) vs Baseline & Best.

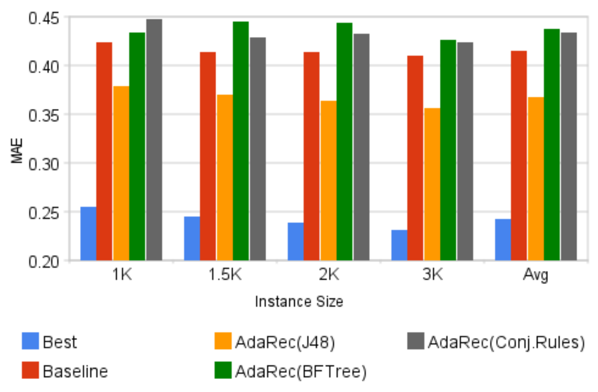

Figure 4: Prediction accuracy comparison of the AdaRec (attached J48, BF-Tree and Conjunctive Rules) vs Baseline & Best for different instance sizes.

We also plot the result of the best MAE (less is better) of the hybrid recommender in the current context at each iteration. The best MAE is referred as *best* at the charts. Therefore it is possible to compare the performance of the algorithms and the best possible result.

Figure 6 presents the prediction quality (average MAE) results of our experiments for the adaptive system as well as the original system referred to baseline. J48 algorithm is used in these experiments. In this chart, prediction quality is plotted for each of the iterations. On each iteration adaptive system re-designs its prediction strategy according to the previous iteration's performance result (feedbacks and results). It can be inferred from chart that, the prediction quality of the adaptive system performs better than the baseline. It can also be observed from the chart that the adaptive system adapts itself to the changes in the domain and users.

Figure 4 presents the prediction accuracy results of the different instance sizes. In the figure, we also plot the overall performance of ML algorithms as average. It can also be observed from the charts that as we increase the instance size of algorithms the quality tends to be superior (decreased MAE). In case of other algorithms it is expected that increasing the number of instances would mean small MAE values. The same trend is observed in the case of 2K and 3K instances.

Figure 5 presents the average MAE of hundred runs for 1K instance size. In this experiment we evaluate the impact of more runs for 1K instance size. It can be observed from the chart that changes in the MAE show the similar trends for both the baseline, adaptive and best systems. A harmony is achieved through time. The curves are similar in such a way that if one of them has a good prediction accuracy in one run, the others also have the good accuracy for that run.

In order to determine the impact of the instance size, we carried out an experiment where we varied the value of instance size (1K, 1.5K, 2K and 3K). For each of these training set/instance size values we run our experiments. Figure 6 presents the whole picture of the adaptive system's performance results. From the plot we observe that the quality of MAE increases as we increase the instance size.

Figure 3 presents the accuracy results of all used ML techniques for 1K instance size. It can be observed from the figure that the J48 attached AdaRec system performs better than the other systems. Also BF-Tree seems good enough to compete against the naive hybrid system. But BF-Tree algorithm needs some domain depended configuration adjustments. From the figure we also observe that the Conjunctive Rules algorithm underperforms among other ML algorithms. The rule learner algorithm seems not stable as the decision tree learners.

The results also show that, when using well tuned algorithms, the adaptive system is stable (better than the baseline) in obtaining the average prediction accuracy. This durability, which can be called the impact of learning, is established by the learning module. In order to adapt recommender engine to the current trends, the learning module re-designs the prediction strategy. The learning ability supports the recommender system adaptation to the changes.

7. CONCLUSION & FUTURE WORK

In this paper, we introduced an adaptive hybrid recommender system, called AdaRec, that combines several recommender algorithms in which the combination parameters are learned and dynamically updated from the results of previous predictions. Research study shows that traditional static hybrid recommender systems suffer from changing user preferences. In order to improve the recommendation performance, we handle domain drifts in our approach. The Learning Module re-designs its prediction (switching) strategy according to the performance of prediction techniques based on user feedbacks. As a result, the system adapts to the application domain, and the performance of recommendation increases as more data are accumulated. In the MovieLens dataset, the proposed adaptive system outperforms the baseline (naive hybrid system).

Initial experimental results show its potential impacts. Therefore, for the next step, we plan to further testing the learning module with various heterogeneous datasets. It would be interesting to examine the different domains other than movie

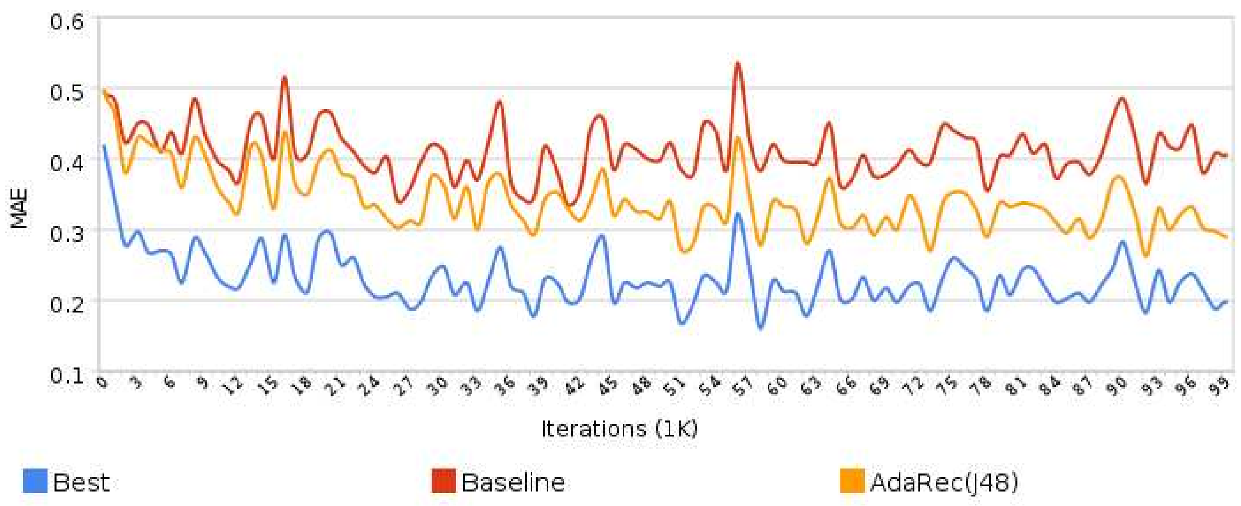

Figure 5: Quality of prediction using J48 (pruned C4.5), Baseline and Best according to 100 iterations. Previous 1000 instances are used for learning on each iteration.

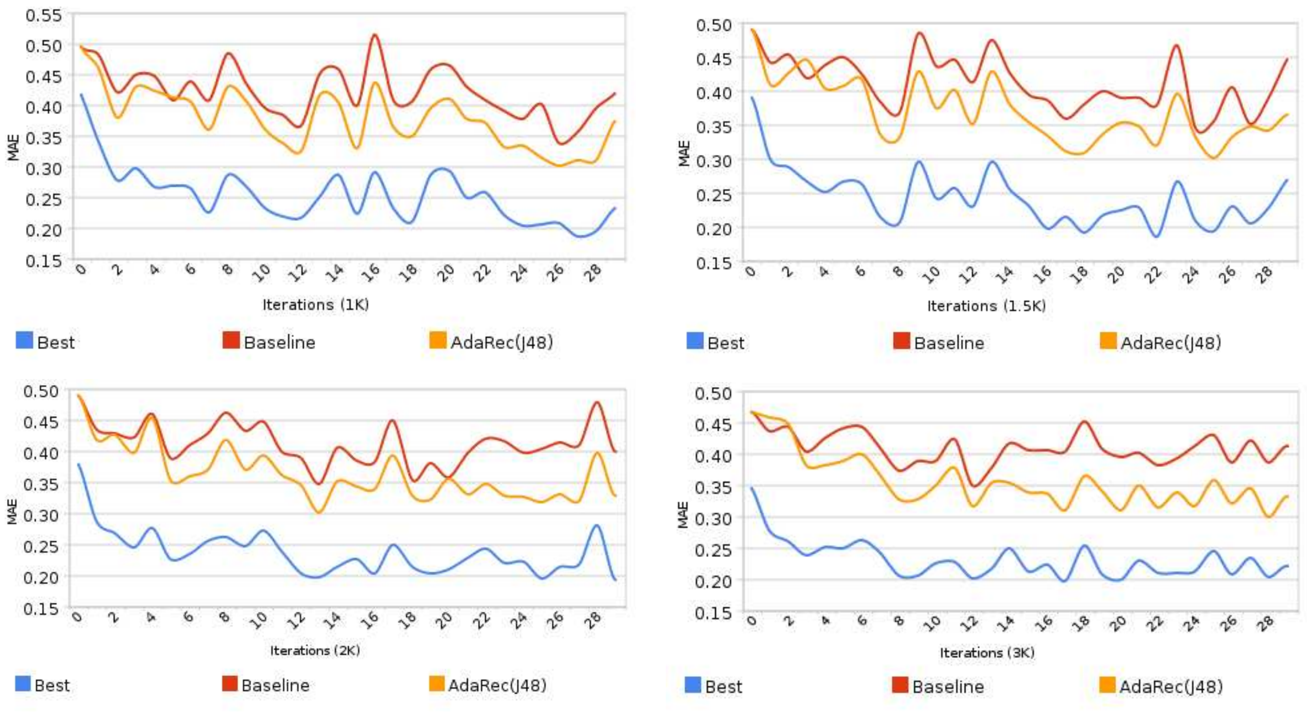

Figure 6: Comparison of the different instance sizes. 1K, 1.5K, 2K and 3K instances are used for learning cycle. 3K learning cycles' prediction quality yields better results than the others

(such as music, book, news etc.). Also, our future work will explore the effectiveness of other machine learning techniques for use in learning module. In our experiments we fixed the used attributes for domain monitoring. It would be also interesting to use dynamic attributes, which means to use different attributes on different iterations. We believe that with this adaptive learning module, a traditional hybrid recommender should have higher chance to allow its users to efficiently obtain an accurate and confident decision.

8. ACKNOWLEDGMENTS

We would like to acknowledge the anonymous reviewers of HaCDAIS'2010 and PRSAT'2010 workshops for their useful comments. An earlier version of this study has appeared as a short paper in the proceedings of HaCDAIS 2010 workshop [2].

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] F. Aksel and A. Birturk. An Adaptive Hybrid Recommender System that Learns Domain Dynamics. *International Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions (HaCDAIS-2010) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2010 (ECML PKDD 2010)*, page 49, 2010.

[3] X. Bao, L. Bergman, and R. Thompson. Stacking recommendation engines with additional meta-features. In *Proceedings of the third ACM conference on Recommender systems*, pages 109–116. ACM, 2009.

[4] R. Bell, Y. Koren, and C. Volinsky. The bellkor solution to the netflix prize. *KorBell Team's Report to Netflix*, 2007.

[5] D. Billsus and M. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, 2000.

[6] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[7] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 117–124, New York, NY, USA, 2009. ACM.

[8] J. Herlocker and J. Konstan. Content-independent task-focused recommendation. *IEEE Internet Computing*, pages 40–47, 2001.

[9] The internet movie database. http://www.imdb.com/.

[10] D. Jensen and P. R. Cohen. Multiple comparisons in induction algorithms. In *Machine Learning*, pages 309–338, 1998.

[11] B. Krulwich and C. Burkey. Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI spring symposium on machine learning in information access*, pages 100–112, 1996.

[12] K. Lang. Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[13] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 187–192. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[14] S. E. Middleton. *Capturing knowledge of user preferences with recommender systems.* PhD thesis, University of Southampton, May 2003.

[15] Movielens dataset. http://www.grouplens.org/node/73.

[16] Netflix dataset. http://www.netflixprize.com.

[17] Novay. Duine recommender — telematica instituut-novay, 2010. [Online; accessed 1-July-2010].

[18] S. Salzberg and A. Segre. Book review: "c4.5: Programs for machine learning" by j. ross quinlan. In *Machine Learning.* morgan kaufmann publishers, inc, 1994.

[19] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. pages 345–354. ACM Press, 1998.

[20] M. V. Setten. *Supporting People in Finding Information- Hybrid Recommender Systems and Goal Based Structuring.* Telematica instituut fundamental research series no:016, Telematica Instituut, November 2005.

[21] M. V. Setten, M. Veenstra, and A. Nijholt. Prediction strategies: Combining prediction techniques to optimize personalization. In *Proceedings of the workshop Personalization in Future TV*, volume 2. Citeseer, 2002.

[22] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[23] C. Ziegler. *Towards Decentralized Recommender Systems.* Freiburg i. br, Albert-Ludwigs-Universität Freiburg,Germany, June 2005.