SCALABLE RECOMMENDER SYSTEM THAT IMPROVES GENERALIZATION

Gökhan ÇAPAN
Master of Science Thesis

Graduate Program of Computer Engineering
July, 2013

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Gökhan Çapan'ın "Scalable Recommender System that Improves Generalization" başlıklı Bilgisayar Mühendisliği Anabilim Dalındaki, Yüksek Lisans Tezi 10.07.2013 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

		Adı-Soyadı	İmza
Üye (Tez Danışmanı)	**:**	**Yard. Doç. Dr. ÖZGÜR YILMAZEL**	**.....**
Üye	**:**	**Doç. Dr. HÜSEYİN POLAT**	**.....**
Üye	**:**	**Yard. Doç. Dr. KAMİL ÇEKEROL**	**.....**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ABSTRACT

Master of Science Thesis

SCALABLE RECOMMENDER SYSTEM THAT IMPROVES GENERALIZATION

Gökhan ÇAPAN

Anadolu University
Graduate School of Sciences
Computer Engineering Program

Supervisor: Assist. Prof. Dr. Özgür YILMAZEL
2013, 36 pages

A major challenge for recommender systems is to generalize to cold-start prediction tasks, where no behavior data is available for the active user or the item. Content-based filtering is able to attack to this problem, while collaborative filtering ends up with accurate recommendations where high quality feedback is available.

Considering the domain of a prediction task can vary, an ensemble learning-based hybrid recommender model is described. The combined model learns separate linear combinations from validation data sets representing each domain: high quality feedback available, user or item is unseen. The problem is illustrated by creating groups of validation and test data sets accordingly, and referring to three kinds of complementary recommenders: matrix factorization based, user demographics, and item content-based. Experiments demonstrate that using those separate validation data sets; the hybrid recommender model adjusts weights such that it converges to the individual recommender that performs the best on a domain.

Keywords: Recommender System, Generalization, Hybrid Recommendation

ÖZET

Yüksek Lisans Tezi

GENELLEMEYİ İYİLEŞTİREN ÖLÇEKLENEBİLİR TAVSİYE SİSTEMİ

Gökhan ÇAPAN

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Özgür YILMAZEL
2013, 36 sayfa

Tavsiye sistemlerinin temel zorluklarından bir tanesi, kullanıcı ya da ürün hakkında yeterli davranış bilgisi bulunmayan soğuk-başlangıç verisine genellemedir. İçerik tabanlı süzgeçleme bu problemi çözebilse de, işbirlikçi süzgeçleme yöntemleri, yeterli davranış verisi olduğunda, daha hatasız tavsiyelerde bulunur.

Bir tavsiye sorgusunun farklı etki alanlarından birine ait olabileceği göz önünde bulundurularak, küme öğrenme tabanlı bir melez tavsiye sistemi anlatılmıştır. Bu birleşik yöntem, her biri farklı etki alanlarını (yüksek kalite geribildirim verisi mevcut, kullanıcı soğuk-başlangıç durumunda, ürün soğuk-başlangıç durumunda) temsil eden doğrulama veri kümeleri üzerinde doğrusal kombinasyonlar öğrenir. Bu problem, uygun şekilde doğrulama ve test verisi oluşturularak; daha sonra üç çeşit tümleyici tavsiye sistemi (matris çarpanlarına ayırma tabanlı işbirlikçi süzgeçleme, ürün içeriği tabanlı süzgeçleme, kullanıcı demografisi tabanlı süzgeçleme) kullanılarak gösterilmiştir. Farklı doğrulama veri kümeleri kullanılarak yapılan deneyler; melez yöntemin, o etki alanında en iyi doğruluğa ulaşan tek tavsiye sistemine yakınsayacak şekilde ağırlıklandırma öğrendiğini göstermektedir.

Anahtar Kelimeler: Tavsiye Sistemi, Genelleme, Melez Tavsiye

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Assist. Prof. Dr. Özgür YILMAZEL, who has guided me with his knowledge throughout this research. Without his patience and support, this thesis could not be completed.

Secondly, I would like to thank to the members of the thesis committee, Assoc. Prof. Dr. Hüseyin POLAT and Assist. Prof. Dr. Kamil ÇEKEROL, for accepting to be in the committee, and for their invaluable reviews, corrections, and advices to improve this thesis.

Last but definitely not least, I would like to thank to my parents and siblings for trusting me throughout my life, supporting me patiently, and encouraging me in my most difficult times. I am one of the luckiest people in the world for having them.

Gökhan Çapan

July, 2013

TABLE OF CONTENTS

ABSTRACT ..... **i**

ÖZET ..... **ii**

ACKNOWLEDGEMENTS ..... **iii**

TABLE OF CONTENTS ..... **iv**

LIST OF FIGURES ..... **vi**

LIST OF TABLES ..... **vii**

1. INTRODUCTION **1**

2. RECOMMENDATION TECHNIQUES AND CHALLENGES **4**

2.1. Recommender System Definition 4

2.2. Collaborative Filtering 4

2.2.1. Nearest Neighbor-based Collaborative Filtering 5

2.2.2. Matrix Factorization Based Collaborative Filtering 8

2.3. Cold-start Problem and Content-based Filtering 10

2.4. Hybrid Recommendation 11

3. LEARNING ENSEMBLES OF RECOMMENDERS **13**

3.1. Recommender Systems Blending 14

4. ELEMENTS OF THE HYBRID MODEL **16**

4.1. Item Content-based Filtering 16

4.2. User Demographics-based Filtering 18

5. DOMAIN-ADAPTIVE HYBRID MODEL 19

5.1. A Linear Stacking Based Model 19

5.2. Hybrid Recommendation of Varying Weights 19

5.3. Relation to Other Ensemble Techniques 22

6. EXPERIMENTAL STUDY 23

6.1. Preparing Experimentation Data 23

6.2. Learning Strategy 25

6.3. Experimental Results 26

6.4. Scalability and Performance 30

7. CONCLUSIONS AND FUTURE DIRECTIONS 31

REFERENCES 32

LIST OF FIGURES

5.1 Hybrid recommendation of varying weights 21
6.1 RMSE per numbers of iterations on different domains 28

LIST OF TABLES

6.1 Domain-specific data splits 23
6.2 Content features 24
6.3 Parameters and number of iterations for base recommenders 25
6.4 Hybridization weights for different domains 26
6.5 RMSE of base recommenders and the hybrid model for each domain 27
6.6 RMSEs of different recommenders in half cold-start domain 29
6.7 Training running times of different recommenders per iteration 30

1. INTRODUCTION

Recommender systems automate personalized discovery for individuals. Users are recommended a small, ranked subset of a very large set of previously unseen items, where ranking is based on the predicted interest of theirs in those items. To predict that interest, a typical recommender infers a function that scores a $user, item$ pair.

One way to infer the user-item scoring function is collaborative filtering, which leverages community's past behavior. Sources other than collaborative activity to score a $user, item$ pair include content of users' favorite items, user demographics, and contextual information. Item content-based recommender predicts the score based on content characteristics of the items that the user positively rated before. User demographics-based estimates how likely the active item may fit the user characteristics. Another technique is contextual recommendation, where the idea is scoring a $user, item$ pair in a particular context, which may depend on location, time, certain task, and so on.

Contextual recommendation can be added on an existing recommender, but content-based recommendation alternates collaborative filtering in some ways. Collaborative filtering is famous for its superior prediction accuracy [1], and ability to discover serendipitous recommendations. However, problems occur when behavior data lack high-quality feedback, that is to say, collaborative filtering cannot generalize to new users or items –known as the cold-start problem. Content-based filtering techniques can generalize to new data –as long as the new entity can be represented by its content or demographic characteristics– but they are limited where collaborative filtering is strong: they cannot predict as accurate as collaborative filtering under high-quality feedback domain, and they tend to over-specialize to the items very similar to a user history in content–which obviously do not help a user much, except certain use cases. In fact, over-specialization of content-based filtering decreases the system quality so much that recommending from a small category all the time dissatisfies, even frustrates the users [2].

Still, when she wants to deploy a recommender in production, a practitioner would desire the system to generalize to new users and items. Consider a successful Web 2.0 application. The application will hopefully attract new users, and new items will be added. The ideal recommender system in this case, should be able to score a $user, item$ pair even when no behavioral data is available for one of the entities, at the same time leverage superior accuracy of collaborative filtering when high-quality feedback is available. As content-based and collaborative filtering complement each other, combining those techniques – by merging ranked recommendation lists, averaging prediction scores of different recommenders as $user, item$ scoring function, sequentially running different recommenders etc.– results in a hybrid scheme that might produce accurate and serendipitous recommendations; and preserve the generalization capability at the same time.

Consider a weighting-based hybrid recommender that combines the scores of collaborative, item content-based, and user demographics-based filtering techniques. Let this system learn an optimal weighting scheme on a validation data set. That recommender, when it is deployed in production, is going to encounter with various query domains: high-quality feedback is available for both user and item, user is cold-start, and item is cold-start. The problem with the hypothetical hybrid recommender is that the weighting scheme is identical for all these domains; even though collaborative filtering should outweigh others in the first domain, user demographics-based in the second, and item content-based in the third.

The hypothetical weighting-based recommender was an example of ensemble learning, particularly linear stacking [3]: learning a linear combination of base learners (meta-learning) on a validation data set. We may further enhance the ensemble, for the combination to adapt to different domains. In stacking, we can incorporate some meta-features to the weighted combination, such as number of ratings. Mixture of experts model [4] on the other hand, defines a weighting scheme –of complementary base learners– that depends on input query. So if the domain of the $user, item$ pair is observed and used for weighting scheme

selection, we can apply a mixture of experts model to achieve domain-adaptive recommendation.

In this thesis, we survey the recommendation techniques those attempt to generalize to new users and items while preserving high accuracy of collaborative filtering-based recommender systems. In addition, we describe an ensemble recommendation approach (a method that lies between mixture of experts and stacking), hybrid recommendation of varying weights, and hypothesize that it can adapt to the input domain. The proposed technique requires that:

- Domains, say set D, are identified upfront
- Validation data sets, each of which reflects a particular domain, are created
- $|D|$ number of recommender techniques, each of which is expected to perform best on one identified domain, are defined
- Domain of a $user, item$ query can be observed at recommendation time

Different recommenders are learned on the same training data set, and varying weights are inferred on mentioned validation data sets. Appropriate weighting scheme is selected at recommendation time.

We test the hypothesis on a Java library that can perform matrix factorization-based collaborative filtering, user demographics-based filtering, and item content-based filtering models. We examine if the hybrid scheme converges to the best-performing base recommender technique in each domain, and outperforms all in the production case, where the test data set contains $user, item$ queries from different domains.

This thesis is organized as follows: Section 2 surveys related recommendation techniques and approaches to improve recommender system generalization. In Section 3, we describe the recommendation algorithms we use, and the hybrid model is introduced in Section 4. In Section 5, we describe the experimentation methodology and present the results. Finally, Section 6 provides a discussion for the study, and provides future directions.

2. RECOMMENDATION TECHNIQUES AND CHALLENGES

2.1. Recommender System Definition

For a system with m users and n items, suppose $Y_{m\times n}$ is a sparse matrix where y_{ij} is the interest indicator of user i on item j. A recommender system aims to predict the missing values of the Y matrix, which may be numerical, categorical (binary or multi-class) or ordinal (For notational convenience, we are going to denote prediction for a missing interest indicator for $\langle i,j\rangle$ as $\tilde{y}_{ij}$). In other words, a recommender is a function $f : U\times I \rightarrow T$, where U is the set of users, I is the set of items, and T is the set of target values; for example, an interval [0,5], a set of categories $\{c_0, c_1\}$, or a set of ordinal values $\{poor, fair, good, great\}$. This function is an approximation for actual interest indicators. Besides the core-recommending job (predicting rating for a $\langle user, item\rangle$ pair or producing top k recommendations for a user u), a recommender system has some other use cases, including finding most similar items to an item and finding most similar users to a user.

The success of a recommender is measured by various methods [5]. Perhaps the most important is measuring accuracy, that is, how accurate the recommender estimates the unknown ratings, compared to the original rating matrix.

2.2. Collaborative Filtering

Collaborative filtering predicts the unknown degree of interest of user u in item i based on other users' declared interest in i. While recommending to user u, other people in the system affect the prediction for u, proportional to how much their tastes are similar to u's.

Early examples of recommender system research, particularly collaborative filtering as an information filtering technique, include the proposed solution of Goldberg et al. [6] to reduce the email overload, Tapestry; idea of producing recommendations based on similarities of the target user's and other users'

interest profiles, and a music album/artist recommender system implementation, Ringo [7]; GroupLens, an architecture of collaborative filtering on Usenet news [8]. Recommender systems have had a wide area of application in industry, too. Many successful businesses such as Amazon[1] (the popular e-commerce site) and Netflix[2] (the DVD rental service) trust recommender systems as a major part of their user interaction process. Netflix even organized a recommender system challenge to improve the accuracy of their proprietary recommender [9]. Front runners of the challenge state that some combination of multiple prediction methods, including neighborhood based and matrix factorization based, give the best results; since it is the ensemble of different approaches learning different aspects of data [10-13].

2.2.1. Nearest Neighbor-based Collaborative Filtering

Neighborhood-based method is the intuitive way of producing collaborative filtering based recommendations. Concretely, a user neighborhood-based recommender computes the neighborhood of the active user u, and then to predict her rating for item i, it outputs a weighted average of the ratings of the users in her neighborhood on that specific item [8, 14, 15]. Similarly, an item neighborhood based recommender computes the degree of similarity between the target item i and the set of items in active user u's history, and then outputs a weighted average of the ratings of u on those items [16].

To describe neighborhood-based recommenders formally, let us define some notation, additional to the aforementioned set. The similarity between users u and v, and items i and j will be denoted by $sim(u, v)$ and $sim(i, j)$ respectively. Note that the actual rating of user u to item j is y_{uj}, while the predicted rating of u to item i is $\tilde{y}_{ui}$. An example similarity measure for computing pairwise user similarity is the Pearson correlation coefficient [8, 14, 15], which is:

[1] http://www.amazon.com

[2] http://www.netflix.com

$$sim(u,v) = \frac{\sum_{i \in I_{u,v}} (y_{ui} - \bar{Y}_u)(y_{vi} - \bar{Y}_v)}{\sqrt{\sum_{i \in I_{u,v}} (y_{ui} - \bar{Y}_u)^2 \sum_{i \in I_{u,v}} (y_{vi} - \bar{Y}_v)^2}} \quad (2.1)$$

Here $I_{u,v}$ denotes the items those u and v both rated, and $\bar{Y}_u$ and $\bar{Y}_v$ denote the mean ratings for users u and v, respectively. To compute the similarity between two items i and j, one can use adjusted cosine similarity[16], which is:

$$sim(i,j) = \frac{\sum_{u \in U_{i,j}} (y_{ui} - \bar{Y}_u)(y_{uj} - \bar{Y}_u)}{\sqrt{\sum_{u \in U_{i,j}} (y_{ui} - \bar{Y}_u)^2 \sum_{u \in U} (y_{ui} - \bar{Y}_u)^2}} \quad (2.2)$$

where $U_{i,j}$ is the set of users who rated both i and j.

With user similarities, one can compute the predicted rating of user u to item i with the following equations. Following equations compute the predicted rating with user neighborhood and item neighborhood based techniques, respectively [15, 16].

$$\tilde{y}_{ui} = \bar{Y}_u + \frac{\sum_{v \in U_i} sim(u,v)(y_{vi} - \bar{Y}_v)}{\sum_{v \in U_i} sim(u,v)} \quad (2.3)$$

$$\tilde{y}_{ui} = \bar{Y}_i + \frac{\sum_{j \in I_u} sim(i,j)(y_{uj} - \bar{Y}_j)}{\sum_{j \in I_u} sim(i,j)} \quad (2.4)$$

Here U_i represents the set of users who rated item i, where I_u means those items rated by user u. Following is the list of alternative strategies of populating U_is and I_us:

- One may allow U_i include all users who rated i, excluding the users having a zero (or undefined) similarity with the active user u. Similarly, a I_u may be a subset of all items rated by u, each of which has a non-zero similarity with the target item i.
- The first option may be constrained by users/items whose similarities with the active user/target item are above a threshold value.

- Other alternative is to keep only the top k users/items in terms of their similarities with the active user/target item in the sets mentioned in the first option.

Neighborhood based collaborative filtering uncovers relationship those other models sometimes cannot. In addition, recommendations from a neighborhood model are explainable, that is, the system can tell its users why they saw the recommended items [17].

However, there are some liabilities. Computing the neighborhood and predicting the score upon a recommendation request is expensive; and pre-computing similarities may be an option. But they cannot provide up-to-date recommendations if this pre-computation phase is infrequent. Item-based neighborhood models are more advantageous since item sets are relatively static, and one can at least use the updated user history on online rating prediction phase. Even if the pre-computation method works, neighborhood models are still costly with frequent updates on the model.

The pre-computation of similarities step will, for each user, loop through the candidate neighbors of that user, and compute the similarity. Since the similarity of two users u and v is non-zero only if there is at least one item that they both have rated, one can only consider the candidates, rather than all other users. For users who rated a little number of items, this approach can result in a huge performance gain.

In summary, neighborhood-based collaborative filtering has advantages, yet it is limited:

- *Generalization:*

  A neighborhood based collaborative filtering system cannot compute the user neighborhood if the user has not rated an item before. Besides, it cannot recommend an item if the item has not been rated by any user yet.

- *Performance:*

  Neighborhood computation is expensive, and there is a trade-off between keeping the recommender up to date and avoiding neighborhood computation at recommendation time. Item neighborhood-based

recommenders are more advantageous than the user neighborhood based ones in that case.

2.2.2. Matrix Factorization Based Collaborative Filtering

Given a matrix Y, matrix factorization is the process where Y is factorized into two matrices, W and H, such that $Y \approx WH$. Given Y is an $m \times n$ matrix, W and H are $m \times r$ and $r \times n$ matrices, respectively. Generally $r << min(m, n)$, implying that the two factor matrices together construct a compressed version of Y [18]. To compute Y_{ij} from W and H, the following dot product is used:

$$Y_{ij} \approx W_i {H^T}_j = \sum_{k=1}^{r} W_{ik} \times {H^T}_{jk} \tag{2.5}$$

For performing collaborative filtering, the sparse interest matrix Y is decomposed into two factor matrices, A_{mxr} and B_{rxn}. The A matrix is the factor matrix for users, and the B matrix is the factor matrix for items. Suppose that the estimated interest matrix is $\tilde{Y} = AB$. Then to find a good estimate for $\tilde{Y}$, one needs to minimize a cost function with respect to A and B. For predicting numerical ratings, a proper optimization objective is minimizing the squared distance, which is:

$$\min_{\alpha,\beta} \frac{1}{2} \sum_{i,j} (y_{ij} - \tilde{y}_{ij})^2 \tag{2.6}$$

For that sparse interest matrix Y whose non-empty elements are denoted by $y_{i,j}$, α_i (row i of the A matrix) and β_j^T (column j of the B matrix) vectors are need to be learned. Since the estimated rating $\tilde{y}_{ij} = \alpha_i \beta_j^T$, the optimization objective for factor matrices becomes the following:

$$\min_{\alpha,\beta} \frac{1}{2} \sum_{i,j} (y_{ij} - \alpha_i \beta_j^T)^2 \tag{2.7}$$

Several minimization approaches may be used to find the optimal α_is and β_j^Ts, including alternating least squares [19] and stochastic gradient descent [20]. We apply stochastic gradient descent algorithm here to incrementally train the model. For each rating y_{ij}, the factor vectors α_i and β_js are updated. The update rules for

k^{th} indices of α_i and β_j^T are shown in (2.8) and (2.9). Note that the updates should be performed for each k from 1 to r, and done simultaneously.

$$\begin{aligned} \alpha_{ik} &\leftarrow \alpha_{ik} - \frac{1}{2} lr \frac{\partial}{\partial \alpha_{ik}} [(y_{ij} - \alpha_i {\beta^T}_j)^2] \\ &= \alpha_{ik} + lr(y_{ij} - \alpha_i {\beta^T}_j) {\beta^T}_{jk} \end{aligned} \tag{2.8}$$

$$\begin{aligned} \beta_{jk} &\leftarrow \beta_{jk} - \frac{1}{2} lr \frac{\partial}{\partial \beta_{jk}} [(y_{ij} - \alpha_i {\beta^T}_j)^2] \\ &= \beta_{jk} + lr(y_{ij} - \alpha_i {\beta^T}_j) \alpha_{ik} \end{aligned} \tag{2.9}$$

Here lr is the learning rate, and should be determined on validation data.

Matrix factorization for collaborative filtering introduces $r(n+m)$ number of parameters, making the learning process prone to overfitting. The learning process may yield a model that fits to training data too well, and causes the loss of generalization. One way to avoid overfitting is applying regularization, that is, introducing additional parameters to penalize the updates. The objective defined in (2.7) becomes the following when $L2$ norm regularization is applied:

$$\min_{\alpha,\beta} \frac{1}{2} \sum_{i,j} \left(y_{ij} - \alpha_i {\beta^T}_j\right)^2 + \frac{\lambda}{2} \left(\sum_{i,k} (\alpha_{ik})^2 + \sum_{j,k} ({\beta^T}_{jk})^2 \right) \tag{2.10}$$

where λ is the regularization rate. The regularized versions of the update rules for α_{ik} and β_{jk} are as follows:

$$\begin{aligned} \alpha_{ik} &\leftarrow \alpha_{ik} + lr\left((y_{ij} - \tilde{y}_{ij}) {\beta^T}_{jk} - \lambda \alpha_{ik}\right) \\ {\beta^T}_{jk} &\leftarrow {\beta^T}_{jk} + lr\left((y_{ij} - \tilde{y}_{ij}) \alpha_{ik} - \lambda {\beta^T}_{jk}\right) \end{aligned} \tag{2.11}$$

Adding bias terms for each user and item is proven to improve the accuracy of matrix factorization based recommenders. The biased model is as follows [21]:

$$\tilde{y}_{ij} = {b^{(u)}}_i + {b^{(i)}}_j + \alpha_i {\beta^T}_j \tag{2.12}$$

In [21], the bias terms are trained by incorporating a parameter called $global\_mean$ originally, but we utilize them as simple intercept terms here:

$$b^{(u)}{}_{jk} \leftarrow b^{(u)}{}_{jk} + lr\left(y_{ij} - \tilde{y}_{ij} - \lambda_2 b^{(u)}{}_{jk}\right)$$
$$b^{(i)}{}_{jk} \leftarrow b^{(i)}{}_{jk} + lr\left(y_{ij} - \tilde{y}_{ij} - \lambda_2 b^{(i)}{}_{jk}\right) \quad (2.13)$$

where λ_2 is the regularization parameter for bias terms.

Finally, we list some techniques to improve matrix factorization model, and refer to the original work for details:

- Incorporating user and item side information [19]
- Incorporating context information [22]
- Incorporating temporal dynamics [23]
- Modeling through binary implicit feedback [24]

2.3. Cold-start Problem and Content-based Filtering

One challenge a collaborative filtering based recommender system faces with is the cold-start problem: the generalization inability to new data (inputs including a new user or item). Consider a successful recommender system that hopefully welcomes new users regularly. The system cannot satisfy those new users unless it takes care of the new-user case of the cold-start problem. The new item problem might be negligible for systems having static item set, but, systems consisting of user-generated content, items with short lifetimes or systems constantly adding new content suffer from the new item case of the cold-start problem.

A recommender system would be able to recommend to new users by utilizing user profiles. Demographic information of users may be included to those user profiles, which allows us to find a match between types of users with items [25-27]. Also, social connections of users may include valuable signals while providing recommendations for users [28].

Item profiles, constructing the basis of content-based recommender systems may provide information while deciding whether an item is worth recommending to a user [29]. An item profile includes category, popularity, title, age, price, genre, and so on. Pazzani and Billsus [29] mention that user profiles

might include content properties of positively interacted items. For instance, in a movie-recommendation domain, knowing that a user likes to watch horror movies, we may recommend him a recent horror movie that she has not seen yet, and has not received adequate feedback to be recommended using collaborative filtering.

Content-based filtering is the recommender system technique where the prediction of the unknown degree of interest of user u in item i is calculated based on the similarity between profile of u and content of i. This approach may be performed by applying relevance feedback techniques, where a user profile is updated with each $user, item$ interaction [2, 29-31]. Probabilistic learning methods, such as naïve Bayes algorithm, can also be used to construct the user model [32, 33].

Content-based filtering can generalize to new items, however, they introduce some problems, for instance, over-specialization [25]. Consider a video recommendation site that trusts content-based filtering for recommendation. It would not surprise its user, since it would consistently recommend videos in the same small category (or another content dimension) that is the user's favorite.

2.4. Hybrid Recommendation

Hybrid (content-based and collaborative) recommender systems aim to overcome the cold-start problem while preserving high-quality recommendations of collaborative filtering.

One way to implement hybrid recommender systems is combining results of collaborative and content-based recommenders. Claypool et al. [1] proposed an approach for predicting user interests in online newspaper articles based on a weighted average of content-based and collaborative filtering based recommenders. Pazzani's method is aggregating ranked results lists of different recommenders [27].

In addition, performing content-based discovery techniques prior to collaborative filtering based recommenders provides a solution to overcome the cold-start problem, too. Li and Kim [34] used a clustering approach discovering

similar existing items to a new item, to use the past behavior data related to those existing items as if they are related to the new item. In their *Content Boosted Collaborative Filtering* approach, Melville et al. [35] provided a way to compute neighborhood of users more intelligently. They extended the user-item interest matrix, whose missing elements are filled with content-based predictions. They then performed collaborative filtering using the extended matrix.

Unified hybrid approaches may combine collaborative and content-based features in the same recommendation model. Basu et al. [36] represented a $user, item$ pair as a combination of collaborative, content-based, and hybrid features; and applied binary classification ($\{liked, disliked\}$) to make predictions. Park and Chu [37] proposed a regression approach that benefits from all possible side information of users and items, together with collaborative features, providing a solution to both new user and new item situations of cold start problem.

Matrix factorization based models naturally allow integrating content features of users or items into the factor model, and learning the parameters on those content features [19].

Some other noteworthy approaches are, definition of new similarity measures that work better than traditional ones for users who rated only a few items [38], and filling the user-item interest matrix with automatic bots [39].

3. LEARNING ENSEMBLES OF RECOMMENDERS

Consider a machine learning problem for which we have a number of alternative algorithms to model. An ensemble learner refers to a combination of those multiple learners. We expect from an ensemble model, if it is composed of complementary base learners, to be more accurate than individual learners [40].

Techniques of combining multiple learners vary, and we list here the related ensemble techniques:

- *Voting* [40]*:*

  In voting, where the prediction domain is numerical, a weighted average of results from individual learners is used to compute the final prediction. Formally, a voting based numerical prediction model is as follows:

$$y(x) = \sum_{i=1}^{L} w_i y_i(x) \tag{3.1}$$

  where $\forall w_i \geq 0$ and $\sum_{i=1}^{L} w_i = 1$. Here, $y(x)$ is the prediction model, L is the number of individual learners, and w_i is the weight assigned to the base learner i.

- *Mixture of Experts* [4, 40]:

  Mixture of Experts is a variant of voting, where the base learners are expected to be complementary, and the weighting scheme depends on the input. For a partition of the input space, one base learner performs the best. So if an input resembles partition i, we expect that learner i outweighs other learners in the weighting scheme, for that particular input. Formally, a mixture of experts ensemble model is as the following:

$$y(x) = \sum_{i=1}^{L} w_i(x) y_i(x) \tag{3.2}$$

- *Stacking [3]:*

  In stacking, we combine the base learners' outputs with a separate learning process, level-1 learning, which may incorporate other features and need not to be linear. Level-1 learning is performed on a separate validation data set.

3.1. Recommender Systems Blending

Ensemble learning for recommendation, or blending, has been shown to improve recommender system accuracy and generalization.

Yu et al. [41] show that a hierarchical Bayesian approach for blending collaborative and content based filtering techniques improves recommender system accuracy.

[42] hybridizes item neighborhood-based collaborative filtering and content-based filtering techniques using a decision template based combination, which has been shown to perform well for classification[43].

[11] describes the details of the winning solution for the Netflix Prize from the *The BellKor Pragmatic Chaos* team. The solution performs a complex blending method on over 100 recommendation techniques, Gradient Boosted Decision Tree, and achieves a higher accuracy than any base recommender [44, 45].

In [46], the authors mention that a hybrid system should adjust the combination of different recommenders based on the properties of the input (the $user, item$ pairs), and they applied stacking on user-based collaborative, item-based collaborative, and content-based filtering techniques. The level-1 learner incorporates runtime meta-features, and they used three different techniques (linear regression, model tree, and bagged model trees) as a level-1 learner. Reported results show that stacked generalization improves recommender system performance. From different methods of level-1 learning process, linear regression performs worst.

Sill et al. [47] presented a linear regression approach for performing stacking based ensemble as a recommender blend. The level-1 learner they describe also incorporates meta-features, features based on individual $user, item$

pairs. It models the weights of level-0 learner outcomes as linear functions of meta-features, and the approach yields accurate results as non-linear blending techniques, yet benefits from the advantages of linear regression. Writing the weight of a learner i on input x as a function of meta-features is formalized as:

$$w_i(x) = \sum_j v_{ij} f_j(x) \tag{3.3}$$

The blending model then becomes:

$$y(x) = \sum_{i,j} w_i(x) g_i(x) \tag{3.4}$$

where $y(x)$ is the prediction model for an input x, and $g_i(x)$ refers to learner i's output for input x.

In [48], the authors propose the *Social Trust Ensemble*, where blending is done on results of trust-based recommendation and matrix factorization based collaborative filtering.

Blending different techniques improve accuracy of a single recommendation technique, and when the blending incorporates input-based features (meta-features), the accuracy of a linear blend is further improved. We refer to [49] for an empirical analysis on blending recommenders.

In most of the reviewed blending techniques those incorporate the meta-features, number of ratings (by the user or for the item) affect accuracy improvement more than other meta-features. This gave us the idea to implement the domain-adaptive hybrid recommender, where a domain is identified in terms of number of ratings.

4. ELEMENTS OF THE HYBRID MODEL

In this section, we describe the individual recommender techniques those are selected to perform the hybrid recommender of varying weights. We start with defining three different domains to adapt, which are:

1. High-quality feedback is available for both user and item
2. Item cold-start
3. User cold-start

The domain-adaptive hybrid model learns a weighted combination of the individual recommenders for each domain, which hopefully results in a model where the best-performing individual recommender outweighs others. To this end, we refer to three different recommender techniques, each of which is expected to result in most accurate recommendations for one domain:

- Matrix factorization based collaborative filtering (Domain 1)
- Item content-based filtering (Domain 2)
- User content-based filtering (Domain 3)

The models we use share some common properties:

- They work on numerical feedback data
- They minimize the squared error function
- They can be trained using online learning techniques

We use the same matrix factorization based collaborative filtering model defined in (2.12), where training is performed as in (2.11) and (2.13).

4.1. Item Content-based Filtering

In item content-based filtering, we represent users and items with profiles. Suppose that an item profile, t_j is the content characteristics vector of the item, such as description, category, price, and so on. We have multiple alternatives to compute recommendations, some of which are:

- Representing a user by its rating history, and replacing the similarity computation in the item neighborhood-based collaborative filtering

approach with content similarity. This way, we can calculate content-based user-item scores using the (2.4).

- Representing a user with a vector of the same dimensionality with the item characteristics vector, whose values are determined by the content characteristics of the items those the user positively rated before. This way, we can combine the user and item vectors to score a $user, item$ pair.
- Representing a user by a parameter vector, α_i on the item content vector, t_j and learning the parameters on user history with an appropriate supervised learning algorithm. This way, we can combine the user parameter vector and the item content vector to score a $user, item$ pair.

We use the third approach to learn user profiles. Note that this approach introduces $m \times n$ number of parameters to be learned, where n is the dimensionality of an item content vector. The model that scores an i, j pair is as the following:

$$\tilde{y}_{ij} = \alpha_{i0} + \sum_{k=1}^{n} \alpha_{ik} t_{jk} \tag{4.1}$$

Using regularized ordinary least squares (linear regression), the objective function is as follows:

$$\min_{\alpha_{ik}} \frac{1}{2} \sum_{i,j} \left(y_{ij} - \tilde{y}_{ij}\right)^2 + \frac{\lambda}{2} \left(\sum_{i} \sum_{k=1}^{n} {\alpha^2}_{ik}\right) \tag{4.2}$$

Here, a y_{ui} is only available if the user u actually provided feedback for item i. One can train the model with each incoming feedback using an online and scalable approach, stochastic gradient descent method such that:

$$\alpha_{ik} = \alpha_{ik} + lr\left(\left(y_{ij} - \tilde{y}_{ij}\right) t_{jk} - \lambda \alpha_{ik}\right) \tag{4.3}$$

Training is done simultaneously for all k.

4.2. User Demographics-based Filtering

To estimate how an item j fits user i characteristics vector, we use the following model:

$$\tilde{y}_{ij} = \pi_{j0} + \sum_{k=1}^{m} \pi_{jk} x_{ik} \tag{4.4}$$

where a π_j denotes the parameter vector representing the item j's weights on content features, x_i is the user i's characteristics features, and m is the dimensionality of a user content features vector. The optimization objective and stochastic gradient descent update rules are straightforward, and shown in (4.5) and (4.6), respectively:

$$\min_{\pi_{jk}} \frac{1}{2} \sum_{i,j} (y_{ij} - \tilde{y}_{ij})^2 + \frac{\lambda}{2} (\sum_{j} \sum_{k=1}^{m} {\pi^2}_{ik}) \tag{4.5}$$

$$\pi_{jk} \leftarrow \pi_{jk} + lr((y_{ij} - \tilde{y}_{ij}) x_{ik} - \lambda \pi_{jk}) \tag{4.6}$$

5. DOMAIN-ADAPTIVE HYBRID MODEL

5.1. A Linear Stacking Based Model

We can combine selected three recommenders in the following way, and estimate the weighting scheme from the validation data as the level-1 learner:

$$\tilde{y}_{ij} = w_1 r_1(i,j) + w_2 r_2(i,j) + w_3 r_3(i,j) \quad (5.1)$$

where w_is represent the weights, and r_is the numerical scoring models of base recommenders. Note that the model combines results from different recommenders, which means prior to learning hybridization weights, the base recommenders need to be learned already.

The weights are learned using a separate validation data set. The optimal model can be found by minimizing a squared error function on validation data:

$$\min_{w_i} \frac{1}{2} \sum_{i,j \in V} (y_{ij} - \tilde{y}_{ij})^2 \quad (5.2)$$

here V denotes the validation data set (not included in training data) consisting of ratings.

If the base recommender set includes user demographics based and item content-based recommenders, this hybridization approach can generalize to new users and items, but for example, is not as accurate as collaborative filtering if high-quality feedback is available.

5.2. Hybrid Recommendation of Varying Weights

Content-based and collaborative filtering techniques have their own strengths and weaknesses. Collaborative filtering algorithms are known for predicting accurate, serendipitous recommendations [50], but they suffer from cold-start problem: they cannot generalize to new users and items. Content-based filtering algorithms are capable to generalize to new data, but the accuracy is lower and the problem of over-specialization occurs [25].

A recommender system in production encounters with various kinds of domains that the $user, item$ pairs are belong to:

- High-quality feedback is available for both the user and the item
- No or little feedback is available for the user
- No or little feedback is available for the item

No single recommendation technique fits all listed domains, which limits a recommender in production. Item content-based filtering approach is expected to perform best for the third case, but it cannot predict $user, item$ scores accurately for the second case, and cannot perform as good as collaborative filtering for the first case, for example.

Hybrid recommendation of varying weights is intended to improve the quality of a recommender, by improving accuracy and increasing generalization performance with a model that is robust to the domain of a $user, item$ input. To this end, a practitioner should first identify different domains for the $user, item$ queries that the recommender deployed in production would encounter with. The second step is to identify alternative recommendation techniques, each of which is expected to perform best for one particular domain. Finally, after carefully designing validation and test data sets representing identified domains, the hybridization weights per domain are learned separately. See Figure 5.1 for a graphical representation of the model.

Say we identified $|D|$ domains, and $r_d(i,j)$s, where a r_d is the function we expect to perform most accurate for inputs belonging to domain d. For instance, for domains we identified here, $r_1(i,j)$ refers to matrix factorization based collaborative filtering algorithm. Then for a prediction task $i, j, d,$ recommendation computation is done by:

$$\tilde{y}_{ijd} = \sum_{k=1}^{|D|} w_k^d r_k(i,j) \tag{5.3}$$

here w^ds are domain specific hybridization weights. Those domain-specific weights are learned using a validation data set (V^d), created for this particular domain d. Since we choose r_ds carefully so that an r_d performs most accurate in domain d, we expect w^ds are adjusted in such a way that w_d^d would be the highest weight among all w_k^ds.

Learning w^ds are performed separately for each d, using the same training data set and custom validation and test data sets. After learning individual recommender functions, we perform weight optimization for each domain d on validation data set V^dsuch that:

$$\min_{w_i^d} \frac{1}{2} \sum_{(i,j)\in V^d} (y_{ijd} - \tilde{y}_{ijd})^2 \tag{5.4}$$

Here, y_{ijd}s are actual rating values for i,js in V^d, and $\tilde{y}_{ijd}$ is the hybrid prediction score as defined in (5.3). Weight optimization per domain is normally performed once –or periodically– so any batch optimization method, such as batch gradient descent can be used to optimize those weights.

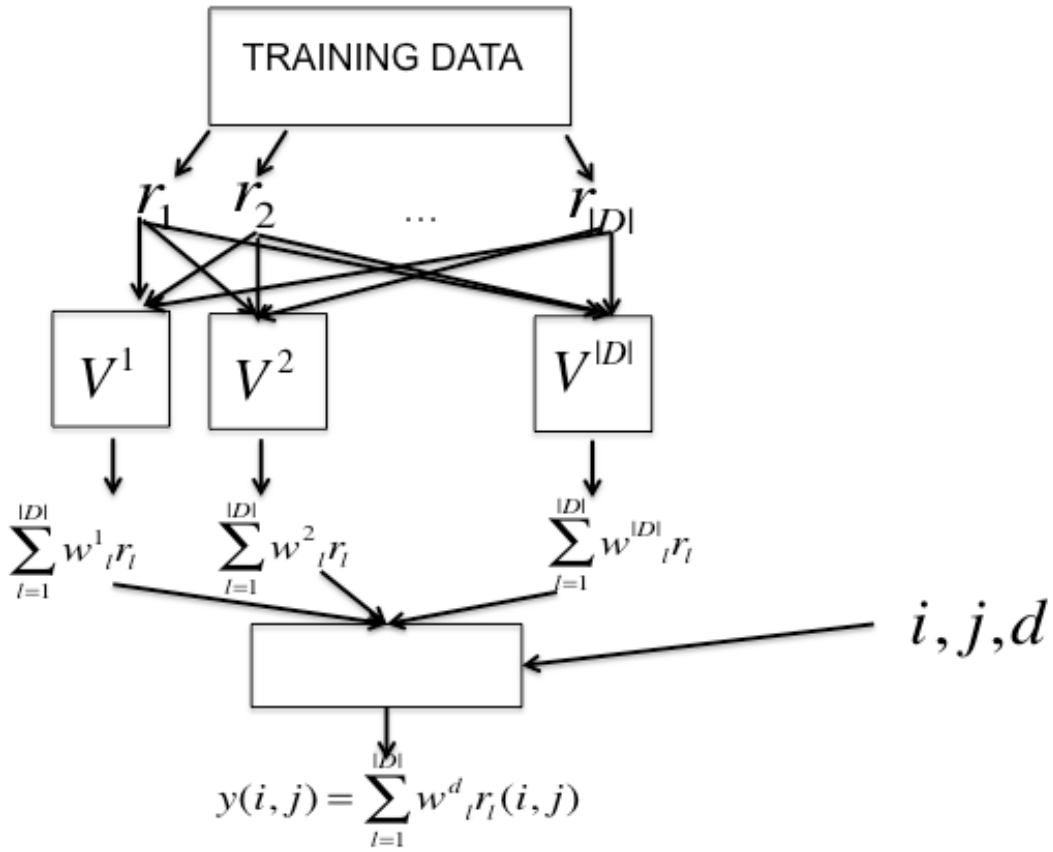

Figure 5-1 Hybrid recommendation of varying weights

5.3. Relation to Other Ensemble Techniques

Hybrid recommendation of varying weights has ties with stacking and mixture of experts ensemble models in some ways.

In stacking, a higher-level model is learned using the outputs of base learners, possibly incorporated with other features, on validation data. Our approach also utilizes outputs of base recommenders, and learns a higher-level weighting scheme on validation data. Hybrid recommender of varying weights differ from stacking such that, one need to carefully design validation data sets those represent identified domains.

Mixture of experts model defines a weighting scheme based on a particular input, which is the exact case in the hybrid recommender of varying weights, since we interpret the domain as a part of the input ($user, item, domain$). As in mixture of experts, our method expects complementary base recommenders, each of which performs the best on a domain.

6. EXPERIMENTAL STUDY

We run our experiments on MovieLens 1M data set, which contains one million numerical ratings of users on movies, and side information for both users and movies. In this section, we describe the steps we perform as experiments[1].

6.1. Preparing Experimentation Data

For learning domain-specific hybridization weights and testing if hybrid recommendation model works well, first, the entire ratings data set is split into three for each domain, as shown in Table 6.1.

Table 6.1 Domain-specific data splits

	Size (%)	**Description**
Train^1	70	Train data for high-quality feedback domain
Validation^1	20	Feedback is available for all users and movies
Test^1	10	Feedback is available for all users and movies
Train^2	70	Train data for item cold-start domain
Validation^2	20	Items are unseen in *Train*^{2}
Test^2	10	Items are unseen in *Train*^{2} and *Validation*^{2}
Train^3	70	Train data for user cold-start domain
Validation^3	20	Users are unseen in *Train*^{3}
Test^3	10	Users are unseen in *Train*^{3} and *Validation*^{3}

[1] All experiments provided here can be reproduced using the commandline tools after downloading and building the code in https://github.com/gcapan/mahout/tree/mahout-matrices.

Side information of users and items are vectorized to be used by content-based recommendation algorithms. Item content features and user characteristics we include to the model are listed in Table 6.2. Categorical features of more than two unique values, say c, were converted to c binary features.

Table 6.2 Content features

	Domain	**Description**
Genre	Categorical	Genre of a movie
YearsPassed	Numerical	Years passed since the movie was released
Gender	Categorical	Gender of the user
Occupation	Categorical	Occupation of the user
Age	Categorical	Age of the user (Discretized)

6.2. Learning Strategy

We first learn r_ds (base recommenders) on training data sets, as it is described in Section 4. The lr and λ parameters we used, and iteration numbers until the algorithms converge are shown in Table 6.3.

Table 6.3 Parameters and number of iterations for base recommenders

	Parameters	**Converges at**
MF-based collaborative	$\lambda = 0.0007, \lambda_2 = 0.0005,$ $r = 100, lr = 0.002$	75
Item content-based	$\lambda = 0.0005, lr = 0.004$	96
User demographics-based	$\lambda = 0.0005, lr = 0.007$	108

We use a simple, yet scalable approach for learning hybridization weights: stochastic gradient descent. For each rating in validation data set, w_k^ds are updated simultaneously, using the following formula:

$$w_k^d := w_k^d + lr\left(y_{ij} - \tilde{y}_{ijd}\right)r_k(i,j) \quad (6.1)$$

Since the hybridization is actually a weighted average of different base recommenders, when an iteration is completed, we simply transformed the learned weights to meet non-negative constraints, and ensured that their sum is equal to 1. Formally, we adjusted weights such that: $\sum_{k=1}^{L} w_k^d = 1, \forall d$ and $w_k^d \geq 0, \forall d \; and \; \forall k$. We observed that the process of learning hybridization weights converges really fast, at typically a couple iterations. Learning rate we used to train the hybridization scheme is 0.002.

6.3. Experimental Results

Root mean squared error (RMSE) is a widely used technique for testing the accuracy of a numerical prediction algorithm. RMSE of the hybrid recommender on the test data set of domain d ($Test^d$) with size $|Test^d|$ is calculated by:

$$RMSE = \frac{1}{|Test^d|} \sum_{i,j,y_{ij} \in Test^d} (\tilde{y}_{ijd} - y_{ij})^2 \quad (6.2)$$

We evaluated $\tilde{y}_{ijd}$ predictions on $Test^d$s, and compare the accuracy with base recommenders, to test our hypothesis that hybrid recommender of varying weights scheme is as accurate as the best-performing base recommender for domain d. In the hybrid scheme, best-performing recommender outweighs the others, as shown in Table 6.4.

Table 6.4 Hybridization weights for different domains

	w_1	w_2	w_3
Domain 1	0.8890	0.1045	0.0065
Domain 2	0	1	0
Domain 3	0.12	0	0.88

Figure 6.1 shows that the hybrid recommender performs as accurate as collaborative filtering when high-quality feedback is available, and preserves the generalization capability of content-based and demographics based recommenders in the cold-start domains. In Table 6.5, final RMSE scores where the algorithms converged are listed

Table 6.5 RMSE of base recommenders and the hybrid model for each domain

	Algorithm	**RMSE**
Domain 1	MF based	**0.857**
	User demographics-based	0.981
	Item content-based	1.018
	Hybrid model	**0.855**
Domain 2	MF based	1.586
	User demographics-based	**0.968**
	Item content-based	2.800
	Hybrid model	**0.968**
Domain 3	MF based	2.221
	User demographics-based	2.834
	Item content-based	**0.996**
	Hybrid model	**1.001**

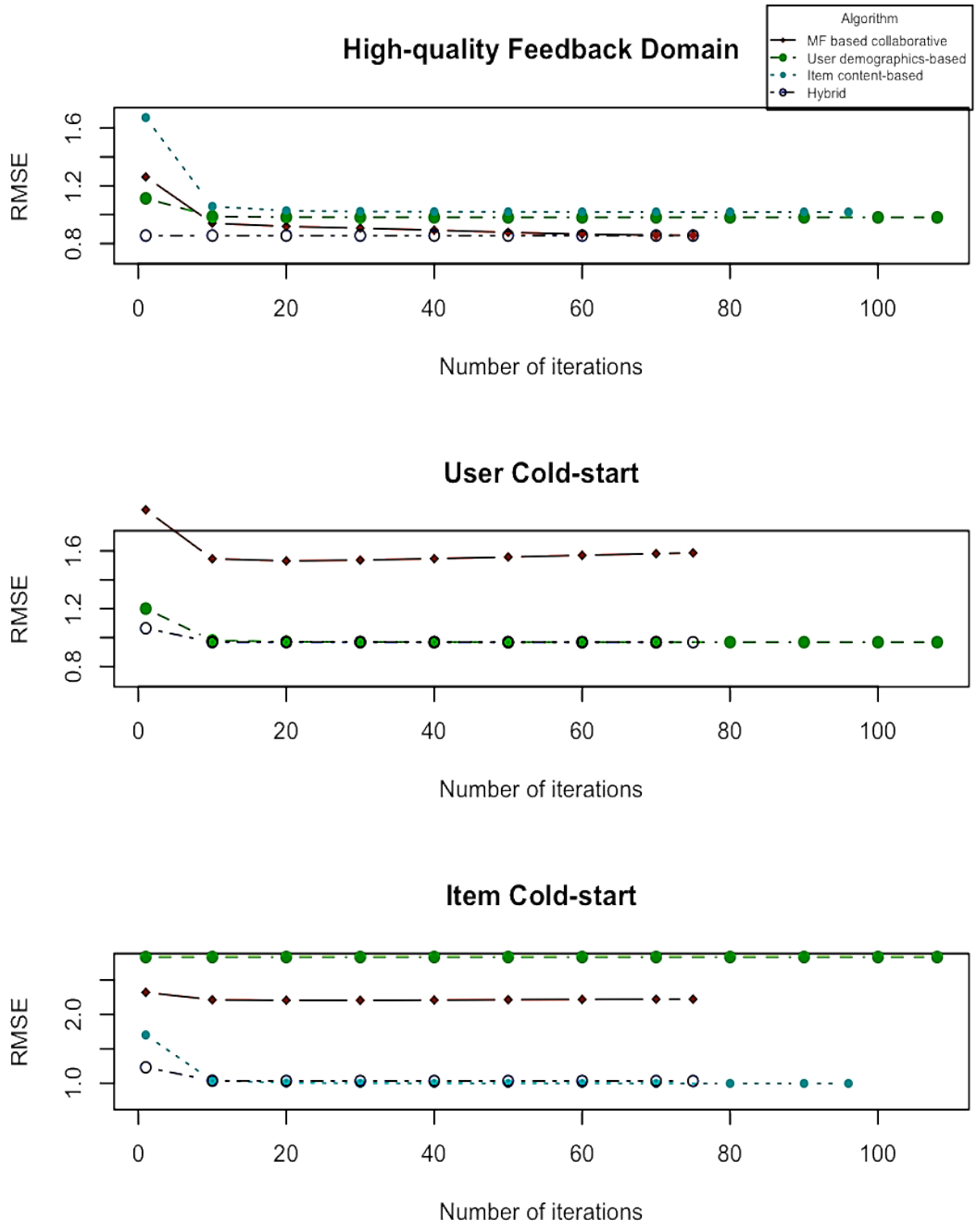

Figure 6-1 RMSE per numbers of iterations on different domains

Finally, to illustrate that the domain-adaptive approach outperforms the base recommenders in the production case, we split the MovieLens 1M data such that:

- We created a training data set from the 70% of entire data.
- We created two validation data sets, each of which constitutes 10% of entire data, where one validation set represents high-quality feedback domain, and the other represents item cold-start domain.
- We created one test data set (10% of entire data), which include $user, item$ queries from both domains, that is to say, half of the queries include cold-start items, rest of them received high-quality feedback that is available in training data.

After learning domain-adaptive hybrid recommender, we compared its accuracy with individual base recommenders, collaborative filtering and item content-based filtering. Results are presented in Table 6.6, showing that the hybrid approach outperforms base recommenders in the production case.

Table 6.6 RMSEs of different recommenders in half cold-start domain

	RMSE
MF-based collaborative	High-quality feedback domain: 0.856
	Real-world scenario: $\mathbf{1.67}$
Item content-based	Item cold-start domain: 1.02
	Real-world scenario: $\mathbf{1.02}$
Hybrid	Real-world scenario: $\mathbf{0.95}$

6.4. Scalability and Performance

We are going to give a note on scalability and runtime performance of the algorithms described used in methods and experiments of that document. To start with, scalability of the hybrid approach depends on the base recommenders, since they should be already available when the ensemble learning is processed.

In the scope of this paper, we approach collaborative, item content-based, and user demographics-based filtering problems as supervised learning problems, and train all of them using stochastic gradient descent algorithm. Stochastic gradient descent algorithm is an example of online learning, where the parameters are updated with each training example. This paradigm not only solves a machine learning task simplistically and keeps the model up to date at all time, but also allows horizontally scalable system architecture. Updating one parameter vector (per user or item) at a time, with a simple row lock to ensure isolation to prevent other nodes (or threads) try to update it at the same time, allow multiple threads, or computing nodes, to update the recommender model concurrently.

To give a sense of training performance, in Table 6.7, we report the running time for performing one iteration on training set of 700000 ratings of three different recommenders, and on validation set of 200000 ratings of the hybrid recommender. The reported results are for the experiments that we run on a UNIX personal computer, with 2.8 GHz 4-cores processor, and 8GB of memory, 1 GB of which is spared for Java Virtual Machine heap.

Table 6.7 Training running times of different recommenders per iteration

	Running time per iteration (seconds)
MF-based collaborative	7.2
User demographics-based	3.4
Item content-based	2.9
Hybrid	0.8

7. CONCLUSIONS AND FUTURE DIRECTIONS

A typical recommender system learns a scoring function for a prediction task i, j, and then when a user i demands a list of interesting items, it ranks unrated items based on this function, and lists the top results. When the system is deployed in production, prediction tasks representing different domains might limit its accuracy. For instance, the system can unify various recommender models in a single one to generalize to cold-start data, but that approach would hurt accuracy when high-quality feedback is available for the user and the item involving in the task. To avoid this trade-off, we developed an ensemble of complementary recommenders, which adapts to various, identified domains, for which an accurate recommender technique exists.

Experimental results show that the model we described converges to the most accurate base recommender, say r_d, on the test data set representing the identified domain d. Considering this r_d performs poor for domains other than d, the hybrid model of varying weights outperforms complementary base recommenders on a test data set that includes prediction tasks from multiple domains, which is the case for a recommender in production.

This approach has some limitations, too. The practitioner should carefully identify the prediction task cases, mimic those cases by creating validation data sets, and learn domain-specific weights separately. Besides at runtime, the system should infer the domain of the query, other than user and item.

The model we described learns hybridization weights on separate validation data sets subsequent to learning base recommenders. A further improvement would be to learn domain specific weights and recommenders simultaneously, which would eliminate the relatively complex multi-step learning process the current model introduces.

REFERENCES

[1] Claypool, M., et al. *Combining Content-based and Collaborative Filters in an Online Newspaper*. in *Proceedings of ACM SIGIR Workshop on Recommender Systems*. 1999. ACM.

[2] Balabanović, M. *An interface for learning multi-topic user profiles from implicit feedback*. in *AAAI-98 Workshop on Recommender Systems*. 1998.

[3] Wolpert, D.H., *Stacked Generalization.* Neural Networks, 1992. **5**(2): 241-259.

[4] Jacobs, R.A., et al., *Adaptive Mixtures of Local Experts.* Neural Computation, 1991. **3**(1): 79-87.

[5] Herlocker, J.L., et al., *Evaluating Collaborative Filtering Recommender Systems.* ACM Transactions on Information Systems, 2004. **22**(1): 5-53.

[6] Goldberg, D., et al., *Using Collaborative Filtering to Weave an Information Tapestry.* Communications of the ACM, 1992. **35**(12): 61-70.

[7] Shardanand, U. and P. Maes. *Social information filtering: Algorithms for Automating "word of mouth"*. in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1995. ACM Press/Addison-Wesley Publishing Co.

[8] Resnick, P., et al. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*. in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. 1994. ACM.

[9] Bennett, J. and S. Lanning. *The Netflix Prize*. in *Proceedings of KDD cup and workshop*. 2007.

[10] Bell, R.M. and Y. Koren, *Lessons from the Netflix Prize Challenge.* ACM SIGKDD Explorations Newsletter, 2007. **9**(2): 75-79.

[11] Koren, Y., *The Bellkor Solution to the Netflix Grand Prize.* Netflix Prize Documentation, 2009.

[12] Töscher, A., M. Jahrer, and R.M. Bell, *The Bigchaos Solution to the Netflix Grand Prize.* Netflix Prize Documentation, 2009.

[13] Piotte, M. and M. Chabbert, *The Pragmatic Theory Solution to the Netflix Grand Prize.* Netflix Prize Documentation, 2009.

[14] Breese, J.S., D. Heckerman, and C. Kadie. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. 1998. Morgan Kaufmann Publishers Inc.

[15] Herlocker, J.L., et al. *An Algorithmic Framework for Performing Collaborative Filtering*. in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999. ACM.

[16] Sarwar, B., et al. *Item-based Collaborative Filtering Recommendation Algorithms*. in *Proceedings of the 10th International Conference on World Wide Web*. 2001. ACM.

[17] Joseph, J.H., J.A. Konstan, and J. Riedl. *Explaining Collaborative Filtering Recommendations*. in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. 2000.

[18] Seung, D. and L. Lee, *Algorithms for Non-negative Matrix Factorization.* Advances in Neural Information Processing Systems, 2001. **13**: 556-562.

[19] Koren, Y., R. Bell, and C. Volinsky, *Matrix Factorization Techniques for Recommender Systems.* Computer, 2009. **42**(8): 30-37.

[20] Funk, S. *Netflix Update: Try This at Home*. 2006 [cited 2013; Available from: http://sifter.org/~simon/journal/20061211.html.

[21] Paterek, A. *Improving Regularized Singular Value Decomposition for Collaborative Filtering*. in *Proceedings of KDD Cup and Workshop*. 2007.

[22] Baltrunas, L., B. Ludwig, and F. Ricci. *Matrix Factorization Techniques for Context Aware Recommendation*. in *Proceedings of the 5th ACM Conference on Recommender Systems*. 2011. ACM.

[23] Koren, Y., *Collaborative Filtering with Temporal Dynamics.* Communications of the ACM, 2010. **53**(4): 89-97.

[24] Hu, Y., Y. Koren, and C. Volinsky. *Collaborative Filtering for Implicit Feedback Datasets*. in *Proceedings of the 8th IEEE International Conference on Data Mining*. 2008. IEEE.

[25] Lops, P., M. de Gemmis, and G. Semeraro, *Content-based Recommender Systems: State of the Art and Trends*, in *Recommender Systems Handbook*. 2011, Springer: 73-105.

[26] Schiaffino, S. and A. Amandi, *Intelligent User Profiling*, in *Artificial Intelligence. An International Perspective*. 2009, Springer: 193-216.

[27] Pazzani, M.J., *A Framework for Collaborative, Content-based and Demographic Filtering.* Artificial Intelligence Review, 1999. **13**(5-6): 393-408.

[28] Bonhard, P. and M. Sasse, *'Knowing me, knowing you'—Using Profiles and Social Networking to Improve Recommender Systems.* BT Technology Journal, 2006. **24**(3): 84-98.

[29] Pazzani, M.J. and D. Billsus, *Content-based Recommendation Systems*, in *The Adaptive Web*. 2007, Springer: 325-341.

[30] Balabanović, M. and Y. Shoham, *Fab: Content-based, Collaborative Recommendation.* Communications of the ACM, 1997. **40**(3): 66-72.

[31] Van Meteren, R. and M. Van Someren. *Using Content-based Filtering for Recommendation*. in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 2000.

[32] Mooney, R.J., P.N. Bennett, and L. Roy, *Book Recommending using Text Categorization with Extracted information*, in *Recommender Systems Papers from 1998 AAAI Workshop.* 1998.

[33] Billsus, D. and M.J. Pazzani. *A Hybrid User Model for News Story Classification*. in *Proceedings of the 7th International Conference on User Modeling*. 1999.

[34] Li, Q. and B.M. Kim. *Clustering Approach for Hybrid Recommender System*. in *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*. 2003. IEEE.

[35] Melville, P., R.J. Mooney, and R. Nagarajan. *Content-boosted Collaborative Filtering for Improved Recommendations*. in *18th International Conference on Artificial Intelligence*. 2002.

[36] Basu, C., H. Hirsh, and W. Cohen. *Recommendation as Classification: Using Social and Content-based Information in Recommendation*. in

Proceedings of the 15th National Conference on Artificial Intelligence. 1998.

[37] Park, S.-T. and W. Chu. *Pairwise Preference Regression for Cold-start Recommendation*. in *Proceedings of the 3rd ACM Conference on Recommender Systems*. 2009. ACM.

[38] Ahn, H.J., *A New Similarity Measure for Collaborative Filtering to Alleviate the New User Cold-starting Problem.* Information Sciences, 2008. **178**(1): 37-51.

[39] Park, S.-T., et al. *Naïve Filterbots for Robust Cold-start Recommendations*. in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2006. ACM.

[40] Alpaydin, E., *Introduction to machine learning*. 2004: MIT press.

[41] Yu, K., A. Schwaighofer, and V. Tresp. *Collaborative Ensemble Learning: Combining Collaborative and Content-based Information Filtering via Hierarchical Bayes*. in *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*. 2002. Morgan Kaufmann Publishers Inc.

[42] Tiemann, M. and S. Pauws. *Towards Ensemble Learning for Hybrid Music Recommendation*. in *Proceedings of the 2007 ACM Conference on Recommender Systems*. 2007. ACM.

[43] Kuncheva, L.I., J.C. Bezdek, and R.P. Duin, *Decision Templates for Multiple Classifier Fusion: An Experimental Comparison.* Pattern Recognition, 2001. **34**(2): 299-314.

[44] Friedman, J.H., *Greedy Function Approximation: A Gradient Boosting Machine.* Annals of Statistics, 2001: 1189-1232.

[45] Friedman, J.H., *Stochastic Gradient Boosting.* Computational Statistics & Data Analysis, 2002. **38**(4): 367-378.

[46] Bao, X., L. Bergman, and R. Thompson. *Stacking Recommendation Engines with Additional Meta-features*. in *Proceedings of the 3rd ACM Conference on Recommender Systems*. 2009. ACM.

[47] Sill, J., et al., *Feature-weighted Linear Stacking*. arXiv preprint arXiv:0911.0460, 2009.

[48] Ma, H., I. King, and M.R. Lyu. *Learning to Recommend with Social Trust Ensemble*. in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2009. ACM.

[49] Jahrer, M., A. Töscher, and R. Legenstein. *Combining Predictions for Accurate Recommender Systems*. in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2010. ACM.

[50] Desrosiers, C. and G. Karypis, *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, in *Recommender Systems Handbook*. 2011, Springer: 107-114.