A Hybrid Approach for Cold-start Recommendations of Videolectures

Eleftherios Spyromitros-Xioufis, Emmanouela Stachtiari
Grigorios Tsoumakas, and Ioannis Vlahavas

Department of Informatics
Aristotle University of Thessaloniki, Greece
{espyromi,emmastac,greg,vlahavas}@csd.auth.gr

Abstract. This paper presents the solution which ranked 2nd in the "cold-start" recommendations task of the ECML/PKDD 2011 discovery challenge. The task was the recommendation of new videolectures to new users of the Videolectures.net Web site. The proposed solution is a hybrid recommendation approach which combines content-based and collaborative information. Structured and unstructured textual attributes which describe each lecture are synthesized to create a vector representation with tf/idf weights. Collaborative information is incorporated for query expansion with a novel method which identifies neighboring lectures in a co-viewing graph and uses them to supplement missing attributes. The cosine similarity measure is used to find similar lectures and final recommendations are made by also accounting the coexistence duration of lectures. The results of the competition show that the proposed approach is able to give accurate "cold-start" recommendations.

1 Introduction

Recommender systems are designed to suggest items which are predicted to be interesting to users, based on some evidence. This technology has allowed for businesses on the web to keep a sense of a local shop where customers are familiar to the owner, while targeting at a global market. Recommender systems filter items to support users to easier decide what to buy. For an e-commerce, like Amazon.com, providing personalized suggestions of products leads to a better alignment with the designed sales' policy, which could aim at augmenting the sales, or enlarging their market. Web sites that don't make profit out of products can also benefit from a recommender system which attracts users by addressing their specialized needs. Examples of such applications include recommending movies at Grouplens.org, videos at Youtube.com etc. Except for identifying which items to recommend, it is also important to determine a ranking for displaying those items, since the top displayed recommendations are more likely to be viewed or visited.

Videolectures.net is an online repository of video lectures which took place at scientific events like conferences, summer schools, workshops etc. Its goal is to promote science ideas by providing high quality didactic content to the scientific

community and to the general public. All lectures, accompanying documents, information and links are systematically selected and classified through the editorial process taking also into account users' comments. ECML/PKDD 2011 discovery challenge was organized in order to improve the Web site's current recommender system. The first task of the challenge is tackled here and it simulates a new-user and new-item recommendation mode, the so-called "cold-start" recommendations problem.

There are two main categories of recommender systems. *Collaborative filtering* methods [3, 8, 2] make use of the observed activity of users in terms of rating, viewing, or buying items, in order to recommend to a user those items that were appreciated by other similar (or *neighboring*) users. *Content-based* or *information filtering* methods [7, 9, 10] recommend items with descriptive characteristics which match user's taste or a given query. Many hybrid systems [4, 2] have also been developed combining collaborative and content-based methods.

Collaborative filtering systems can recommend items even when nothing is known about their description, which in many cases may not be available or may be extremely noisy. However, it gives poor recommendations to infrequent, new, or anonymous users, because their observed activity is small or nonexistent. They also fail to address "unusual" users (neighboring users may not be found) and "unusual" items (they may have no ratings yet). Regarding content-based techniques, a known advantage over collaborative filtering is that they perform well in "cold-start" situations: they deal with new users by recommending items with similar description to a query item. Another strength is that they are indifferent to the frequency of the selection of items, so new (or rare) items will also be returned. Among its drawbacks is that performance depends a lot on feature selection and content assignment to the items, which for some domains (like multimedia) requires advanced methods.

The solution proposed here is mainly content-based: for a query lecture we recommend lectures that are similar in their descriptive features, taking also into account the duration that they coexisted in the Web site. We deal with the problem of missing attributes in queries by a *query expansion* method, which introduces collaborative information in the method. Missing attributes are replaced with the corresponding attributes of the most neighboring lectures in a co-viewing graph.

The rest of the paper is organized as follows. Section 2 refers to related work on recommender systems. Section 3 gives an overview of the task and introduces the evaluation system that we developed in order to assess the performance of our method. Section 4 describes the given solution and finally Section 5 concludes this paper.

2 Related Work

A variety of collaborative filtering techniques have been developed [3, 5, 8]. Typically these techniques compute similarity scores between pairs of users and give recommendations for a user by taking into account the feedback of other users

proportionally to their similarity to the given user. As a measure of similarity, correlations of the feedback of users have been used in [8]. An alternative to the typical approach is an item to item collaborative filtering algorithm which was presented in [5]. This technique keeps an item to item similarity matrix, in which items that tend to be purchased by common customers have high similarity. Upon a recommendation request, the algorithm first aggregates items that are similar to each of the user's purchases and ratings and then recommends the most popular or correlated items. Our query expansion method, being also based on item to item collaborative information, differs in that we form a graph instead of a matrix. This representation allows us to apply Dijkstra's shortest path algorithm to find similar items. These items are not recommended (since the recommendations should come from a different pool of items) but used to expand the query item.

Pure content-based systems rely on content of items to make recommendations [7, 10]. For example, the authors in [6] suggest text-categorization of movie synopses in the domain of movie recommendation. They also examined the use of semantically richer representations than the standard bag of words representation, such as phrases instead of words. Another approach [10] builds a discriminative classifier for each user profile, using a factored model as its prior, where different factors contribute in different levels. Opposite to collaborative filtering, content-based systems can even recommend previously unrated items to users without any observed behavior in the system, and perform better in cases that users have particular interests.

Some hybrid systems aim at combining collaborative with content information in the features of each example and then provide recommendations using content-based methods. For example, experiments for movie recommendation were reported in [2] where features were drawn from content and user ratings and an inductive rule learner was applied. Other hybrid methods augment the existing feedback using content-based techniques and then produce recommendations through collaborative methods. Such an approach in the movie recommendation domain [4] tackles sparsity of existing feedback by generating ratings in an automatic manner using content-based agents. Our method resembles the first example, since it is mainly content-based and it exploits some collaborative information to expand the content of queries if needed.

3 Task Description

3.1 Task Overview

The solution of the "cold-start" recommendations task should deal with the "cold-start" problem, in the sense that new lectures should be recommended to new users. The scenario assumes that each user has watched only one lecture from a set of old lectures which are lectures published at an early stage of the site's life. Given this old lecture as query, the task is to return a ranked list of 30 similar lectures from a set of new lectures. New lectures are considered to be unseen at the time of recommendation.

3.2 The Given Data

The given data contains two disjoint sets of lectures: the test and the training lectures. All the test lectures have been published in the site after July 1st, 2009. The majority of the training lectures were published before July 1st, 2009, with a smaller subset of lectures having been published after that date. A subset of the training lectures were selected to form the set of query lectures which are all published prior to July 1st, 2009.

Lecture co-viewing information is also given in a table which contains the pairwise co-viewing frequencies for the lectures of the training set. In general, all lecture co-viewing frequencies were taken on July 2010. By applying the train/test split on July 1st, 2009, the split is both "vertical" (all test lectures are published after July 1st, 2009) and "horizontal" (the training set contains approximately half of the lectures published after July 1st, 2009). As we will discuss in Section 4.4, this split allows learning the temporal impact on lecture co-viewing frequencies from the training set.

For each lecture we have information about its language, event type, parent event, date of recording, publication date, name description, slide titles, category/ies and author/s. For the training lectures, the total number of views is also given. Except for lectures we also have information on events and the event taxonomy used to group lectures. Table 1 gives the details of the database tables which contain the given data.

Table 1. Details of the given data.

Table name	Description
authors	Contains data on 8,092 authors registered on Videolectures.net and their information. However, not all authors are assigned to a lecture.
authors_lectures	Contains pairwise information on which author authored which lecture or event. A single author can author multiple lectures (or events), and one lecture (or event) can be authored by multiple authors.
categories	Contains information on categories in scientific taxonomy used on Videolectures.net in a pairwise manner (parent and child pairs). The taxonomy is a direct acyclic graph (several categories have multiple parent categories). Only the root category does not have a parent. There are 348 distinct categories.
categories_lectures	Contains information on pairs of categories and assigned lectures (or events). Some lectures (or events) belong to more than one categories.
events	Contains information on events and the event taxonomy used to group lectures. The taxonomy is a forest (a disjoint union of trees) since: a) each lecture is part of only one event, b) an event has only one parent and c) there are root events that do not have a parent. Events contain a set of lectures rather than videos. There are 519 distinct events.
lectures_train, lectures_test	Contain information about the 6,983 training and the 1,122 test lectures.
pairs	Contains records about pairs of lectures viewed together (not necessarily consecutively) with at least two distinct cookie-identified browsers. There are 363,880 distinct pairs.
task1_query	This is the query file for the "cold-start" recommendations task. It contains only lecture ids from the subset of the lectures_train table, for which a recommended ordered list of 30 lectures from the lectures_test table is expected as a submission. There are 5,704 query lectures.

3.3 Evaluation method

Taking into account the scarcity of items available for learning, recommending and evaluation in the "cold-start" recommendations task, the challenge organizers defined an evaluation measure called mean average R-precision (MARp), inspired from standard information retrieval measures. Given q query lectures, a set of solution lists $S = s_1, \ldots, s_q$ and recommended lists $R = r_1, \ldots, r_q$ for these lectures and a set of cut-off lengths $Z = 5, 10, 15, 20, 25, 30$, this measure is defined as:

$$\mathrm{MARp}(S, R) = \frac{1}{q} \sum_{i=1}^{q} \mathrm{AvgRp}(s_i, q_i, Z), \tag{1}$$

where for a given solution list s, recommended list r and set of cut-off lengths Z, the average R-precision (AvgRp) is defined as:

$$\mathrm{AvgRp}(s, r, Z) = \sum_{z \in Z} \mathrm{Rp@}z(s, r), \tag{2}$$

where for a given solution list s, recommended list r and cut-off length z the R-precision at this cut-off length ($\mathrm{Rp@}z(s, r)$) is defined as:

$$\mathrm{Rp@}z(s, r) = \frac{|s^z \cap r^z|}{min(|s|, z)}, \tag{3}$$

where l^z denotes a list containing the first z elements of list l.

The preliminary results, comprising of randomly sampled 20% of the final results, are evaluated after submission and published on a leaderboard, allowing comparison with other participants. The final results are scored on the full test dataset.

3.4 Internal Evaluation

In order to be able measure the performance of our recommender we developed an internal evaluation system which allowed us to experiment with variations of our approach and tune its parameters without submitting results to the leaderboard (only 60 submissions were allowed in total). To simulate the "cold-start" recommendations task, we split the given training lectures in two sets. The first set contained all the lectures of the original training set which had been published prior to July 1st, 2009 and formed the new training set. The second set contained the rest of the lectures of the original training set (published after July 1st, 2009) and formed the new test set. The set of query lectures was the same as in the original task, since all the query lectures appear prior to July 1st, 2009 and were all contained in the new training set. Given a query lecture, we recommended the 30 most relevant lectures from the new test set. The ground truth was created using the co-viewing information which was available in the pairs table (described earlier). Specifically, for each query lecture, we found the (at most) 30 test lectures with which it had the highest co-viewing frequency and

ranked them in descending order according to co-viewing frequency. The AvgRp measure was calculated by comparing our recommendations to the ground truth and finally averaged over all query lectures to get the MARp score. It was found that the accuracy results obtained using our internal evaluation system were (in most cases) quite close to the final evaluation results. In the following section we refer to variations we tried and parameters we tuned using our evaluation system without, however, giving the exact evaluation results since they were not recorded.

4 Our Solution

4.1 Basic Recommendation Model

We tackled the "cold-start" recommendations problem by using a well-known content-based recommendation technique which has its roots in the theory of Information Retrieval and is known as the *vector space model* [1]. Each lecture was represented as a text document by synthesizing various sources of textual information related to it. Each document was then transformed into a vector of size k where k is the total number of distinct terms (words) in the whole collection of documents (the union of the test and the query lectures). To measure the importance of each term inside a document, we used *term frequency/inverse document frequency* (tf/idf) weights:

$$TF_{t,d} = \frac{f_{t,d}}{max_x\{f_{x,d}\}} \tag{4}$$

where $f_{t,d}$ is the frequency of the term t in document d and $max_x\{f_{x,d}\}$ is the maximum frequency of a term in that document.

$$IDF_t = log\frac{N}{n_t} \tag{5}$$

where N is number of documents in the collection and n_t is the number of documents with the the term t.

The tf/idf weight for a term t in document d is defined as:

$$w_{t,d} = TF_{t,d} \cdot IDF_t \tag{6}$$

In order to measure the similarity between two vectors q and d we used the cosine similarity:

$$S_{cosine}(q,d) = \frac{\sum_{i=1}^{k} w_{t_i,q} w_{t_i,d}}{\sqrt{\sum_{i=1}^{k} w_{t_i,q}^2}\sqrt{\sum_{i=1}^{k} w_{t_i,d}^2}} \tag{7}$$

The above formulas for calculating the tf/idf weights combined with the cosine similarity were found to give the best results among other variations that we tried.

4.2 Synthesis of Textual Attributes

In order to create the document representation of each lecture we synthesized the various textual attributes related to it, which were distributed among the given database tables. The used attributes fall into two categories: *unstructured* text attributes (name, description, slide-titles) and *structured* text attributes with a known closed set of values (event type, language, parent event id, category/ies, author/s). We found that this semi-structured representation, which included both attributes with restricted values and unstructured text attributes worked better than using unstructured text alone (the typical approach).

Structured and unstructured text attributes were treated differently in terms of preprocessing. To preprocess the unstructured attributes, we first removed any non alphanumeric characters. Then, we used an English stop-word list to filter out common terms and removed terms with less than 2 or more than 20 characters (this allowed us to get rid of long DNA sequences in the descriptions of some biology/genetics videos). We also removed terms consisting only of numbers. Stemming of English words was applied without improvement in the results which can be attributed to the fact that the collection included non-English documents which were improperly stemmed. Perhaps, applying stemming only to the English documents or using language specific stemmers would produce better results. Filtering out infrequent terms performed worse than keeping all the available terms.

A different type of preprocessing was applied to the structured attributes. Their values were prefixed with the attributes' names. For example, the value "education" of the category attribute was substituted by "category_education". This substitution was performed in order to distinguish a term inside a lecture's name or description from the same term as the value of a structured attribute. For example, the term "education" in the title of the lecture "Women in university education", which refers to gender issues, should be distinguished from the same term as a lecture's category.

Next, we give a more detailed description of the structured attributes:

- *Parent event id.* The parent event to which the lecture belongs. While the sets of query and new lectures are disjoint, it may happen that a query and a new lecture share the same parent event. This is considered a piece of information contributing to the similarity between two lectures.
- *Lecture type.* The specific type of the lecture, which could be one of the following: lecture, keynote, debate, tutorial, invited talk, introduction, interview, opening, demonstration video, external lecture, thesis proposal, best paper, panel, advertisement, promotional video, thesis defence, summary.
- *Language.* The language of the lecture. Although the majority of the lectures in the collection were in English, there were also non-English lectures (699 out of 6983 in the training set and 213 out of 1122 in the test set) belonging to 10 different languages. This attribute was included in order to increase the probability of recommending lectures of the same language.
- *Category/ies.* The categories under which a lecture has been categorized. Obviously, lectures belonging to the same category are likely to be similar.

We also tried including the ancestors of the actual categories into the textual representation of lectures. This was based on the intuition that two lectures belonging to categories which share a common ancestor are probably more similar than two lectures whose categories have no common ancestors. Although intuitively rational, this variation did not improve the evaluation results.
- *Author/s.* The authors of the presentation related to each lecture. Users are often interested in lectures of the same author.

A description of the unstructured attributes is given here:

- *Name.* The name of the lecture or event in natural language. Terms in lecture names are usually highly descriptive (e.g. "Research on position of women in science" and "Women in technical sciences research". However, some times names are misleading (e.g. "Where did they all go?").
- *Description.* The description of the lecture or event in natural language. Note that not all lectures/events are given a description. However, it is expected to be a very informative attribute.
- *Slide titles.* The titles of the slides accompanying the lecture. Note that slide titles are not available for all lectures. Usually slide titles in the beginning and the end of a presentation ("Introduction", "Conclusions") are not as informative as the titles in the middle. However, the tf/idf scheme will assign small weights to terms which are frequent in all documents.

One can notice that some attributes contribute more than others to the similarity between two lectures. For example, a lecture having the same author and category with a query lecture should be favored as a recommendation compared to a lecture that only shares some common terms with the query in its description. In order to take advantage of this intuition and to compensate for the large number of terms in the unstructured attributes compared with the few terms of the structured attributes, we assigned a different weight to each attribute by repeating its terms in the textual representation of each query lecture. The final weights were tweaked using the internal evaluation system. The terms of parent event id, lecture type, language, category/ies, author/s and name were repeated sixteen times, the terms of description four times and the terms of slide titles one time.

4.3 Query Expansion

We noticed that some query lectures had missing attributes (descriptions, slide-titles, authors and/or events). This resulted in uncertain recommendations due to the sparsity of the tf/idf vectors. We tried to tackle this problem by using neighboring lectures to enrich the original queries. The lecture pairs table was used for this purpose. The pairs contained in this table involve only lectures from the training set, thus the co-viewing information can not be used for recommending new lectures. However, this information can be used for identifying training lectures which are frequently co-viewed with query lectures and are thus assumed to have similar content.

Finding Neighbors To find neighboring lectures, we construct a co-viewing graph where the training lectures represent the vertices. For every pair of lectures in the pairs table we add an undirected edge connecting the lectures of the pair. The weight of the edge is equal to the pair's frequency. These edges show the strength of connection between two nodes or the likelihood of moving from one lecture to another. A straightforward approach to find the nearest neighbors of a query lecture in the graph, would be to find all the lectures which are connected to the query with some edge and then rank them in descending order according to edge weight. This approach identifies only lectures which are directly connected to the query as neighbors. However, there are cases where two lectures have very low or zero co-viewing frequency but have both been co-viewed many times with a third lecture. With the previous approach these two lectures would not be returned as neighbors, although it is likely that they are similar. In order to overcome this problem we developed a method which is based on *Dijkstra*'s shortest path algorithm and is able to identify neighbors even if they are not directly connected to the query lecture. Since Dijkstra's algorithm requires cost (distance) edges, we apply a transformation to the weights of the edges. This is done by first finding the weight of the edge with the largest weight $\max w$ in the original graph and then using the formula shown in Equation 8 where $w(x, y)$ is the weight of the edge connecting the vertices x and y before the transformation and $w'(x, y)$ is the transformed weight.

$$w'(x, y) = \max w - w(x, y) + 1 \tag{8}$$

Given a source vertex (lecture), Dijkstra's algorithm finds the shortest path between that vertex and every other vertex. The algorithm guarantees that in its k-th iteration, the shortest paths between the source and the k nearest vertices have been identified. Since we are interested only in the k nearest neighbors, we stop the algorithm on its k-th iteration, thus achieving a small execution time. In our internal evaluation we found that this way of finding nearest neighbors in the co-viewing graph yielded better results than the straightforward approach and we therefore used it in our recommender.

Using Neighbors Two different ways to use the nearest neighbors for expanding the original query were tested. In the first approach, the query lecture was expanded by including all the attributes of its nearest neighbors. The evaluation showed that we could obtain better results with this approach compared with using only the original query. Even better performance was achieved by assigning larger weight to the attributes coming from the original query than the attributes coming from the query's neighbors. A degradation in performance was observed when more than two nearest neighbors were considered. This is attributed to the fact that including attributes from distant neighbors adds noise to the query.

In the second approach, instead of expanding all the query lectures with information from their nearest neighbors we tried to do that selectively, only in the cases where the original query was missing some attributes. For example, if

the original query had no description assigned, we looked for a description at its nearest neighbor. The process was repeated until a neighbor with description was found or until the distance of the neighboring lecture to the original query lecture exceeded a fixed threshold. The threshold was used to ensure that missing attributes will be supplemented using attributes of really close neighbors. This approach outperformed the previous one.

4.4 The Temporal Effect

An important factor for improving the evaluation score was to consider how the ground truth was generated. Ideally, recommender systems success should be measured through user satisfaction analysis. For the challenge, a quantitative measure was needed. In order to be able to score solutions, the organizers took a snapshot of the Videolectures.net database on July 2010 and lecture co-viewing frequencies were recorded for all lectures (both train and test) at that moment. The list of relevant lectures for each query lecture was created by ranking the test lectures in descending order according to withheld lecture co-viewing frequencies.

By observing co-viewing frequencies in the training data (no co-viewing information was available for the test data), we noticed that the duration of coexistence between lectures had an impact in the frequencies. In fact, the more the time that two lectures coexisted in the site the more likely it was to have a high co-viewing frequency. Intuitively, a lecture that was published in Videolectures.net just one week before the day that the snapshot was taken could not have had many co-views with any of the other lectures, even with the ones most similar to it.

Algorithm 1: Make recommendations

Input: a query lecture l_q, the set of test lectures L_T, k
Output: the sorted list of recommended lectures L_R

// Find the k most similar items to l_q based on the cosine similarity S_{cosine} and initialize the recommended list.
$L_R \leftarrow kNearestNeighbors_{cosine}(l_q, L_T)$
foreach $l_t \in L_R$ **do**
    Calculate the coexistence-based similarity $S_{duration}(l_q, l_t)$ between l_q and l_t
    $S_{total}(l_q, l_t) \leftarrow S_{duration}(l_q, l_t) \cdot S_{cosine}(l_q, l_t)$
Sort L_R on S_{total}
Return the top 30 items of L_R

To account the impact of both the content-based similarity and the impact of the coexistence duration in co-viewing frequencies, we used the algorithm listed in Algorithm 1. We first find the k nearest neighbors of the (expanded) query in the test set according to the cosine similarity. k is a parameter of the algorithm with values ranging between 30 and $|L_T|$ where L_T is the test set. Then, we

multiply the cosine similarity between the query and each one of its k nearest neighbors with a similarity based on their coexistence duration, to get a total similarity score. The coexistence-based similarity is just the time (in ms) that two lectures coexisted divided by the maximum coexistence duration (approximately one year) between a query lecture and a test lecture. Both similarities are normalized to the [0,1] scale. To make the final recommendations, we sort the list of neighbors on the combined score and return the top 30 lectures.

The value for the parameter k was tweaked using the internal evaluation system and the best results where obtained with $k = 40$. By taking the temporal effect into account, the overall performance was increased by 10%.

5 Conclusions and Future Work

In this paper we presented the recommendation system that we developed for the "cold-start" recommendations task of the ECML/PKDD 2011 Discovery Challenge. The system uses a traditional content-based filtering technique to recommend similar lectures, based on both structured and unstructured attributes related to each lecture. Collaborative information is also incorporated to the system using a novel method which identifies neighboring lectures in terms of co-viewing and uses them to supplement missing attributes. Finally, temporal aspects are studied and taken into account to produce the final recommendations. The results of the competition show that the proposed hybrid approach is able to produce accurate recommendations in "cold-start" situations. We expect that better results can be obtained if the coexistence duration between pairs of lectures is taken into account in the process of finding neighbors in the co-viewing graph. It would also be interesting to examine whether a better way to combine the content-based similarity and the coexistence-based similarity could learned from the training data.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
2. Basu, C., Hirsh, H., Cohen, W.W.: Recommendation as classification: Using social and content-based information in recommendation. In: AAAI/IAAI. pp. 714–720 (1998)
3. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35, 61–70 (December 1992)
4. Good, N., Schafer, J.B., Konstan, J.J., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: Proceedings of the 1999 Conference of the American Association of Artifical Intelligence (AAAI-99) (1999)
5. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing 7, 76–80 (January 2003)
6. Mak, H., Koprinska, I., Poon, J.: Intimate: A web-based movie recommender using text categorization. Web Intelligence, IEEE / WIC / ACM International Conference on 0, 602 (2003)

7. Pazzani, M., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The adaptive web, Lecture Notes in Computer Science, vol. 4321, pp. 325–341. Springer, Berlin / Heidelberg (2007)
8. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work. pp. 175–186. CSCW '94, ACM, New York, NY, USA (1994)
9. Wang, Y., Wang, S., Stash, N., Aroyo, L., Schreiber, G.: Enhancing content-based recommendation with the task model of classification. In: Cimiano, P., Pinto, H.S. (eds.) EKAW. Lecture Notes in Computer Science, vol. 6317, pp. 431–440. Springer (2010)
10. Zhang, L., Zhang, Y.: Discriminative factored prior models for personalized content-based recommendation. In: Huang, J., Koudas, N., Jones, G.J.F., Wu, X., Collins-Thompson, K., An, A. (eds.) CIKM. pp. 1569–1572. ACM (2010)