Using Co-views Information to Learn Lecture Recommendations

Haibin Liu, Sujatha Das, Dongwon Lee, Prasenjit Mitra, C. Lee Giles

The Pennsylvania State University
{haibin@,gsdas@cse.,dongwon@,pmitra@ist.,giles@ist.}psu.edu

Abstract. Content-based methods are commonly adopted for addressing the cold-start problem in recommender systems. In the cold-start scenario, usage information regarding an item and/or item preference information of a user is unavailable since the item or the user is new in the system. Thus collaborative filtering strategies cannot be employed but instead item-specific attributes or the user profile information are used to make recommendations. We focus on lecture recommendations for the data in videolectures.net that was made available as part of the ECML/PKDD Discovery Challenge. We propose the use of co-view information based on previously seen lecture pairs for learning the weights of lecture attributes for ranking lectures for the cold-start recommendation task. Co-viewed triplet and pair information is also used to estimate the probability that a lecture would be seen, given a set of previously seen lectures. Our results corroborate the effectiveness of using co-view information in learning lecture recommendations.

1 Introduction

Given a set of users and a set of items, the goal of a recommender system is to predict the items a particular user is most likely to be interested in. Recommending products for users on a shopping website like Amazon, predicting the ratings that a user is likely to assign to a movie, predicting the citations a paper is likely to make are some common scenarios where automatic recommender systems are desirable[11, 4, 14] [1].

We focus on lecture recommendations for lectures from `videolectures.net`, an open-access repository of educational lectures [2]. Lectures given by prominent researchers and scholars at conferences and other academic events are made available on this website for educational purposes. This year's ECML/PKDD Discovery Challenge involved two recommendation tasks using lectures from this website. Figure 1 denotes a snapshot of the existing system at `videolectures.net`. We indicate in this figure some of the information available with lectures on this website. Most lectures on this website contain information on the language in which the lecture was given, content of the slides, the category (discipline-area) of the lecture etc. Sometimes, additional information such as the description of the event (such as conference, workshop) in which the lecture was given and author affiliation is also available. The training data for ECML/PKDD challenge contains a subset of lectures from `videolectures.net`. Along with the lecture, authors and event attribute information, the training data includes information about pairs and

[1] Partially supported by NSF DUE-0817376 and DUE-0937891 awards.
[2] http://videolectures.net/site/about/

triples of lectures that were frequently co-viewed in the past. The datasets are described in more detail in Section 3 and [3].

Task 1 of the challenge pertains to the cold-start scenario where recommendations are sought for new lectures. In this task, we are given a set of training lectures Q, and a co-viewed pairs set $P = \{(l_1, l_2, f) \mid l_1, l_2 \in Q\}$, where f is the frequency that l_1, l_2 were co-viewed together. The test set, T contains lectures without any viewing history and the task requires the participants to recommend lectures, $R_q \subset T$ for each query lecture $q \in Q'$, $Q' \subset Q$. This task simulates the scenario in which recommendations are to be made for a new user or a new lecture where no co-viewed history information is available.

Task 2 of the challenge simulates a typical scenario for recommender systems. For this task, recommendations are sought on what lectures are likely to be viewed next given three lectures of a stream of previously viewed lectures. The training data for this task includes triplets $T_{left} = \{(t_{id}, l_1, l_2, l_3, f_l) \mid l_1, l_2, l_3 \in Q\}$ where Q is the set of lectures, f_l is the frequency that l_1, l_2, l_3 appeared together in click streams and t_{id} an identifier for the triplet. The data also includes the set of lectures that have the highest co-view frequencies, $T_{right} = \{(t_{id}, l, f_r) \mid l \in Q\}$ where f_r is the co-view frequency of the lecture l with the triple t_{id}. Given a list of query triplets T_{left}^{query}, task 2 involves predicting lectures that are most likely to be viewed next given that each lecture in the triplet $t \in T_{left}^{query}$ was seen.

Our solutions to task 1 and 2 make use of the lecture co-view information available in the training data. We adopt a content-based approach for the cold-start scenario of task 1 where co-view information is used to learn the feature weights for ranking lectures for the recommendation task. We use the co-viewed lecture pairs to form training instances for a supervised learning setup. Support Vector Machines were used where the learnt feature weights indicate the importance of each lecture attribute for recommending lectures in the cold-start scenario. For task 2 that involves making recommendations based on a set of previously seen lectures, we propose a scoring technique to estimate the likelihood that a lecture would be seen next using concepts from item-set mining. Our solutions based on the above strategies performed on par with the top-performing systems in the Discovery challenge. In the final rankings on the leader board, our system was ranked 8th among 62 participants for task 1 and 4th among 22 participants for task 2.

The remainder of this paper is organized as follows: We briefly summarize previous work most related to our approach in Section 2. Section 3 describes our solution and experiments related to task 1 where as in Section 4 we describe our algorithm for task 2 and its performance. Section 5 concludes the paper.

2 Related Work

Recommendation strategies can be broadly classified into collaborative filtering and content-based strategies. We briefly describe the basic ideas behind these approaches and include references to some surveys for further understanding. Collaborative filtering (CF) methods use previous item-user history to generate lists of recommendations. For example, in movie recommendations, CF strategies use movie ratings previously submitted by other users to predict the rating a user might assign to a movie based on user-similarity or movie-similarity [1]. In addition to historical information, a user's or item's properties and attributes can be used for personalized recommendations using

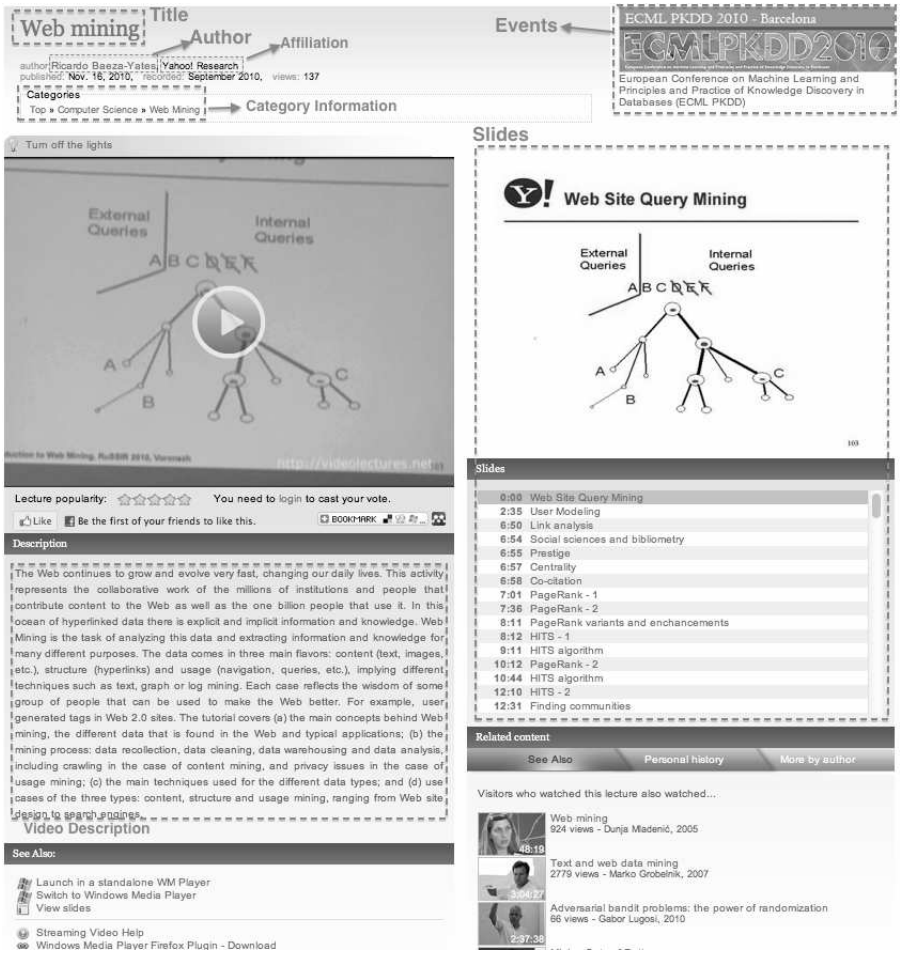

Fig. 1. Lecture video attributes

content-based approaches [13]. Content-based methods are common in addressing the `cold start` problem where ratings and preference information is unavailable or sparse.

Recently, hybrid strategies are being used to leverage the benefits of both collaborative filtering and content-based strategies. For example, to tackle the cold start problem, Gantner, et al. used collaborative information to compute similarity between existing items or users using matrix factorization, and then proposed mapping techniques like a linear combination of various attributes of new items to fit content into same model [8]. Our techniques for learning feature weights for content-based recommendations is closest to the techniques adopted by Strohman, et al. [14] and Debnath, et al [7]. As opposed to the regression framework adopted by them, we formulate the attribute-weight learning problem in a classification framework for cold-start recommendations for task 1. For task 2, we design an algorithm that makes use of the fundamental concepts of `support` and `confidence` from item-set mining [2].

3 Task 1: Learning Attribute Weights for Cold Start Recommendations

We briefly summarize the attribute information available with the data provided for the challenge in Table 1.

Table 1. Description of Tables in the dataset

Table Name	Attribute Information
authors	id, name, e-mail, homepage, gender, organization
categories	id, name, parent-id, wikipedia-url
lectures	id, type, lang, parent-id, views, rec-date, pub-date, name, desc., titles
events	id, type, lang, parent-id, views, rec-date, pub-date, name, desc., titles
pairs	lec-id1, lec-id2, frequency
triplets-left	lec-id1, lec-id2, lec-id3, freq , pooled sequences of 3 lectures
triplets-right	top 10 lectures with highest view frequencies for triplets

3.1 Design of Features

The pairs information available for task 1 indicates the frequency with which a given lecture pair was co-viewed. This information is very significant in understanding the features that a pair of lectures that tend to be co-viewed often share. For instance, it is reasonable to expect that a highly co-viewed pair of lectures are in the same language and perhaps in the same category. Similarly, a pair that is co-viewed frequently is likely to be on related topics such as two lectures presented in the same conference or two parts of a tutorial on a topic. It is also intuitive to expect the co-view frequencies of lectures belonging to diverse categories such as Graph Theory and Ecology to be small. Based on the above intuitions, we designed the set of following features to measure the similarity between two lectures in terms of their attributes.

1. **Co-author similarity** This feature indicates whether two lectures have the same author. It has a value 1 when two lectures share the same author and 0 otherwise.
2. **Type similarity** This feature has a value 1 when two lectures share the same type and 0 otherwise. Example lecture types include lecture, keynote, thesis proposal, tutorial etc.
3. **Language similarity** has a value 1 when the two lectures are in the same language and 0 otherwise.
4. **Event similarity** A value of 0 or 1 indicates whether the two lectures belong to the same event such as conference, workshop series etc. In addition to using the above boolean-valued feature, we used the description fields associated with events to compute a similarity value using the cosine similarity function. This score is meant to capture events that are similar though not the same. For instance, the conferences ECML and ICML are related despite being distinct venues since they are both machine learning conferences. Similarly, lectures belonging to the same conference venue but presented in different years are related.

5. **Category similarity** The category information pertains to the subject area assigned to a lecture. The categories used by `videolectures.net` are those available in Wikipedia. Connections between categories are captured via a directed graph in Wikipedia and can be used to compute similarity. For instance, if two lectures are assigned the categories "Computer Science" and "Graph Theory", they share some commonality since "Graph Theory" is a sub-category of "Computer Science". To capture this aspect, we used four different binary indicators for capturing category similarity between two lectures l_1, l_2:
   - *C1:* 1 if $l_1.categories \cap l_2.categories \neq \emptyset$ and 0 otherwise.
   - *C2:* 1 if $l_1.categories \cap l_2.parent\ categories \neq \emptyset$ and 0 otherwise.
   - *C3:* 1 if $l_1.parent\ categories \cap l_2.categories \neq \emptyset$ and 0 otherwise.
   - *C4:* 1 if $l_1.parent\ categories \cap l_2.parent\ categories \neq \emptyset$ and 0 otherwise.
6. **Text similarity** The name, titles and description fields of a given lecture have textual content. We represent these fields using TFIDF [12] vectors and use the cosine similarity of the corresponding fields of two lectures to compute these features.
7. **Topic similarity** We use Latent Dirichlet Allocation (LDA) [5], a popular tool used for modeling documents as topic mixtures. The generative process in LDA expresses each document in terms of its topic proportions. We modeled the training set of lectures (name+description+titles) using 1000 topics and obtained the topic proportions for each lecture. Similarity between a pair of lectures can be computed using the cosine similarity between the topic vectors or by measuring the overlap among the top topics from each lecture. We used Jaccard Coefficient [12] to compute the similarity score based on the overlap among the top-10 topics of the two lectures.
8. **Affiliation similarity** The author affiliation information is also available with lectures. We compute the affiliation similarity between two affiliations with the Jaccard similarity measure on the set of words describing the affiliation.

3.2 Learning attribute weights for pairwise prediction

Support Vector Machines (SVM) is a discriminative supervised learning approach widely used for classification and regression problems in several areas. For binary classification where the set of class labels is restricted to +1 and -1, the SVM learns a maximally separating hyperplane between the examples belonging to the two classes based on the training data. During testing, the distance between a given instance and this hyperplane is computed and used to assign a prediction label. We formulate the recommendation task for the cold start scenario as a binary classification problem. We treat the co-viewed lecture pairs available in the training data as positive examples for the classification problem. Negative instances for training the classifier are obtained by randomly selecting lecture pairs that were never co-viewed (in the training data). The features described in Section 3.1 were used to train a SVM classifier. Task 1 includes query lecture ids (from say, the set Q) for which recommendations are to be predicted from the set of given test lectures (say, set T). We used each $q \in Q$ to form a pair with each $t \in T$ and score the pair using the trained SVM classifier, namely the distance from this lecture pair instance to the hyperplane. The final list of predictions for each query is obtained by sorting the pairs based on these scores and choosing the test lectures corresponding to the top pairs.

When trained with the linear kernel option, SVMs learn a set of weights that satisfy the maximum number of constraints of the following form imposed by the training data:

$$y_i(\boldsymbol{w}.\boldsymbol{x}_i - b) \geq 1 - \epsilon_i, 1 \leq i \leq n$$

In the above formula, i is the index over the training examples, $\boldsymbol{x}_i$ pertains to the features of a given example, y_i its label (+1 or -1) [6]. In our case, the feature values refer to similarity values based on different attributes of a given lecture pair. That is, as part of learning the classifier, we are in effect, learning a scoring function for lecture pairs (l_i, l_j) based on a linear combination of individual attribute similarity values such that

$$score(l_i, l_j) = \sum_{f=1}^{F} w_f \times sim_f(l_{fi}, l_{fj})$$

where w_f indicates the weight assigned to the similarity value based on a particular attribute f of the given lecture pair.

3.3 Experiments and Observations

The challenge uses R-precision variant for evaluating recommendation performance a mean value over all queries R defined as:

$$MAR_p = \frac{1}{|R|} \sum_{r \in R} AvgRp(r)$$

Here average R-precision for a single recommended ranked list is given by $AvgRp = \sum_{z \in Z} \frac{Rp@z(r)}{|Z|}$ where $Rp@z(r) = \frac{|relevant \cap retrieved|_z}{|relevant|_z}$ the R-precision at some cut-off length z where $z \in 5, 10, 15, 20, 25, 30$ for task 1 and $z \in 5, 10$ for task 2.

For training the SVM classifier for task 1, from the training set P we filtered out pairs that occurred with a frequency less than 5% (for either lecture in the pairs): $P' = \left\{(l_1, l_2, f) \mid (l_1, l_2, f) \in P\ ,\ \frac{f}{S(l_1)} \geq 0.05\ \vee\ \frac{f}{S(l_2)} \geq 0.05\right\}$, where $S(l_1) = \sum_{(l_1, l_i, f_{1i}) \in P} f_{1i}$, $S(l_2) = \sum_{(l_j, l_2, f_{2j}) \in P} f_{2j}$. These were assigned the class label +1. For negative instances, we randomly selected lecture pairs of a comparable size to P' that do not appear in the training pairs set P. In total we had a balanced data set with about $40,000$ pairs for training the classifier. We used the SVMLight [9] implementation provided by Joachims. We set the margin-loss penalty parameter C to 10 after experimenting with values between 0.1-100. The performance on a validation set was the best for C values ranging between 5-20. To show the stability of feature weights we show their mean and variance over five-folds of training runs in Table 2.

As shown in the above table the positive weights for some features such as co-author similarity, event similarity and LDA topic overlap support our intuitions on what attributes are common in lectures that are co-viewed frequently. The negative weights for description and slide content similarity is surprising. We reason that this is possibly due to the fact that a large number of lectures in the training data have empty values for these fields. Similarity based on the concatenated field combining the name, description

Table 2. Feature Weights Learnt By SVM

Feature	Mean	Variance
Co-author similarity	1.895	0.050
Type similarity	-0.087	0.005
Language similarity	0.015	0.003
Event similarity (exact match)	0.779	0.110
Event description similarity	1.421	0.118
Category similarity $C1$	1.889	0.038
Category similarity $C2$	0.114	0.015
Category similarity $C3$	-0.058	0.019
Category similarity $C4$	0.268	0.195
Name TFIDF similarity	8.555	0.125
Description TFIDF similarity	-1.412	0.094
Slide Content TFIDF similarity	-0.360	0.100
All Text fields TFIDF	9.729	0.273
Jaccard similarity based on LDA Top 10 topics	0.329	0.016
Affiliation similarity	0.705	0.189

and slide content fields and the name similarity fields have high positive weight values that are not surprising. Videos belonging to the same event such as lectures from a course series are likely to share a lot of content similarity in their name fields and are also likely to be viewed together. For our final run we discarded features with negative weights and re-trained the classifier based on the remaining features.

The classification setup treated all paired lectures uniformly as positive instances. However, since it is likely that lectures with higher co-view frequencies are most similar, we also tried unequal weighing strategies based on co-view frequencies as a ranking or regression problem using SVM. With similar features in classification, rather than +1 or -1 as class label, we defined different target values based on co-view frequencies of pairs for regression and ranking setup. For regression setting [9], the target similarity value of a pair instance (l_1, l_2, f) is defined as $s = \frac{f}{S(l_1)+S(l_2)}$ and normalized later. In ranking setting [10], for each query lecture video q, the target value is defined as pairwise preference according to co-viewed frequency, namely, in training set for each video p paired with q, the larger co-viewed frequency (p, q) has, the higher ranking it stands. Table 3 shows that our preliminary experiments where a regression and ranking formulation was adopted performed worse than classification, but further experiments on understanding this aspect are required.

Table 3. Task 1 preliminary performance with different SVM settings

SVM Mode	MAR_p
Classification	0.2517
Regression	0.1100
Ranking	0.1697

4 Task 2: Estimating lecture probabilities given a previously seen set

The task 2 of the ECML/PKDD challenge models the common use-case in recommender systems where the goal is to estimate what lecture a user is most likely to see given a set of lectures that were previously seen by him/her. The triplets provided from pooled sequences in the datasets do not imply an ordering. The task seeks recommendations of the top ten lectures given that a particular set of three lectures was viewed previously.

In theory, we could use the setup of task 1 for deriving predictions for task 2 as well. That is, we can classify the lectures not specified in the set of three (seen) lectures by forming pairs with each of the three lectures and designing a method to aggregate individual scores. However, we found this method to not work well on the test dataset (We obtained a score of 0.123 using this method which is almost three times worse than our final score). For task 2, the triplets-right and triplets-left tables in the training data capture the sets of lectures that are commonly seen together using which triplet-lecture pairs information can be derived. This data can be directly used to estimate the likelihood that a particular lecture will be seen given a set of three lectures. We describe this estimation with a simple example. Let l_i refer to a lecture i where as f_{ijkl} corresponds to the frequency of seeing lectures i, j, k and l together. Assume that the following triplet-lecture pairs information is available from the training data.

$$(l_1, l_2, l_3; l_4; f_{1234}), (l_1, l_2, l_3; l_6; f_{1236}), (l_1, l_3, l_5; l_4; f_{1354}), (l_2, l_5, l_6; l_1; f_{2561})$$

For the above triplet-left-right pairs, we can estimate the number of times the set of lecture triples $(l_1, l_2, l_3), (l_1, l_2, l_4), (l_3, l_1, l_4), (l_2, l_3, l_5) \ldots$ was seen in the training data by using the associated frequency information. The number of times pairs of lectures are seen together can also be similarly estimated based on the training data. Given a query triplet such as (l_i, l_j, l_k) and a potential candidate lecture l_p, we form the possible triplet and pair sets:

$$(l_i, l_j, l_p), (l_j, l_k, l_p), (l_i, l_k, l_p), (l_i, l_p), (l_j, l_p), (l_k, l_p)$$

and use the counts estimated based on the training data to compute a score for the potential candidate l_p w.r.t. the given set of seen lectures (l_i, l_j, l_k). Clearly, not all possible pairs and triplets are likely to be found in the training data and a smoothing strategy is required for cases where triplets and pairs information is unavailable.

Note that the task 2 scenario where potential lectures are to be scored given a set of three seen lectures (triplet) parallels the item-set mining task in market-basket analysis. Market-basket analysis involves the estimation of "interestingness" of particular items given the transaction information of previous purchases. Inspired by this similarity, we design our score to capture two basic concepts from item-set mining, viz., **support** and **confidence** [2]. The support of a set X ($supp(X)$) of items is defined as the proportion of transactions in the dataset containing the set X whereas the confidence of a rule $conf(X \Rightarrow Y)$ which is interpreted as a probability estimate of seeing the set Y given that the item set X was seen is defined as $\frac{supp(X \cup Y)}{supp(X)}$.

4.1 Algorithm Description

The pseudo-code for computing the recommendation list for a given query triplet is described in Algorithm 1. We assume that the auxiliary functions *GetTriplets* and *GetPairs* are available to us. $GetTriplets(T, l_i, l_j)$ returns the set of all lectures l_k that occur with l_i and l_j in the training data T. Similarly, $GetPairs(T, l_i)$ returns all lectures that occur with l_i in T. We start by accumulating all lectures from the training data that occur with all three pairs of lectures from the query triplet. The aggregate score for a lecture is obtained by using an aggregator function over the individual confidence values. We experimented with 'product', 'max' and 'sum' as aggregator functions and found product to perform the best among those tried. If sufficient number of recommendations (input parameter) are unavailable, we relax the overlapping criterion by first considering lectures that occur with any two pairs of lectures from the triplets and finally with any pair of lectures in the triplet. Algorithm 1 can be directly used with pairs information from the training data (by obtaining potential lectures using *GetPairs* instead of *GetTriplets*).

In general, estimation based on triplets is more reliable since it captures the co-occurrence of a potential lecture with two lectures in the query. This is also illustrated in one of our experiments. Different strategies for combining scores from *GetTriplets* and *GetPairs* and for smoothing are a subject of future study. The smoothing strategy mentioned in the pseudo-code uses popular lectures (those with high number of views) as recommendations when triplets related to query lectures are unavailable in the training data.

4.2 Observations

For computing the estimates of lecture triples and pairs for task 2, we used the data available in the tables `triplets_train_left`, `triplets_train_right`, `task2_query` and `pairs` of the training data. This information was stored in memory and looked up during calls to *GetTriplets* and *GetPairs*. For each query triplet of lectures in the test set, we use Algorithm 1 to compute the recommendation list. In case of insufficient number of desired recommendations, we can use smoothing strategies. We explored the use of Algorithm 1 with *GetPairs* and most popular lectures as recommendations as smoothing strategies.

In general, we found that the scores computed based on triplets rather than pairs result in better recommendations. This is not surprising since a lecture that co-occurs with larger number of lectures in the query triplet would be a better candidate for recommending. We experimented with sum, max and product as aggregation functions on the individual confidence values. The performance with triplets, pairs, and other aggregation functions (using Algorithm 1) is shown in Table 4. We used the best setting, Algorithm 1 with *GetTriplets*, with product as the aggregation function for our final run.

Although the task description mentions a lack of sequence information among the lectures of a triplet, based on the description of how the dataset was created, there seems to be an implicit ordering among the lectures. The scores in Table 4 are obtained from runs that assume sequence information among the lectures of a query triplet. The last row shows a run that assumes that the lectures in a query triplet are unordered and

Algorithm 1 Computing Recommendations Using Lecture Triplets

Input: T(set of triplets and their frequencies from training data),
Query lecture triple $q =< l_1, l_2, l_3 >$,
k (number of recommendations desired)
Output: R (Recommendation list for q)
$R \leftarrow \phi$
$S_1 \leftarrow GetTriplets(T, l_1, l_2)$
$S_2 \leftarrow GetTriplets(T, l_1, l_3)$
$S_3 \leftarrow GetTriplets(T, l_2, l_3)$
$R_1 = S_1 \cap S_2 \cap S_3 \setminus \{l_1, l_2, l_3\}$
for all $r \in R_1$ **do**
$score(r) \leftarrow AggFunc(conf(\{l_1, l_2\} \Rightarrow r), conf(\{l_1, l_3\} \Rightarrow r), conf(\{l_2, l_3\} \Rightarrow r))$
end for
Sort R_1 in descending order and append to R
$R_2 \leftarrow ((S_1 \cap S_2) \cup (S_1 \cap S_3) \cup (S_2 \cap S_3)) \setminus (R \cup \{l_1, l_2, l_3\})$
for all $r \in R_2$ **do**
$f_1 = f_2 = f_3 = 1$
if $r \in S_1$ **then**
$f_1 = conf(\{l_1, l_2\} \Rightarrow r)$
end if
if $r \in S_2$ **then**
$f_2 = conf(\{l_1, l_3\} \Rightarrow r)$
end if
if $r \in S_3$ **then**
$f_3 = conf(\{l_2, l_3\} \Rightarrow r)$
end if
$score(r) \leftarrow AggFunc(f_1, f_2, f_3)$
end for
Sort R_2 in descending order and append to R
$R_3 \leftarrow (S_1 \cup S_2 \cup S_3) \setminus (R \cup \{l_1, l_2, l_3\})$
for all $r \in R_3$ **do**
if $r \in S_1$ **then**
$score(r) = conf(\{l_1, l_2\} \Rightarrow r)$
else if $r \in S_2$ **then**
$score(r) = conf(\{l_1, l_3\} \Rightarrow r)$
else if $r \in S_3$ **then**
$score(r) = conf(\{l_2, l_3\} \Rightarrow r)$
end if
end for
Sort R_3 in descending order and append to R
if $|R| < k$ **then**
append to R lectures with top recommendation using pairs until $|R| = k$
end if
return R

combines all possible orderings into the frequency information. This score being worse than the other runs hints at the possibility of ordering information among the lectures of a triplet in the given dataset.

Table 4. Task 2 Performance with different Algorithm Settings

Setting	MAR_p
GetPairs	0.1407
GetTriplets (product)	0.4843
GetTriplets (sum)	0.4780
GetTriplets (max)	0.4664
GetTriplets (unordered, product)	0.3107

5 Conclusions and Future Work

Using the techniques described in Sections 3 and 4, we obtained the MAR_p score of 0.2456 for task 1 and 0.4843 for task 2. The top-performing system at the ECML/PKDD Discovery challenge obtained the scores 0.35857 and 0.62415 on task 1 and task 2 respectively. Our system was ranked 8th out of 62 participating teams for task 1 and 4th out of 22 participating teams for task 2.

Since the "correct" predictions on the test data are now available, our current focus is on improving the performance of our techniques after an error analysis. We need further study to understand the performance difference between SVM classification and regression or ranking formulation. Further, in our experiments, we fit a single model over all lectures in the training data. It is possible that the lectures can be somehow clustered so that a different model is learnt for each cluster. For task 2, our scoring function uses simple estimates of confidence based on triplets seen in the training data. For queries where the required information is missing, back-up options based on content of the lectures in the query (such as our model in task 1) can be used. Other smoothing strategies and combination methods also need to be carefully studied.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE TKDE (2005)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB (1994)
3. Antulov-Fantulin, N., Bošnjak, M., Šmuc, T., Jermol, M., Žnidaršič, M., Grčar, M., Keše, P., Lavrač, N.: Ecml/pkdd 2011 - discovery challenge: "videolectures.net recommender system challenge. http://tunedit.org/challenge/VLNetChallenge (2011)
4. Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., Aly, M.: Video suggestion and discovery for youtube: taking random walks through the view graph. In: WWW (2008)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. (2003)

6. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. (1998)
7. Debnath, S., Ganguly, N., Mitra, P.: Feature weighting in content based recommendation system using social network analysis. In: WWW (2008)
8. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning Attribute-to-Feature mappings for Cold-Start recommendations. In: ICDM (2010)
9. Joachims, T.: Making large-scale support vector machine learning practical, chap. Support Vector Learning, pp. 169–184. MIT Press (1999)
10. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 133–142. KDD '02, ACM, New York, NY, USA (2002)
11. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing (2003)
12. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval (2008)
13. Pazzani, M.J., Billsus, D.: Content-Based recommendation systems. In: The Adaptive Web (2007)
14. Strohman, T., Croft, W.B., Jensen, D.: Recommending citations for academic papers. In: SIGIR. SIGIR '07 (2007)