Proceedings of the ECML PKDD 2011 Workshop on

DISCOVERY CHALLENGE

Edited by

Tomislav Šmuc
Nino Antulov-Fantulin
Mikołaj Morzy

European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
Athens, Greece, September 5-9, 2011

Proceedings of the ECML PKDD 2011 Workshop on Discovery Challenge

Edited by

Tomislav Šmuc
Nino Antulov-Fantulin
Mikołaj Morzy

Published by: Rudjer Bošković Institute, Zagreb, Croatia
ISBN-13: 978-953-6690-89-3
EAN: 9789536690893

Workshop and Challenge Organization

Workshop Chairs

Vassilis Plachouras, University of Glaskow, UK
Alexandros Kalousis, University of Geneva, Switzerland

Challenge Organizing Committee

Tomislav Šmuc, Rudjer Bošković Institute, Croatia
Nino Antulov-Fantulin, Rudjer Bošković Institute, Croatia
Matko Bošnjak, Rudjer Bošković Institute, Croatia
Martin Žnidaršič, Jožef Stefan Institute, Slovenia
Miha Grčar, Jožef Stefan Institute, Slovenia
Mitja Jermol, Jožef Stefan Institute, Slovenia
Nada Lavrač, Jožef Stefan Institute, Slovenia
Peter Keše, Viidea Ltd, Slovenia
Mikołaj Morzy, Poznań University of Technology, Poland

Programm Committee

Sarabjot S. Anand, University of Warwick, UK
Nino Antulov-Fantulin, Rudjer Bošković Institute, Croatia
Shlomo Berkovsky, CSIRO, Australia
Matko Bošnjak, Rudjer Bošković Institute, Croatia
Amancio Bouza, University of Zurich, Switzerland
Dragan Gamberger, Rudjer Bošković Institute, Croatia
Zeno Gantner, University of Hildesheim, Germany
Miha Grčar, Jožef Stefan Institute, Slovenia
Nada Lavrač, Jožef Stefan Institute, Slovenia
Agnieszka Ławrynowicz, Poznań University of Technology, Poland
Mikołaj Morzy, Poznań University of Technology, Poland
Markus Schedl, Johannes Kepler University, Austria
Yue Shi, Delft University of Technology, The Netherlands
Steffen Rendle, University of Konstanz, Germany
Tomislav Šmuc, Rudjer Bošković Institute, Croatia
Martin Žnidaršič, Jožef Stefan Institute, Slovenia

Table of Contents

Preface

The 2011 ECML-PKDD Discovery Challenge deals with the learning problems from the domain of recommender systems. Datasets and problems designed by the organizers of this Challenge, originate from the VideoLectures.Net site, a free and open access multimedia repository of video lectures, mainly of research and educational character. The lectures are given by distinguished scholars and scientists at the most important and prominent events like conferences, summer schools, workshops and science promotional events from many fields of science. The Challenge was organized with multiple aims in mind: to improve the current websites recommender system, discover new algorithms or computational workflows and provide new dataset for the research community. It encompassed two tasks: first one related to new-user/new-item recommendation problem, and the second task in which "normal mode", click-stream based recommendation is simulated. Dataset for the challenge is somewhat specific as it does not include any explicit nor implicit user preference data. Instead, implicit profiles embodied in viewing sequences have been transformed into a graph of lecture co-viewing frequencies and pooled viewing sequences. The data also includes content related information: topic taxonomy, lecture titles, descriptions and slide titles, authors' data, institutions, lecture events and timestamps. The dataset (including the leaderboard and the test set) will remain publicly available for experimentation after the end of the challenge.

Over 300 teams registered for the challenge, resulting in more than 2000 submitted results for the evaluation from 62/22 active teams for task 1 and task 2, respectively. The teams approached the tasks with diverse algorithms and in several cases novel feature construction approaches. The following are the winners of the challenge:

Task 1 Cold-start problem:

- Alexander Dýakonov (1st place)
- Eleftherios Spyromitros-Xioufis, Emmanouela Stachtiari, Grigorios Tsoumakas, and Ioannis Vlahavas (2nd place)
- Martin Možina, Aleksander Sadikov, and Ivan Bratko (3rd place)

Task 2 Pooled sequence recommendation:

- Alexander Dýakonov (1st place)
- Javier Kreiner (2nd place)
- Vladimir Nikulin (3rd place)

The Discovery Challenge workshop at the ECML-PKDD 20011 conference in Athens is aimed for discussion of the results, approaches, VL.net dataset and lecture recommendation setting in general. We wish to express our gratitude to:

- the participants of the challenge,
- the authors of the submitted papers,
- Viidea Ltd for disclosing the data on video lectures and for the technical support

Zagreb
August 2011

Tomislav Šmuc
Nino Antulov-Fantulin
Mikołaj Morzy

This workshop was supported by the European Union Collaborative Project e-LICO (*e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science* GA 231519). The partners of e-LICO are:

- University of Geneva - Co-ordinator (Switzerland)
- Institut National de la Sant et de la Recherche Mdicale (France)
- Jošef Stefan Institute (Slovenia)
- National Hellenic Research Foundation (Greece)
- Poznań University of Technology (Poland)
- Rapid-I GmbH (Germany)
- Rudjer Bošković Institute (Croatia)
- University of Manchester (UK)
- University of Zurich (Switzerland)

ECML-PKDD 2011 Discovery Challenge Overview

Nino Antulov-Fantulin[1], Matko Bošnjak[1], Martin Žnidaršič[2], Miha Grčar[2], Mikołaj Morzy[3], and Tomislav Šmuc[1]

[1] Rudjer Boškovic Institute, Zagreb, Croatia
[2] Jožef Stefan Institute, Ljubljana, Slovenia
[3] Poznań University of Technology, Poznań, Poland

Abstract. This year's Discovery Challenge was dedicated to solving of the video lecture recommendation problems, based on the data collected at VideoLectures.Net site. Challenge had two tasks: task 1 in which new-user/new-item recommendation problem was simulated, and the task 2 which was a simulation of the clickstream-based recommendation. In this overview we present challenge datasets, tasks, evaluation measure and we analyze solutions and results.

1 General description of the challenge

VideoLectures.Net (VL.Net)[4] is a free and open access multimedia repository of video lectures, mainly of research and educational character. The lectures are given by distinguished scholars and scientists at the most important and prominent events like conferences, summer schools, workshops and science promotional events from many fields of science. The website is aimed at promoting science, exchanging ideas and fostering knowledge sharing by providing high quality didactic contents not only to the scientific community but also to the general public. All lectures, accompanying documents, information and links are systematically selected and classified through the editorial process taking into account also users' comments.

This challenge was organized through the support of the EU project e-LICO[5]. The aims of the challenge are multifold: from research in recommender systems, improvement of the current recommender system of the VL.Net site, to provision of the problem and datasets to the research community. The challenge consisted of two main tasks. Due to the nature of the problem, each of the tasks has its own merit: task 1 simulates new-user and new-item recommendation cold-start mode, while task 2 simulates clickstream (implicit preference) based recommendation. Due to the privacy preserving constraints, data from VL.Net website includes neither explicit nor implicit user profiles. Instead, implicit profiles embodied in viewing sequences (clickstreams) have been transformed, so that no individual viewing sequence information can be revealed or reconstructed. This transformed, viewing related data includes: i) lecture co-viewing frequencies ii) pooled viewing sequences (whose construction will be described later) and is accompanied with rich lecture description related information available: lecture category taxonomy, lecture names, descriptions and slide titles (where available), authors, institutions, lecture events and timestamps. Unlike most of the other publicly available recommendation problems datasets, this dataset contains original content, names and taxonomy. The dataset of the challenge[6] (including

[4] `http://videolectures.net`
[5] `http://www.e-lico.eu`
[6] `http://lis.irb.hr/challenge/`

the leaderboard and the test set), together with task and evaluation descriptions is publicly available for the non-commercial research purposes [28].

We have ensured prize-sponsoring (5500€) from the European Commission through the e-LICO EU project, 2009-2012 whose primary goal is to build a virtual laboratory for interdisciplinary collaborative research in data mining and data-intensive sciences.

The prizes, for each of the tracks are:

- 1500€ for the first place
- 700€ for the second place
- 300€ for the third place

The prizes, for the Workflow contest are:

- 500€ for the best workflow
- Free admission to RapidMiner Community Meeting and Conference 2012 for the best RapidMiner workflow (sponsor: Rapid-I)

The challenge has been hosted on TunedIt[7].

2 Background

Recommender systems have become an important research area since the first appearance of the information overload for the typical user on the internet. Personalized recommender systems take user profiles into account when the prediction for particular user and item is generated. The prediction techniques for the recommender systems [1–3] can be divided into three main categories: content-based, collaborative-based and hybrid-based prediction techniques.

Content-based techniques [4, 5] are based on interactions between a particular user and all the items in the system. Content-based recommender systems use information about items and the user's past activities on items in order to recommend similar items.

Collaborative filtering techniques [6–8] analyze interactions between all users and all items through users' ratings, clicks, comments, tags, etc. Collaborative filtering recommender systems do not use any specific knowledge about the items except their unique identifiers. These prediction techniques are domain-independent and can provide serendipity recommendations for users. However, collaborative filtering needs sufficient amount of collaborative data in order to recommend for the new user or the new item (the cold-start problem) [9, 10].

Hybrid prediction techniques [11–13] merge collaborative-based and content-based techniques and are more resistant to cold start problems. This challenge was designed to tackle the problems of cold start and hybridization of content and collaborative data in realistic setting of the VL.Net website. In comparison to recommender challenges of recent years (Netflix challenge, KDDCup challenge 2008, KDDCup challenge 2011) this challenge relies on indirect collaborative data, and is more focussed on utilization of content and descriptions of items.

[7] `http://tunedit.org`

3 Description of the challenge dataset

The data snapshot which is the basis for the VideoLectures.Net dataset was taken in August 2010. At that time, the database contained 8 105 video lectures. 5 286 lectures were manually categorized into taxonomy of roughly 350 scientific topics such as Arts, Computer Science, and Mathematics.

VideoLectures.Net dataset includes:

1. **Data about lectures**: every lecture has a title, type (e.g. lecture, keynote, tutorial, press conference, etc.) language identifier (e.g. *en*, *sl*, *fr*, etc.), number of views, publication date, event identifier, and a set of authors. Many lectures come with a short textual description and/or with slide titles from the respective presentations. Specifically, 5 724 lectures are enriched with this additional unstructured data. The training part of data contains also lecture-pairs coviewing frequencies (CVS - common view score), and pooled sequences related collaborative data, which is not available for the set of test lectures. Test set contains lectures with publication date after July 01, 2009, which are used for task 1 scoring. Neither CVS nor pooled viewing sequences containing these lectures are available in the training data.
2. **Data about authors**: each author has a name, e-mail address, homepage address, gender, affiliation, and the respective list of lectures. The dataset contains 8 092 authors. The data about the authors is represented by authors' names, VL.Net url, e-mail, homepage, gender, affiliation, and pairwise relations to the lectures delivered by the author at VL.Net
3. **Data about events**: a set of lectures can be associated with an event (e.g. a specific conference). In a similar fashion, events can be further grouped into meta-events. An event is described in a similar way as a lecture: it has a title, type (e.g. project, event, course), language identifier, publication date, and a meta-event identifier. The VideoLectures.Net dataset contains data about 519 events and meta-events (245 events are manually categorized, 437 events are enriched with textual descriptions).
4. **Data about the categories**: The data about the categories is represented in the shape of the scientific taxonomy used on VL.Net. The taxonomy is described in a pairwise form, using parent and child relations.
5. **View statistics**: The VideoLectures.Net software observes the users accessing the content. Each browser, identified by a cookie, is associated with the sequence of lectures that were viewed in the identified browser. Temporal information, view durations, and/or user demographics are not available. The dataset contains anonymized data of 329 481 distinct cookie-identified browsers. The data about view statistics is given in the form of frequencies: (i) for a pair of lectures viewed together (not necessarily consecutively) with at least two distinct cookie-identified browsers; (ii) for pooled viewing sequences - triplets of lectures viewed together prior to a given sequence of ten lectures. This is a special construct based on aggregation of click-streams, which is used for training and scoring in task 2.

3.1 Creating pooled viewing sequences

In order to comply with privacy-preserving constraints, lecture viewing sequences for the task 2 have been transformed into what we named pooled sequences. Pooled

viewing sequence is given by the set of three lectures on the left side (triplet) and a ranked list of at most ten lectures on the right side. The set of three lectures does not imply an ordering, it is merely a set that comes *upstream* of lectures given on the right of a pooled viewing sequence. Ranked list on the right side of some pooled viewing sequence is constructed from all the clickstreams with the particular triplet on the left side. The transformation process for the construction of pooled viewing sequences is given below.

Consider a sequence of viewed lectures:

$$id_1 \to id_7 \to id_2 \to id_1 \to id_4 \to id_5 \to id_6 \to id_3$$

We first filter out duplicates (here - id_1):

$$id_1 \to id_7 \to id_2 \to id_4 \to id_5 \to id_6 \to id_3$$

Then, we determine all possible unordered triplets in the sequence. For each triplet, cut the sequence after the right-most lecture from the triplet.

In the above example, if $\{id_1, id_4, id_5\}$ is the triplet, the sequence is cut right after id_5. Finally, increase triplet-specific counts for all the lectures after the cut. In the above example, given the triplet $\{id_1, id_4, id_5\}$, the triplet-specific counts for id_6 and id_3 are increased:

$$\{id_1, id_4, id_5\} \to id_6 : 1, id_3 : 1$$

Suppose there is another click-stream sequence, that amongst others, contains unordered triplet id_1, id_4, id_5 and that id_6, id_3, and id_7 are lectures appearing after the cut. Then the counts for the $\{id_1, id_4, id_5\}$ are increased as follows:

$$\{id_1, id_4, id_5\} \to id_6 : 2, id_3 : 2, id_7 : 1$$

3.2 Creating lecture co-viewing frequencies

Consider two sequences of viewed lectures:

$$id_1 \to id_7 \to id_2 \to id_1,$$

$$id_2 \to id_3 \to id_7.$$

We first filter out duplicates in sequences:

$$id_1 \to id_7 \to id_2,$$

$$id_2 \to id_3 \to id_7.$$

Then, we determine lecture co-viewing frequencies (CVS):

$$CVS(id_1, id_2) = 1, CVS(id_1, id_7) = 1,$$

$$CVS(id_2, id_7) = 2, CVS(id_2, id_3) = 1,$$

$$CVS(id_3, id_7) = 1.$$

Table 1: Train-test data statistics

Moment t_2	05.08.2010.
Moment t_1	01.07.2009.
Total number of lectures in the train set	6983
Total number of lectures in the test set	1122
Number of common-view pairs in the train set	363 880
Number of common-view pairs in the test set	18 450

3.3 A train-test split logic

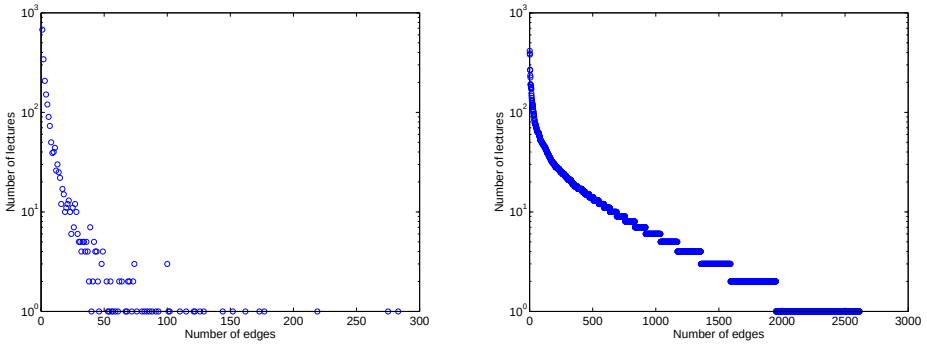
Basic statistics of lectures in the training and test sets are given in Table 1. Common view score matrix CVS is a lecture co-viewing frequency matrix collected at the site at some moment t_2 and represents lecture viewing adjacency matrix of lecture-lecture graph G at the moment t_2. G is undirected weighted graph of all lectures. Each lecture in this graph has associated temporal information - date of publishing at the VideoLectures.Net site. We partition G using the publishing date by some threshold t_1, into two disjoint graphs G_1 and G_2: each lecture in G_1 has publishing date before the date threshold while each lecture in G_2 has publishing date after the date threshold t_1. We define pair common viewing time as a period that two lectures spend together in the system. All lecture pairs $(x_i, x_j) : x_i \in G_1, x_j \in G_1$ have pair common time strictly greater than $(t_2 - t_1)$ value and all lecture pairs $(x_i, x_j) : x_i \in G_1, x_j \in G_2$ have pair common time strictly less than $(t_2 - t_1)$ value.

In oder to make proper the training and test set split based on G_1 and G_2, we had to ensure similar distribution of pair common times in both training and test sets. We have divided nodes from subgraph G_2 in randomized fashion (with some constraints) into two approximately equal sets (G_{21}, G_{22}) and we have appended G_{21} to the training set. Now, the subset of lecture pairs $(x_i, x_j) : x_i \in G_1, x_j \in G_{21}$ from the training set has similar distribution of pair common times that overlaps with times $(x_i, x_j) : x_i \in G_1, x_j \in G_{22}$ from the test set. Figure 1 gives the distribution of edges related to the graphs G_1, G_{22}.

Finaly, the train-test split logic was implemented through the series of steps:

1. Split the lectures by publication date into two subsets: *old* (publication date $<$ July 01, 2009) and *new* (publication date $\geq$ July 01, 2009). Put the *old* lectures into the training set;
2. Move all *new* lectures with parent id occuring in the *old* lecture subset to the training set;
3. Split the rest of the *new* lectures randomly into two disjoint sets of similar cardinality, taking care of their parent ids:
   (a) lectures with the same parent id can be only in one of the sets;
   (b) lectures without parent id are just randomly divided between two sets.
4. Finally, add one of the disjoint sets to the training set; the other disjoint set represents the test set.

At the end of the process, we get the training set consisting of all the lectures with publishing date prior to July 01, 2009, together with approximately half of the lectures after the aforementioned date, and the test set consisting of the rest of the lectures published after the aforementioned date.



(a) (b)

Fig. 1: Distribution of edges of lecture-lecture graph with adjacency matrix of co-ocurrences between lectures. (a) for all lecture pairs $(x_i, x_j) : x_i \in G_1, x_j \in G_{22}$ (b) for all lectures pairs $(x_i, x_j) : x_i \in \{G_1 \cup G_{21}\}, x_j \in \{G_1 \cup G_{21}\}$

4 Challenge task definiton

Due to the nature of the problem, each of the tasks has its own merit: task 1 simulates new-user and new-item recommendation (cold start mode); task 2 simulates clickstream-based (implicit preference) recommendation.

4.1 The cold start task

The first task of the challenge is related to solving the so called *cold start problem*, commonly associated with pure collaborative filtering (CF) recommenders. Generally, *cold start* recommending quality should be measured through user satisfaction surveys and analysis. For the challenge, one needs a quantitative measure and a simulated cold start situation. In order to be able to score solutions, new video lectures are those that entered the site more recently, but for which there is already some viewing information available.

In this task, we assume that the user has seen one of the lectures which are characterized by the earlier times of entering the site (*old* lectures). As a solution for this task a ranked list of lectures from the *new* lectures set, is to be recommended after viewing some of the *old* lectures. The length of the recommended list is fixed at 30 lectures. Overall score for the submission/solution is based on the mean average R-precision score (*MARp*) (explained in Section 5).

Solution for the task 1 is based on ranking of lectures according to withheld lecture co-viewing frequencies in descending order. Suppose, the co-viewing frequencies (CVS) for some old lecture id_1 to new lectures $\{id_2, id_3, id_4, id_5\}$ are:

$$CVS(id_1, id_2) = 12, CVS(id_1, id_3) = 2,$$

$$CVS(id_1, id_4) = 43, CVS(id_1, id_5) = 3,$$

then we construct solution ranked list for old-lecture id_1:

$$id_1 : id_4, id_2, id_5, id_3.$$

4.2 Pooled lecture viewing sequences task

In task 2 contestants are asked to recommend a ranked list of ten lectures that should be recommended after viewing a set of three lectures. In contrast to the task 1, this is the situation close to typical recommendation scenario (submission and evaluation for the task 2). Solution for the task 2 is based on ranking of lectures according to frequencies in withheld pooled lecture viewing sequences in descending order. Test lectures from the task 1 are in this case not included into training pooled sequences, but can be a part of the ranked solution list for the task 2.

Suppose, there is a pooled lecture viewing sequences:

$$\{id_1, id_4, id_5\} \rightarrow id_6 : 5, id_3 : 4, id_7 : 2, id_2 : 1,$$

then we construct solution ranked list for triplet $\{id_1, id_4, id_5\}$:

$$\{id_1, id_4, id_5\} \rightarrow id_6, id_3, id_7, id_2.$$

5 Challenge evaluation function

Taking into account relative scarcity of items available for learning, recommending and evaluation (esp. in case of cold start task), we have defined an R-precision variants of standard evaluation measures in information retrieval $p@k$ and MAP. The overall score of the submission is mean value over all queries R (recommended lists r) given in the test sets:

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

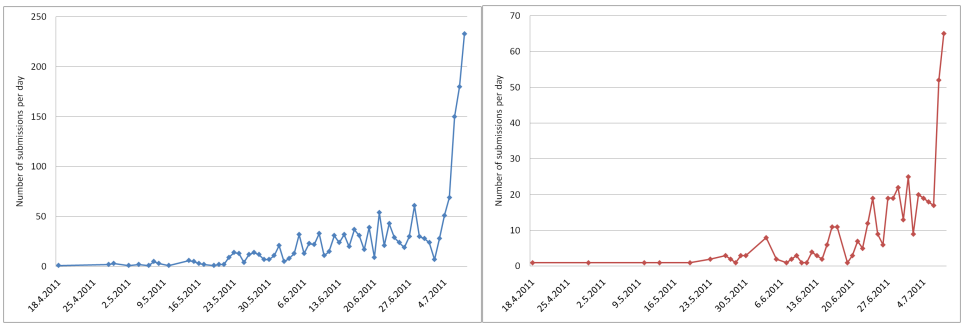
Average R-precision score - $AvgRp(r)$ for a single recommended ranked list r is defined as:

$$AvgRp(r) = \sum_{z \in Z} \frac{Rp@z(r)}{|Z|}$$

where $Rp@z(r)$ is R-precision at some cut-off length $z \in Z$. $Rp@z(r)$ is defined as the ratio of number of retrieved relevant items and relevant items at the particular cut-off z of the list:

$$Rp@z(r) = \frac{|relevant \cap retrived|_z}{|relevant|_z} = \frac{|relevant \cap retrived|_z}{min(m, z)}$$

Number of relevant items at cut-off length z is defined as $min(m, z)$, where m is the total number of relevant items. When $m \leq z$, number of relevant items at z is m, while for other situations it is limited to top z relevant items from the (real) solution ranked list s. A special situation happens when there are more equally relevant items at the same rank (ties) at the cut-off length of the s list. In that case, any of these items are treated as relevant (true positive) in calculating $Rp@z(r)$. For the task 1, cut-off lengths z for the calculation of $MARp$ are $z \in \{5, 10, 15, 20, 25, 30\}$. For the task 2, cut-off lengths z for the calculation of $MARp$ are $z \in \{5, 10\}$.



(a) (b)

Fig. 2: Number of submissions per days for (a) task 1 (b) task 2

Why average R-precision?

We have introduced R-precision because it is more apt to our situation: it adjusts to the size of the set of relevant documents. Typically, in information retrieval tasks one has to filter and rank from a large pool of both relevant and irrelevant items. This is not the case with the simulated cold start situation of this challenge. As an example, if there were only 4 items (lectures) in the whole collection relevant to the particular query, a perfect recommender system would score 1, measured by $Rp@10$, whereas its $p@10$ would be only 0.4. Using this measure for our application makes more sense, as the number of relevant items can vary from 1 to above 30, and in such situations $Rp@z$ expresses the quality of retrieval more fairly at some predefined retrieval (cut-off) length, than $p@z$. The reason why we use $AvgRp(r)$ over set of different $Rp@z$, is that through the averaging we can also take into account ranking and at the same time improve the ability to differentiate between similar solutions (recommenders).

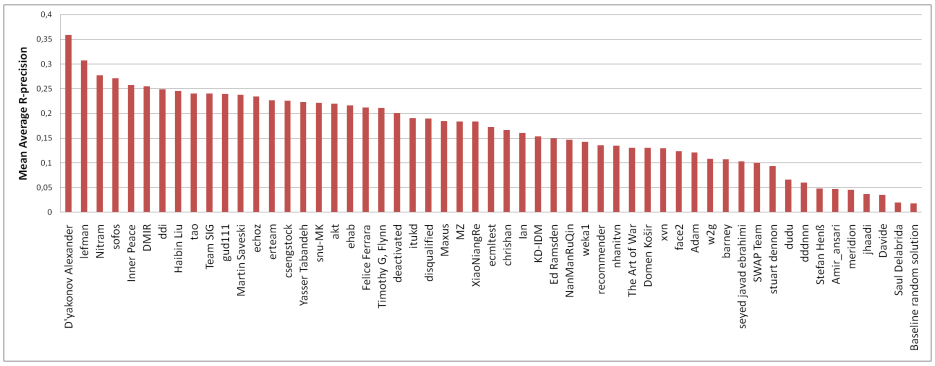
We have also considered MAP (mean average precision) measure, which is the closest to the proposed measure. However, MAP does not take into account absolute ranking positions of recommended items since permutations of relevant or true positive items in recommended list do not affect MAP score.

Normalized discounted cumulative gain ($NDCG$) [16, 17] takes into account that relevant documents are more useful when apperaing earliear in a recommendation list. It is the most common measure used for ranking the results of the search list in information retrieval. This measure has also been used in other challenges where the main task was to learn ranking [14, 15].

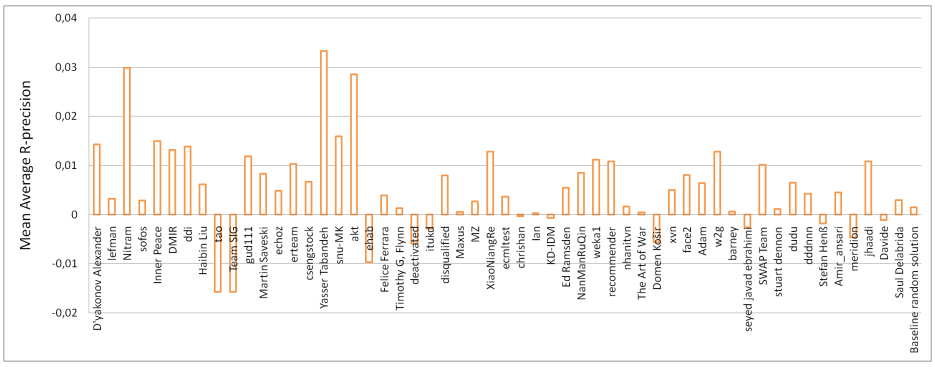
If ranking order is not to be so strict for the top-n item recommendations [18], the "granularity" of ranking can be relaxed. This is the main reason why we are using $MARp$ measure instead of the $NDCG$. Proposed measure $MARp$ takes into account absolute ranking positions with granularity of five items. This granularity was chosen after studying the ranking-recall influence on recommender system evaluation.

6 Challenge submissions results

ECML-PKDD 2011 Discovery Challenge started on 18th of April and ended on 8th of July 2011. The competition attracted significant number of participants: 303 teams



(a)



(b)

Fig. 3: (a) The *MARp* scores of final submissions for task 1. (b) Difference between *MARp* preliminary score and the *MARp* final score for task 1.

with 346 members, with 62/22 active teams per task. More than 2000 submissions were sent and best approaches outperformed baseline solution several times.

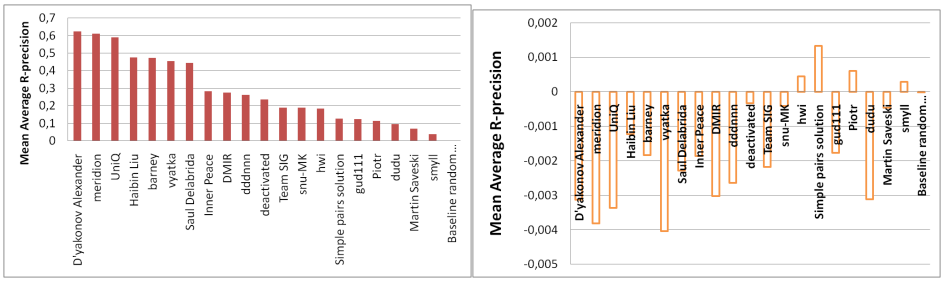
Winners of the challenge for task 1 are:

1. D'yakonov Alexander, Faculty of Computational Mathematics and Cybernetics, Moscow State University (Username: "D'yakonov Alexander")
2. Eleftherios Spyromitros-Xioufis and Emmannouela Stachtiari, Department of Informatics, Aristotle University of Thessaloniki (Username: "lefman")
3. Martin Možina, Faculty of Computer and Information Science, University of Ljubljana, Slovenia (Username: "Nitram")

Winners of the challenge for task 2 are:

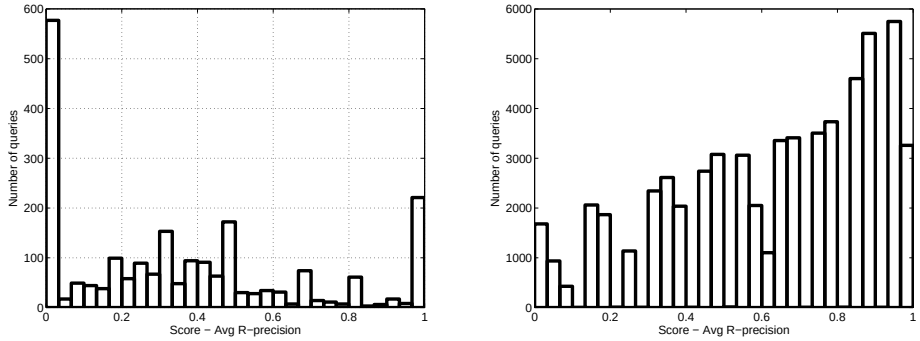
1. D'yakonov Alexander, Faculty of Computational Mathematics and Cybernetics, Moscow State University (Username: "D'yakonov Alexander")
2. Javier Kreiner, University of Trento, Italy (Username: "meridion")
3. Vladimir Nikulin, Department of Mathematics, The University of Queensland, Australia (Username: "UniQ")

The final scores, for the teams that scored better than the random recommender, are presented in the Figures 3 and 4, for each of the tasks respectivelly. The scores are



(a) (b)

Fig. 4: (a) the *MARp* scores of final submissions for task 2. (b) Difference between *MARp* preliminary score and the *MARp* final score for task 2.



(a) (b)

Fig. 5: Number of queries in the solution vs. *AvgRp* scores for the winning entries (a) task 1, (b) task 2

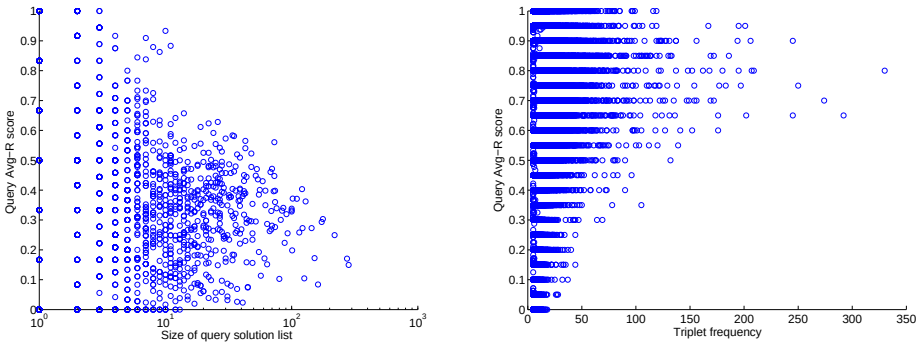
.

accompanied with the graphs of differences between preliminary *MARp* score on the leaderboard set and the final *MARp* score on the test set. For the task 1, from Figure 3, we can conclude that majority of the teams had positive difference scores, which may suggest overtraining. To the contrast, the majority of the teams had negative difference scores in tasks 2 (see Figure 4).

The distributions of the average R-precision over queries for the winning entry on each of the tasks are presented in Figure 5. Difference in distributions between the tasks reflects also the difference in the approaches used: while for the first task main features for solving the problem are constructed from lecture content and meta-data similarity, for the second task only co-viewing information is utilized. We have also noted that these distributions are qualitatively very similar between first three positioned entries on each of the tasks, reflecting general similarity in approaches of different teams.

Dependence of query average R-precision score on the size of the solution list for the task 1 is presented in Figure 6 (graph on the left). On average, query score just slightly diminishes with the increase of the solution list. To the contrast, dependence

of the query average R-precision score to the triplet frequency, for the task 2 (graph on the right in Figure 6) shows that on average the quality of result for the query is proportional to the triplet frequency.



(a) (b)

Fig. 6: (a) Dependence of individual query *AvgRp* score on the size of the query solution list (number of relevant lectures in the solution list); (b) Dependence of individual query *AvgRp* score on the triplet frequency in the test set (reflecting popularity and co-viewing connectedness of the lectures in the triplet)

.

6.1 Methods used and discussion of approaches

The teams approached task 1 using quite different learning techniques with the primary effort focussed on feature engineering and optimization. Almost all of the participants have utilized all the lecture content related data (lecture taxonomy, event tree, types of lectures, descriptions, etc.), differing however slightly in definitions of the similarity of any two lectures. Important with respect to the overall score was the process of filling the missing values for the lectures that lack some of the content related data. Winning solutions used more sophisticated approach of filling lecture content and meta-data features' missing values using lecture co-viewing information (weighted CVS feature vector expansion [19], query expansion[20]) - thus utilizing collaborative information to "enrich" content-based features.

Table 2 gives a summary of the feature engineering approaches and learning methods used in solving challenge tasks.

7 Conclusion

In the last couple of years, a number of challenges was organized in the field of recommendation problems. Most of them were focussed on prediction problems related to large scale explicit or implicit user preference matrices, in some cases combined with (mostly obfuscated) user, item and/or context related information. ECML-PKDD

2011 Discovery Challenge differed from this mainstream through two aspects: (i) instead of user preferences, only item to item preference information is available in the shape of the co-viewing frequencies graph; (ii) a rich and explicit description of lectures is available in the form of structured and unstructured text. On both tasks participants have obtained significantly higher $MARp$ values than set by the baseline solutions.

The analysis of the results shows that the most important part of a successful solution was careful feature engineering. Definition of the similarity scoring function capable of capturing content, context and temporal information turned out to be crucial for the success in the cold start (task 1) competition. Task 2, pooled sequence completion problem, was easier to solve and both approaches and results of the participants were mutualy much more similar. Rather unexpectedly, content related information was not used in ranking lectures to be viewed in succession to test set triplets. Most of the participants have also reported about the complexity/scaling of their solutions.

Table 2: Approaches in solving challenge tasks

Solution (Track/pos)	**Feature engineering approach**	**Model learning approach**
1/1	combined similarity vector; uses feature expansion with CVS graph weighting; uses temporal transform for final similarity indices (LENKOR methodology)	optimization of weights of the linear model by the coodinate descent
1/2	textual based features synthesis; *tf-idf* based; use "query" expansion for missing terms; use temporal information (co-existence similarity)	cosine-similarity based *k-nn*, fitting to the optimal k; train the model utilizing "temporal" split in the training set
1/3	meta-data into categorical features (do not use in the model: lecture viewing, lecture description, slides' content); use co-viewing information to expand the content based rank prediction	content based linear regression for learning rank; utlize stochastic gradient descent to learn parameters of the linear model; learn hyperparameters from leaderboard submissions
2/1	two level normalization of pooled sequence vectors from the training set constructed from "pairs" and "singles" of triplets from the training set (LENKOR methodology) (do not use in the model: content and other lecture meta-data)	optimization of weights of the linear model by the coodinate descent
2/2	forming conditional probabilities for RHS lectures of triplets based on triplet lecture training co-viewing; no use of content based data	probabilistic model; entropy "like" scoring formulation; greedy grid based search fitting of coefficients in the scoring function
2/3	use singles', pairs' and couples' frequencies and define score updating	resampling based combination of the individual scoring functions; propose gradient based matrix factorization and recommendation model
Other track 1 approaches	tf-idf for different content/meta-data; Jaccard similarity based on LDA for topics	SVM for co-viewing prediction (binary classification, regression and ranking); random walk model using CVS graph (Katz and RF based), ordinary linear regression
Other track 2 approaches	using pairs and single lecture co-viewing data to construct test triplet RHS solution frequency vectors; hybridization of content similarity and co-viewing data	frequent item-set formulation - support/confidence based scoring and ranking

Our opinion is that the results of the challenge could be quite useful for constructing a new recommendation system for the VideoLectures.Net. In particular, there are several approaches that could significantly improve recommendation quality of new lectures at the site, with modest consumption of additional computational resources. Using lecture co-viewing frequency information instead of original preferences information in the form of click-streams should be studied in more detail, in order to understand the implications of this transformation on the personalized recommendation quality from the user's perspective.

Acknowledgements

The Discovery Challenge 2011 has been supported by the EU collaborative project e-LICO (GA 231519). The organizers of the Challenge are grateful to the Center for Knowledge Transfer in Information Technologies of the Jožef Stefan Institute and Viidea Ltd for the data of the VideoLectures.Net site, and TunedIT for the professional support in conducting the competition. Finally, we want to thank all the active participants of the challenge for their effort in the challenge and willingness to share their solutions and experience through the contributions in this workshop.

References

1. S.Rendle, K.Tso-Sutter, W.Huijsen, C.Freudenthaler, Z.Gantner, C.Wartena, R.Brussee and M. Wibbels: Report on State of the Art Recommender Algorithms (Update). MyMedia public deliverable D4.1.2., (2011).
2. G.Adomavicius and A.Tuzhilin Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on knowledge and data engineering, **17(6)** (2005).
3. M.Montaner, B.Lopez and J.L. de la Rosa: A Taxonomy of Recommender Agents on the Internet. Artificial Intelligence Review, **19**, (2003), 285-330.
4. G.Salton: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley, (1989).
5. R.Baeza-Yates and B. Ribeiro-Neto: Modern Information Retrieval. Addison Wesley, (1999).
6. W.Hill, L. Stead, M. Rosenstein and G.Furnas: Recommending and Evaluating Choices in Virtual Community of Use. Proc. Conf. Human Factors in Computing Systems, (1995).
7. P.Resnick, N.Iakovou, M.Sushak, P.Bergstrom and J.Riedl: GroupLens: An Open Architecture for Collaborative Filtering of Netnews Proc. Computer Supported Cooperative Work Conf., (1994).
8. U.Shardanand and P.Maes: Social Information Filtering: Algorithms for Automating 'word of Mouth' Proc. Conf. Human Factors in Computing Systems, (1995).
9. C.Boutilier , R.S.Zemel and B.Marlin: Active Collaborative Filtering. In Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence, (2003).
10. A.Schein, A.Popescul, L.Ungar, and D.Pennock: Generative models for coldstart recommendations. In Proceedings of the 2001 SIGIR Workshop on Recommender Systems, (2001).
11. M.Balabanovic and Y.Shoham: Fab: Content-based, collaborative recommendation. Communications of the ACM, **40(3)**, (1997).
12. J. Basilico and T. Hofmann: Unifying collaborative and content-based filtering. In Proceedings of the Twenty-First International Conference on Machine Learning, pages 65-72, New York, NY, USA, ACM Press, (2004).
13. R.Burke: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, **12(4)**, pp 331-370, (2002).

14. O. Chapelle, Y. Chang: Yahoo! Learning to Rank Challenge Overview, JMLR: Workshop and Conference Proceedings 14, pp 1-24, (2011).
15. Internet Mathematics 2009 contest: Limited Liability Company, http://imat2009.yandex.ru/academic/mathematic/2009/en/.
16. K. Jarvelin, J. Kekalainen: Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems 20(4), pp 422-446 (2002).
17. B. Croft, D. Metzler, and T. Strohman: Search Engines: Information Retrieval in Practice. Addison Wesley, (2009).
18. A. Turpin, W. Hersh: Why batch and user evaluations do not give the same results, In Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, pp 17-24, (2001).
19. A. Dýakonov: Two Recommendation Algorithms Based on Deformed Linear Combinations. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 21-27 (2011).
20. E. Spyromitros-Xioufis, E. Stachtiari, G. Tsoumakas, and I. Vlahavas: A Hybrid Approach for Cold-start Recommendations of Videolectures. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 29-39, (2011).
21. M. Možina, A. Sadikov, and I. Bratko: Recommending VideoLectures with Linear Regression. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 41-49, (2011).
22. J. A. Kreiner and E. Abraham: Recommender system based on purely probabilistic model from pooled sequence statistics. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 51-57, (2011).
23. V. Nikulin: OpenStudy: Recommendations of the Following Ten Lectures After Viewing a Set of Three Given Lectures. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 59-69, (2011).
24. H. Liu, S. Das, D. Lee, P. Mitra, C. Lee Giles: Using Co-views Information to Learn Lecture Recommendations. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 71-82, (2011).
25. M. Chevalier, T. Dkaki, D. Dudognon, J. Mothe: IRIT at VLNetChallenge. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 83-93, (2011).
26. L. Iaquinta and G. Semeraro: Lightweight Approach to the Cold Start Problem in the Video Lecture Recommendation. In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 95-101, (2011).
27. G. Capan, O. Yilmazel: Joint Features Regression for Cold-Start Recommendation on VideoLectures.Net In Proc. of ECML-PKDD 2011 Discovery Challenge Workshop, pp 103-109, (2011).
28. N. Antulov-Fantulin, M. Bošnjak, T. Šmuc, M. Jermol, M. Žnidaršič, M. Grčar, P. Keše, N. Lavrač: ECML/PKDD 2011 - Discovery challenge: VideoLectures.Net Recommender System Challenge, `http://lis.irb.hr/challenge/`

Two Recommendation Algorithms Based on Deformed Linear Combinations*

Alexander D'yakonov

Moscow State University, Moscow, Russia,
djakonov@mail.ru,

Abstract. Data mining for recommender systems has gained a lot of interest in the recent years. "ECML/PKDD Discovery Challenge 2011" was organized to improve current recommender system of the VideoLectures.Net website. Two main tasks of the challenge simulate new-user and new-item recommendation (cold-start mode) and clickstream based recommendation (normal mode). This paper provides detailed descriptions of two simple algorithms which were very successful in the both tasks. The main idea of the algorithms is construction of a linear combination equal to a vector of estimations of lectures popularity after viewing a certain lecture (or lectures). Each addend in the combination describes similarity of lectures using the part of the data. The algorithms are improved by transforming the combination to non-linear function. Lectures with the highest estimations of popularity are recommended to users.

1 Introduction

Algorithms which have taken the first places in the competition "ECML/ PKDD Discovery Challenge 2011 (VideoLectures.Net Recommender System Challenge)" [1] are described. The competition was focused on algorithm development for making recommendations for video lectures, based on historical data from the VideoLectures.Net website [2]. The competition consisted of two independent tasks. In the first task it was necessary to recommend a list of "new lectures" (which had been published on the portal recently). So there was not information on popularity of the new lectures, only their detailed descriptions were available. In the second task it was necessary to recommend lectures from the entire lecture set, using information on viewed triple of lectures. The tasks are described in detail below. We do not describe the evaluation metrics used by organizers and the data offered to participants that have not been used by our algorithms. Algorithms are simple enough, universal, can be used for different problems.

2 First Task "Cold Start"

Descriptions of the lectures from VideoLectures.net website are available. Every lecture has the lecture id, the language of the lecture ("English", "Slovene", "French", etc.),

* This work was supported by the Russian Foundation for Basic Research, project 10-07-00609; by the President of the Russian Federation, project no. MD-757.2011.9. The author is also grateful to the organizers of "ECML/PKDD Discovery Challenge 2011" for running the interesting competition.

the categories of the lecture (for example "Machine Learning", "Biology"), the total (aggregated) number of views, the date when the lecture was published on the portal, the authors of the lecture (their ids, names, e-mails, homepages), the name of the lecture (a sentence in the specified language), the lecture description (a small text). The lectures were given at events (conferences, summer schools, workshops, etc.). Similar information is available on the events. Besides, for every pair of lectures the total number of the users viewed both lectures is known (if the number is more than one). Other information, for example, the descriptions of the slides or the dates, when the lectures were recorded, was also available, however it was not used in the final version of the algorithm. Note that some data is unknown (for example, descriptions are not known for all lectures).

The set of the described lectures is divided into two subsets: "older lectures" (all information is available) and "new lectures" (which have been published on the portal recently, so the viewing information is not available). A test set is a subset of the older lectures set. The task is to recommend the list of 30 new lectures for every lecture from the test set (it is recommendation of new lectures to a new user who has watched one lecture).

3 Algorithm for Solving the First Task

Let some information on a lecture can be written as the n-dimensional vector $f = (f_1, \ldots, f_n)$. For example, if n is the number of the authors of all lectures than the binary vector f describes the authors of concrete lecture: $f_i = 1$ iff the i-th author is the author of the lecture. It is similarly possible to describe the language of the lecture, its categories, etc. Naturally, the vectors describing different types of information are of different dimensionality. In each case it is possible to estimate similarity of the lectures. For example, for the i-th lecture presented by the vector $f(i) = (f_1(i), \ldots, f_n(i))$ and the j-th lecture presented by the vector $f(j) = (f_1(j), \ldots, f_n(j))$ their similarity is estimated as changed cosine similarity [3]

$$\langle f(i), f(j) \rangle = \frac{f_1(i)f_1(j) + \cdots + f_n(i)f_n(j)}{\sqrt{f_1(i)^2 + \cdots + f_n(i)^2 + \varepsilon}\sqrt{f_1(j)^2 + \cdots + f_n(j)^2 + \varepsilon}} \ . \qquad (1)$$

The change "$+\varepsilon$" is for preventing division by zero (for example, if the authorship of a lecture is unknown). In the final version of the algorithm $\varepsilon = 0.01$.

Idea of the algorithm is very simple: for the test lecture to calculate similarity to each new lecture by summing (with some coefficients) values (1) for all presented "types of information" (language, categories, authors, etc.). First, we will mark the main modification of the algorithm which essentially improves performance. Together with similarity to the lecture from the test set it is necessary to consider similarity to co-viewed older lectures (similar from the point of view of users' behavior).

Let the set of older lectures be indexed by numbers from I, let $f(i)$ be the vector of the description of the i-th lecture, let m'_{ij} be the estimation of similarity of the i-th and the j-th lectures (from the point of view of users' behavior, see below). Then let

$$f'(i) = \sum_{j \in I} \left(m'_{ij} \frac{f(j)}{\sqrt{f_1(j)^2 + \ldots + f_n(j)^2 + \varepsilon}} \right)$$

and similarity to the new t-th lecture is calculated by summing of $\langle f'(i), f(t)\rangle$ for all types of information. Let us describe how values m'_{ij} are calculated. Let L be the number of lectures, m_{ij} be the number of the users that viewed both the i-th and the j-th lectures, $i \in \{1, 2, \ldots, L\}$, $j \in \{1, 2, \ldots, L\}$, $i \neq j$, and m_{ii} be the number of views of the i-th lecture divided by 2 (such "strange" definition of the diagonal elements is a result of optimization of algorithm performance). Then

$$m'_{ij} = \frac{m_{ij}}{\sum\limits_{t=1}^{L} m_{it}} \ .$$

The sense of this value is clear enough. If the numbers m_{ii} were equal to zero, $i \in \{1, 2, \ldots, L\}$, than it would be an estimation of probability that user viewed the j-th lecture under the condition that he viewed the i-th (the performance of the algorithm was 26.02%, see below). Nonzero diagonal elements are necessary to consider also similarity to the i-th lecture, not only to co-viewed lectures (the performance was 29.06%, without division by 2 the performance was 28.17%).

Let us enumerate types of information which were used to calculate similarity. For each type we will specify the vector

$$\gamma_{\text{index}} = (\langle f'(i), f(j_1)\rangle, \ldots, \langle f'(i), f(j_r)\rangle) \ ,$$

where $J = \{j_1, \ldots, j_r\}$ is the set of new lecture indexes.

1. Similarity of categories. Here $f(j)$ is the characteristic vector of lecture categories, i.e. a binary vector, in which the t-th coordinate is equal to 1 iff the j-th lecture belongs to the t-th category. As a result we receive the vector γ_{cat}.

2. Similarity of authors. Here $f(j)$ is the characteristic vector of lecture authors, i.e. a binary vector, in which the t-th coordinate is equal to 1 iff the t-th author is the author of the j-th lecture. As a result we receive the vector γ_{auth}.

3. Similarity of languages. Here $f(j)$ is the characteristic vector of lecture language, in which the first element corresponding to English is set to 1 (to make all lectures similar on lectures in English, because lectures in English are popular among Internet users). As a result we receive the vector γ_{lang}.

4. Similarity of names. At first all words which are included into names and descriptions of the lectures are parsed and reduced to word stems (we used Porter Stemmer [4], [5]). Note, all special symbols (brackets, commas, signs of arithmetic operations, etc.) were deleted, but stop words were reserved (it does not essentially influence performance of the algorithm). The name of every lecture is described by the vector $(h_1, \ldots, h_W)$, in which h_i is the number of words with the i-th word stem. Then we apply TF-IDF-like weighting scheme [3]:

$$f_i = \frac{h_i}{\sqrt{w_i + \varepsilon}} \ , \tag{2}$$

where w_i is the total number of words with the i-th word stem in names and descriptions of all lectures. Such vectors $(f_1, \ldots, f_W)$ are used for calculation of the vector γ_{dic}. Note that for calculation of similarity of names we use information on names and descriptions (for weighting scheme). Standard TF-IDF proved to perform a bit worse ($\sim$ 1–4%).

5. Similarity of names, descriptions, names and descriptions of events. Each lecture has the name, the description (may be empty), the name of appropriate event and the event description (if information on event is not present we consider that the even name and the event description coincide with the lecture name and the lecture description). All it is united in one text, that is described by the vector $(h_1, \ldots, h_W)$, and further operations are the same as in the previous item. As a result we receive the vector γ_{dic2}.

For task solving the algorithm construct the vector

$$\gamma = 0.19 \cdot \sqrt{0.6 \cdot \gamma_{\text{cat}} + 5.6 \cdot \gamma_{\text{auth}}} + \sqrt{4.5 \cdot \gamma_{\text{lang}} + 5.8 \cdot \gamma_{\text{dic}} + 3.1 \cdot \gamma_{\text{dic2}}} \ . \qquad (3)$$

Here the square root is elementwise operation. At first the linear combination of the vectors γ_{cat}, γ_{auth}, γ_{lang}, γ_{dic}, γ_{dic2} was used. The coefficients in the linear combination were a result of optimization problem solving. But the usage of square roots gives small improvement of performance (coefficients also tuned solving optimization problem). For optimization problem the method of coordinate descent [6] was used. The algorithm recommends the lectures with the highest values of elements in the vector $\gamma = (\gamma_1, \ldots, \gamma_N)$. Such problem solving technology (selection of various types of information, construction of an appropriate linear combination, its further tuning and "deformation") is being developed by the author and is named "LENKOR" (the full description of the technology will be published in the near future).

In the final submitted solution one more change was made: the vector $\gamma = (\gamma_1, \ldots, \gamma_N)$ was transformed to the vector

$$\left(\gamma_1 \cdot \left(1 + \delta \frac{t_{\max} - t_1}{t_{\max} - t_{\min}}\right), \ldots, \gamma_N \cdot \left(1 + \delta \frac{t_{\max} - t_N}{t_{\max} - t_{\min}}\right)\right) \ , \qquad (4)$$

where t_j is the time (in days) when the j-th new lecture was published, $t_{\min}$ is the minimum among all these times, $t_{\max}$ is the maximum. The transformation increased performance approximately on 5%. The reason of the transformation is that it is important how long is lecture available online (not only popularity of the lecture). In the final version of the algorithm $\delta = 0.07$, because this value maximizes performance of the algorithm in uploads to the challenge website [1] (37.24% for $\delta = 0.09$, 37.28% for $\delta = 0.07$, 36.24% for $\delta = 0.05$).

The described algorithm has won the first place among 62 participants with the result of 35.857%. We do not describe the evaluation metric, the interested reader can find the definition of the metric on the challenge website [1]. For local testing we used the same metric. After data loading and processing (it takes 1 hour, but can be launched once for recommender system construnction) the running time for the first task solving was 17.3 seconds on a computer HP p6050ru Intel Core 2 Quad CPU Q8200 2.33GHz, RAM 3Gb, OS Windows Vista in MATLAB 7.10.0. 5704 recommendations (30 lectures in each) were calculated. The dictionary consisted of 35664 word stems.

4 Second Task "Pooled Sequences"

In the second task the training set T consists of triples $\{a, b, c\}$ of lecture numbers. For every triple the number $n(\{a, b, c\})$ of users who viewed all three lectures is known.

Besides, the pooled sequence [1] is specified. It is the ranked list of lectures which were viewed by users after all lectures of the triple $\{a, b, c\}$. The numbers of views are also known (the pooled sequence is ordered according to these values). Definition and examples of pooled sequence construction can be found on the official site of the competition [1]. We formalize this concept by means of the vector $v(\{a, b, c\}) \in \mathbb{Z}^L$, where L is the number of lectures,

$$v(\{a, b, c\}) = (v_1(\{a, b, c\}), \ldots, v_L(\{a, b, c\})) \ ,$$

$v_j(\{a, b, c\})$ is the total number of views of the j-th lecture after triple $\{a, b, c\}$ (informally speaking, it is popularity of the j-th lecture after lectures from $\{a, b, c\}$). The test set also consists of triples (the test set is not intersected with the training set). The task is to define pooled sequences for triples from the test set, to be exact 10 first members of the sequences (10 highest elements of each vector $v(\{a, b, c\})$. So, these 10 lectures should be recommended to user after viewing the three lectures.

5 Algorithm for Solving the Second Task

At first, two normalizations of the vectors corresponding to triples from the training set T are performed, the first is

$$v'(\{a, b, c\}) = \left(\frac{v_1(\{a, b, c\})}{\log(|\{\tilde{t} \in T \,|\, v_1(\tilde{t}) > 0\}| + 2)} \cdots \right.$$
$$\left. \cdots \frac{v_L(\{a, b, c\})}{\log(|\{\tilde{t} \in T \,|\, v_L(\tilde{t}) > 0\}| + 2)} \right) \ . \quad (5)$$

It is clear that $|\{\tilde{t} \in T \,|\, v_j(\tilde{t}) > 0\}|$ is the number of triples from the training set such that their pooled sequences include the j-th lecture. The reason for performing such normalization is that lectures included into many pooled sequences are generally less relevant. The second normalization is

$$v''(\{a, b, c\}) = \left(\frac{v'_1(\{a, b, c\}) \cdot \log\left(\sum\limits_{j=1}^{L} v'_j(\{a, b, c\}) + 1 \right)}{\sqrt{3 \cdot n(\{a, b, c\}) + \varepsilon}} \cdots \right.$$
$$\left. \cdots \frac{v'_L(\{a, b, c\}) \cdot \log\left(\sum\limits_{j=1}^{L} v'_j(\{a, b, c\}) + 1 \right)}{\sqrt{3 \cdot n(\{a, b, c\}) + \varepsilon}} \right) \ , \quad (6)$$

$\varepsilon = 0.01$. It is difficult enough to describe sense of this normalization. It was a result of exhaustive search of different variants and increased performance by 1–2%, that was essential in competition.

Let

$$s(\tilde{d}) = \sum_{\tilde{t}\in T\,:\,\tilde{d}\subseteq\tilde{t}} v''(\tilde{t})$$

and $n(\tilde{d}) = |\{\tilde{t} \in T\,:\,\tilde{d} \subseteq \tilde{t}\}|$ be the number of addends in the sum, T be the training set. For example, $s(\{a,b\})$ is the sum of vectors $v''(\{a,b,d\})$ for all d such that $\{a,b,d\} \in T$. Let operation ω deletes from a vector all zero elements except one and adds one zero if there was not any zero element. For example, $\omega(1,0,0,2,0) = (1,0,2)$, $\omega(1,2) = (1,2,0)$. Let

$$\mathrm{std}(x_1,\ldots,x_n) = \sqrt{\frac{1}{n-1}\sum_{t=1}^{n}\left(x_t - \frac{1}{n}\sum_{i=1}^{n} x_i\right)^2}$$

(standard deviation [7]).

The algorithm is very simple: for the triple $\{a,b,c\}$ from the test set if there are at least two nonzeros among numbers $n(\{a,b\})$, $n(\{a,c\})$, $n(\{b,c\})$ (it corresponds to having enough information) than

$$\begin{aligned}\gamma = &\frac{\log(s(\{a,b\})+0.02)}{\mathrm{std}(\omega(s(\{a,b\})))+0.5} + \\ &\frac{\log(s(\{b,c\})+0.02)}{\mathrm{std}(\omega(s(\{b,c\})))+0.5} + \\ &\frac{\log(s(\{a,c\})+0.02)}{\mathrm{std}(\omega(s(\{a,c\})))+0.5}\ .\end{aligned} \tag{7}$$

Otherwise we add to this sum addends

$$\frac{\log(s(\{a\})+0.02)}{\mathrm{std}(\omega(s(\{a\})))+0.5} + \frac{\log(s(\{b\})+0.02)}{\mathrm{std}(\omega(s(\{b\})))+0.5} + \frac{\log(s(\{c\})+0.02)}{\mathrm{std}(\omega(s(\{c\})))+0.5}\ . \tag{8}$$

Here the log is taken elementwise. Elements of the received vector γ are treated as the estimations of popularity of lectures from pooled sequence (the higher value, the more popular). Lectures with the highest estimations are recommended.

Let us try to explain the process of the development of the algorithm. It is very logical to operate simply by a rule

$$\begin{aligned}\gamma &= \log(s(\{a,b\})) + \log(s(\{b,c\})) + \log(s(\{a,c\})) = \\ &= \log(s(\{a,b\}) \cdot s(\{b,c\}) \cdot s(\{a,c\})),\end{aligned} \tag{9}$$

where "$\cdot$" is the elementwise multiplication of vectors. Indeed, if there is no information on triple $\{a,b,c\}$ we parse triples $\{a,b,d\}$ for all d. Thus we sum vectors $v(\{a,b,d\})$ and receive a vector $s(\{a,b\})$. It corresponds to union of multisets [8]. Similarly for triples $\{a,c,d\}$, $\{b,c,d\}$ we receive vectors $s(\{a,c\})$ and $s(\{b,c\})$. Now it is logical to intersect the received multisets. Standard operation for intersection in the theory of multisets is min (minimum). However in our experiment product proved to be better:

$$s(\{a,b\}) \cdot s(\{b,c\}) \cdot s(\{a,c\})\ ,$$

this operation is popular in the theory of fuzzy sets [9] for intersection (the performance was 49%, and for min the performance was 47%). The expression

$$(s(\{a,b\})+\varepsilon)\cdot(s(\{b,c\})+\varepsilon)\cdot(s(\{a,c\})+\varepsilon)$$

is needed to prevent zeroing many elements of the vector and information losses (the performance became 57%). Then experiments on normalizations of vectors and scaling were made. As a result, division by $\mathrm{std}(\omega(s(\{\cdot,\cdot\})))+0.5$ increased performance approximately by 1–3%. Adding of (8) does not influence performance, it was made "just in case".

In this solution the ideas of "LENKOR" were also used: linear combination tuning (for this reason the product (9) was written down as the sum of logs) and subsequent "deformation" (we used data normalizations). Each addend in the linear combination is a vector estimated popularity of lectures. The algorithm did not use detailed descriptions of the lectures as in the first task. In our experiments the usage of descriptions did not improve performance.

The algorithm has won the first place among 22 participants with the result of 62.415%. The algorithm running time on computer HP p6050ru Intel Core 2 Quad CPU Q8200 2.33GHz, RAM 3Gb, OS Windows Vista in MATLAB 7.10.0 for one lecture recommendation is 0.0383 seconds, full task 2 solving (60274 recommendations) takes 38.33 minutes.

The algorithms (for the first and the second tasks) can be efficiently parallelized. Calculations (1)–(9) can be made on matrices to produce several recommendations at once. For this reason we used MATLAB in our experiments: there are efficient matrix calculations in this system.

References

1. N. Antulov-Fantulin, M. Bošnjak, T. Šmuc, M. Jermol, M. Žnidaršič, M. Grčar, P. Keše, N. Lavrač: ECML/PKDD 2011 – Discovery challenge: "VideoLectures.Net Recommender System Challenge", http://tunedit.org/challenge/VLNetChallenge/
2. http://www.videolectures.net/
3. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze: Introduction to Information Retrieval, Cambridge University Press (2008). http://nlp.stanford.edu/IR-book/information-retrieval-book.html
4. Porter, Martin F.: An algorithm for suffix stripping. Program. 14(3), 130–137 (1980)
5. http://tartarus.org/∼martin/PorterStemmer/matlab.txt
6. T. Abatzoglou, B. O'Donnell: Minimization by coordinate descent, Journal of Optimization Theory and Applications. 36(2), 163–174 (1982).
7. http://en.wikipedia.org/wiki/Standard_deviation
8. Wayne D. Blizard: Multiset theory. Notre Dame Journal of Formal Logic. 30, 1, Winter, 36–66 (1989)
9. http://en.wikipedia.org/wiki/Fuzzy_mathematics

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:

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

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:

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

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

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

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.

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

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

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)

Recommending VideoLectures with Linear Regression

Martin Možina, Aleksander Sadikov, and Ivan Bratko

Faculty of Computer and Information Science
University of Ljubljana, Slovenia
{martin.mozina,aleksander.sadikov,ivan.bratko}@fri.uni-lj.si

Abstract. This paper describes our approach to the task 1 of the ECML PKDD 2011 VideoLectures.Net Recommender System Challenge. The task was to select a set of lectures to be recommended to a visitor of the VideoLecture.Net homepage after already seeing another lecture. Our proposed approach is a hybrid recommender system combining content and collaborative approaches. The core of the system is a linear regression model for predicting the rank of a lecture, whereas by rank we mean the lecture's position in the list of all lectures ordered by the interest of the visitor. Due to the complexity of the problem, the model could not be learned by a classical approach - instead, we had to employ the stochastic gradient descent optimization. The present paper furthermore, through evaluation, identifies and describes some interesting properties of the domain and of the algorithm that were crucial to achieve a higher prediction accuracy. The final accuracy of the model was enough to take the third place in the competition.

1 Introduction

In this paper, we describe our approach to the first task (cold start) of the ECML PKDD 2011 Discovery Challenge (VideoLectures.Net Recommender System Challenge) hosted by TunedIT[1]. The task was to make recommendations for video lectures on the VideoLectures.Net website.

Our proposed approach is a hybrid recommender system [2] based on linear regression for predicting the ranks of the newly acquired lectures after viewing some of the "older" lectures. First, we give a short description of the domain, of the learning problem, and present the attributes that we decided to use in learning. Afterwards, we define the problem of learning ranks, present a method for learning a linear regression based on stochastic gradient descent and use it to make content-based and collaborative-based predictions. In section 4, we evaluate the method and assess the results. We finish the paper with conclusions.

2 Domain description

The coldstart problem in the challenge was defined as:

[1] http://tunedit.org/challenge/VLNetChallenge

1. after seeing an "old" lecture (a lecture present in the system for a while),
2. recommend 30 of the "new" (published after the 1st of July 2009) lectures that the user might like. The results have to be ranked - more "likeable" lectures should come first.

The organizers collected 6983 training (old) lectures and 1122 test (new) lectures. The task of competitors was to make a selection of 30 test lectures for almost each training lecture and submit the results for evaluation.

The most important part of the provided data was the so-called pairs data set. It contained the frequencies of co-occurrences of lectures from the training set. Two lectures are said to co-occur if they were seen together (in the same browser session, not necessarily consecutively) and the user watched at least half of each of the two lectures. The co-occurrences between training and test lectures were withheld from the competitors and were used to produce the true rankings, which were used to evaluate the submitted solutions.

Provided lectures were described with the following attributes:

type of lecture (discrete; sample values: normal lecture, tutorial, debate, invited talk, etc.)
language (discrete; sample values: Slovene, English, etc.)
recorded date (date value)
published date (date value)
title of lecture (textual attribute)
description (a short textual description of the lecture)
slide titles (textual attribute)
categories (discrete; a lecture can belong to multiple categories, e.g. machine learning, reinforcement learning, etc.)
event (discrete; the event where the lecture was recorded)
author (discrete; a lecture can have several authors)
views (number of all views of the lecture; this information was given for the training lectures only)

For categories, events and authors, additional information was provided. Each category was specified by its name and a link to its description on Wikipedia. Events were described with: **type** (normal event, project, etc.), **language** of the event, **recorded** and **published dates**, **name** and a short **description** (both textual) of the event. Finally, for each author we were given his or her **name**, **email**, **homepage**, **gender**, and **organization**.

2.1 Attributes used in Learning

We manually constructed a set of attributes to be used in learning. Some of them were simply taken as originally provided, while some were combinations of the original attributes. The set of used attributes is summarized in Table 1. Published date is the number of days between the day the lecture was published and the day the snapshot of the database was taken (31st July 2010). Categories is a list of categories that the lecture belongs to. The authors attribute contains the names of actual authors and also

their domain names extracted from their email addresses. Therefore, if a lecture was presented by two persons coming from Google, lecture's value of authors attribute would be: "name of first author", "name of second author", and "google.com". Significant title words are a selection of specific words found in the titles of lectures that, as assumed, imply a higher number of views. These words are: "introduction", "tutorial", "basic", "lecture 1:", "lecture 2:","lecture 3:","lecture 4:", and "lecture". The last attribute, Title words, contains all words mentioned at least once in a title.

The constructed attributes include almost all the information provided, with the exception of the textual attributes, where only titles of lectures were used. Original attributes not included in our model are: recording date of lectures, lecture's description and slide titles, lecture's number of views, description of categories on Wikipedia, anything about events, authors homepage and organization.

Table 1. An overview of attributes used in learning. *No. of values* is the number of distinct values found in the training and test data. The column *Can have multiple values?* specifies whether a single lecture can have several values (e.g. a lecture can belong to several categories) or not.

Name	Type	No. of values	Can have multiple values?
Published date	continuous	n/a	no
Type	discrete	17	no
Language	discrete	9	no
Categories	discrete	291	yes
Authors	discrete	8109	yes
Events	discrete	520	yes
Significant title words	discrete	9	yes
Title words	discrete	12812	yes

3 Content Based Recommendations with Linear Regression

3.1 Terminology

Throughout the remainder of the paper we will use the following conventions. Lectures L_O and L_N will correspond to the "old" and "new" lectures, respectively. To address the attribute value of a lecture we will use a dot and the name of the attribute. For instance, $L_N.pub\_date$ is the published date of a new lecture. Greek letters β and γ will be used as parameters of the linear regression model. Abbreviation $Attr$ will represent the set of learning attributes from Table 1, and $Train$ will be the set of all training lectures.

3.2 The Linear Regression Model for Predicting Rank

In this section, we will describe our linear regression model for predicting the rank of a lecture L_N given that the user has already seen another lecture L_O. To learn such a

model, we needed to estimate the true ranks of lectures in the training data and use them as the values of the class variable. Let $Rank(L_O, L_N)$ be the rank of lecture L_N, given L_O has been seen by the user, and let $Pairs(L_O, L_N)$ be the number of co-occurrences of L_O and L_N taken from the pairs data set. To compute ranks $Rank(L_O, L_N)$ for all training lectures L_N, we used the standard ranking algorithm:

1. For a given training lecture L_O, create a list of co-occurrences $Pairs(L_O, L_N)$ for all L_N, where $L_N \neq L_O$ and L_N and L_O are not from the same event.
2. Order list in ascending sequence.
3. The rank $Rank(L_O, L_N)$ is the position of $Pairs(L_O, L_N)$ in the above ordered list. The pair with the lowest $Pairs(L_O, L_{low})$ gets rank 1 and the pair with the highest value gets rank that equals the length of the list. In the case of ties, a mean rank is assigned.
4. Divide all ranks $Rank(L_O, L_N)$ by the length of the ordered list;

$$Rank(L_O, L_N) \leftarrow \frac{Rank(L_O, L_N)}{\text{Length of list}}.$$

This procedure is repeated for every training lecture L_O. We avoided using lectures from the same event (first step) to make learn data more alike to test data. Namely, two lectures, one from the learn set and one from the test set, always belong to different events. The second and the third step are standard steps in ranking. The normalization in the last step was applied to avoid different maximal ranks when the lengths of lists differ.

The main reason to use ranking, instead of predicting $Pairs(L_O, L_N)$ directly, was to minimize the influence of lecture L_O. For example, if L_O is a tutorial, it will for this reason have a high number of views, and hence also $Pairs(L_O, L_N)$ will be relatively high for all L_N. On the other hand, the ranks depend only on L_N and on similarities between L_O and L_N, but not much on L_O itself.

The linear regression model for predicting rank has the form:

$$\begin{aligned}\widehat{Rank}(L_O, L_N) = \beta_0 + \beta_{date} \times min(L_0.pub\_date, L_N.pub\_date) + \\ + \sum_{a \in Attr \setminus \{pub\_date\}} \left[\sum_{v \in L_N.a} \beta_{a.v} \right] + \\ + \sum_{a \in Attr \setminus \{pub\_date\}} \left[\sum_{v \in L_N.a \cap L_O.a} \gamma_{a.v} \right] \end{aligned} \quad (1)$$

The first term β_0 is the intercept and is usually close to 0.5, which is the average normalized rank of lectures. The second term contains β_{date}, the parameter that models the influence of published date. This parameter has to be positive, as lectures with low published date were published later and therefore had less chances to be viewed. The third term is the sum over all remaining attributes and values of the lecture L_N. Each attribute value has an assigned parameter $\beta_{a.v}$ that models the increase (or decrease) of the rank of a lecture if the lecture contains this value. The last term is again the sum over all attributes, however considering only the values that are the same for L_O and L_N. The parameter $\gamma_{a.v}$, therefore, models the change of rank of L_N if both lectures have

the same value of attribute a. This parameter is usually positive, as lectures with same values are more alike, therefore it is more probable they that will be viewed together.

Such a process, where the recommendation relies only on the characteristics of the lectures, is commonly called content-based recommendation [2]. In section 4, we will describe a possible approach to extend this model to exploit some techniques from collaborative filtering.

3.3 The Learning Algorithm

The learning algorithm has to find such parameters (βs and γs) to minimize the residual sum of squares. A penalty factor λ, as in ridge regression [1], was included to prevent overfitting:

$$Err = \left[\sum_{L_O \in Train} \sum_{\substack{L_N \in Train \\ L_O.event \neq L_N.event}} \left(\widehat{Rank}(L_O, L_N) - Rank(L_O, L_N) \right)^2 \right] + \\ + \lambda * \left[\beta_{date}^2 + \sum_{a \in Attr \setminus \{pub\_date\}} \left(\sum_{v \in a} \left(\beta_{a.v}^2 + \gamma_{a.v}^2 \right) \right) \right] \quad (2)$$

Usually, the parameters of linear regression are fit easily using the formula for linear regression that involves matrix multiplication and inversion. However, considering 6983 training lectures, we have approximately $6983^2 = 48762289$ ranks to predict. Moreover, we are dealing with a large number of parameters, as adding all values from the Table 1 and multiplying by 2 sums up to 43544 parameters. The data matrix, in statistics called the design matrix, would therefore be enormous. If each multi-valued attribute would have only one value, it would still add up to 48762289*43544 values. Even with optimal coding, where each value takes only 1 byte, such matrix would still require over 2TB in memory. The data could be coded more efficiently by considering sparseness of the input (most of the values are 0), however we believe it would still present a problem for many implementations of linear regression.

Instead, we used the stochastic gradient descent algorithm. This algorithm iteratively updates the model given one learning example at a time and removes the need to have the complete data set stored. An outline of the algorithm is given in Alg. 1. The algorithm begins by setting all parameters to 0. With the third line, it commences an iteration over all attributes (including an attribute with constant value 1 to learn the β_0 parameter). In each iteration, the parameters for the values of only one attribute will be fit. The algorithm will keep optimizing parameters of one attribute until the error of the model is decreasing (5th line).

Lines 8-17 contain the core of the stochastic gradient descent algorithm. Here, the algorithm makes one sweep over the pairs of training lectures (avoiding those with the same event) and updates the values of parameters in the *updateFeatures* procedure. The update of parameters consists of the two classical steps of gradient optimization: 1) it computes the gradient of the error function (Eq. 2) on the current learning example for all relevant parameters, and 2) updates these parameters for a small negative ratio of the corresponding gradient.

The learning is governed by 4 hyperparameters of the algorithm:

Algorithm 1 Skeleton of the algorithm for fitting parameters of the linear regression model. Inputs: training data $Train$ and the hyperparameters: ITER, MP, λ, MF. Outputs: parameters βs and γs of the linear model.

set values of all parameters (βs and γs) to 0.
for $n = 1 \rightarrow$ ITER **do**
  for each a **in** $Attr$ **do**
    $oldError, newError \leftarrow 1, 0$ {To satisfy the condition in while clause.}
    while $oldError > newError$ **do**
      $oldError \leftarrow newError$
      call *shuffle*($Train$) {Shuffling order of training examples is a standard step in stochastic gradient descent optimization.}
      for each L_O **in** $Train$ **do**
        for each L_N **in** $Train$ **do**
          if $L_O.event == L_N.event$ **or** L_O in less than MP pairs **then**
            continue
          end if
          $newError \leftarrow newError + (predictRank(L_O, L_N) - trueRank(L_O, L_N))^2$
          call *updateFeatures*(L_O, L_N, a, λ, MF)
          {Function *updateFeatures* computes derivatives and updates features accordingly.}
        end for
      end for
    end while
  end for
end for

- ITER: the number of passes over attributes made by the algorithm.
- MP: the minimal number of pairs $Pairs(L_O, L_N)$ for a given L_O, where $Pairs(L_O, L_N) > 0$. A high number of non-zero pairs assures that the ranks $Rank(L_O, L_N)$ were estimated better.
- λ: the penalty factor from the error function in Equation 2.
- MF: minimal frequency of an attribute value. In order to update a parameter for an attribute value, at least MF of lectures must have this value. Otherwise, the parameter's value will remain at 0. For example, if MF=10, and since there are only 6 lectures in Russian language in the training set, the parameters related to Russian language would remain at 0.

4 Towards a Collaborative Approach

When the description of a lecture L_O is inadequate or even wrong, the content-based approach by itself cannot provide good results. In such cases, it might be better to predict using some of the lectures that were often viewed together with L_O. The question is how to select these lectures? A natural choice to measure the importance of a "friendly" lecture is the number of its co-occurrences with L_O. A weighted sum, where weights are the co-occurrences, implements this idea:

$$Coll(L_O, L_N) = \frac{\sum_{j \in Train} Pairs(L_O, L_j) * \widehat{Rank}(L_j, L_N)}{\sum_{j \in Train} Pairs(L_O, L_j)}, \tag{3}$$

where ranks $\widehat{Rank}(L_j, L_N)$ are estimated with the linear model described in the previous section.

The above formula neglects the content-based prediction $\widehat{Rank}(L_O, L_N)$ completely. A "hybrid" approach, combining content and collaborative approaches, could result in a more accurate predictor:

$$Hybrid(L_O, L_N) = \frac{\widehat{Rank}(L_O, L_N) + \mathrm{CC} * Error(L_O) * Coll(L_O, L_N)}{1 + \mathrm{CC} * Error(L_O)}, \tag{4}$$

The term $Error(L_O)$ is the mean squared error of the content-based predictor with respect to the lecture L_O:

$$Error(L_O) = \frac{\sum_{L_N \in Train}^{L_N.event \neq L_O.event} \left(\widehat{Rank}(L_O, L_N) - Rank(L_O, L_N)\right)^2}{\text{Number of lectures in Train}}. \tag{5}$$

The motivation to use the error term is to give a larger weight to the collaborative predictor when the content-based predictor makes less accurate predictions. The parameter CC sets the proportion of the error to be used as a weight.

5 Evaluation

The submitted rankings were evaluated with the mean average R-precision score (MARp), a measure commonly used in information retrieval. Initially, we tried to produce a representative holdout set from the provided training data, which would be used to optimize the algorithm's hyperparameters. However, as we were unable to sample a set that would give similar results to those on the leaderboard, we decided to use the available number of submissions (60) to select the hyperparameters.

Table 2. The results of the experiments on 20% of test data (for the leaderboard) during parameter optimization. Only the relevant experiments are shown. The initial values were: MF=10, λ = 200, MP=0, CC=0, ITER=4. The final values (in bold) are: MF=3, λ=200, MP=200, CC=3, ITER=8. Alongside the values, MARp's achieved on leaderboard are provided in brackets.

MF	10 (0.157)	5 (0.162)	**3 (0.264)**
λ	**200 (0.264)**	100 (0.263)	50 (0.262)
MP	0 (0.264)	50 (0.271)	100 (0.288) **200 (0.290)**
CC (MP=100)	0 (0.288)	**3 (0.290)**	6 (0.289)
ITER	4 (0.292)	**8 (0.307)**	

Table 2 shows some of the relevant experiments. We omit several initial experiments and all experiments not leading to the final model. At the start, the values of the hyperparameters were MF=10, $\lambda = 200$, MP=0, CC=0, and ITER=4.

We started off with larger values of MF and noticed a significant growth in performance when MF was reduced to 3. As it seems, there must be some attribute values occurring in only three training lectures, yet are critical to make good predictions for the new lectures. The following parameter tested was λ, which appeared to have a small influence on prediction. There was a slight decline in precision when λ was decreased, however the change was negligible.

After fixing MF and λ we moved on to MP. We increased its value from 0 up to 200 and noticed that values over 100 clearly lead to more accurate models. As described above, with high values of MP the algorithm will learn from lectures with better estimations of ranks. In machine learning terms, we are removing examples with large noise levels in class value. Since the increase of precision was minimal when MP went from 100 to 200, we set 200 as the final value of the parameter.

Afterwards, we investigated whether the hybrid approach helps to improve the model. The CC weighs the importance of the collaborative predictor; setting it to zero will result in the use of content predictor only, high numbers will give preference to the collaborative predictor. As it turned out (to our surprise), different values of CC did not affect the performance much. The results suggest that both approaches make similarly good predictions, but their combination brings only a slight improvement. The final value of CC was set to 3.

Lastly, we explored the ITER parameter. This parameter is the number of passes made by the algorithm over all attributes. Initially, it was set to 4 and was then increased to 8. According to the obtained results, where 8 repeats lead to notably better results, it seems that our algorithm slowly converges towards the global optimum and it would be useful to let it run even longer. However, we were out of available submissions.

The final score on the leaderboard was 0.307, while the final results on the complete test set were only 0.277. Such a difference was expected as the hyperparameters of the algorithm were optimized using the leaderboard data. This was just enough to achieve the third place in competition with a small advantage over the fourth place (0.271). The first and the second place competitors scored 0.359 and 0.307, respectively.

6 Conclusions

In this paper, we have presented a model for predicting the ranks of lectures. It is a large-scale linear regression model that can not be fit by conventional methods, hence we used the stochastic gradient descent. We implemented content-based and collaborative-based approaches. Both approaches exhibited similar prediction accuracies with a small improvement if they were used together. The evaluation showed an interesting property of the data; there are some rare attribute values occurring in only 3 (out of 6983) training lectures that are critical for a good prediction on the test data. It would be interesting to further examine this matter and seek out which are these values. Furthermore, it turned out that it is important to have reliable class values. Removing the lectures with low number of views and learning the model from well represented lectures resulted in significantly better models.

During experiments we noticed that our stochastic gradient descent only slowly converged towards an optimum. Increasing the number of passes visibly improved the accuracy of the model and we believe that letting it run even further would result in

even larger improvements. Another way of improving our model would be to consider the textual attributes, to which we did not pay much attention. Finally, it would be also useful to construct more sensible attributes for the given domain. A possible approach would be to use argument based machine learning (ABML), where new attributes are constructed from arguments (explanations) given to some of the critical learning examples [4, 3].

Acknowledgements

This work was partly supported by the Guru Cue d.o.o. company (http://www.gurucue.com/), and Slovene Agency for Research and Development (ARRS). We would also like to thank the organizers for putting up an interesting challenge.

References

1. Hoerl, A.E., Kennard, R.W.: Ridge regression: Applications to nonorthogonal problems. Technometrics 12(1), 69–82 (1970)
2. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010)
3. Možina, M., Guid, M., Krivec, J., Sadikov, A., Bratko, I.: Fighting knowledge acquisition bottleneck with argument based machine learning. In: The 18th European Conference on Artificial Intelligence (ECAI). pp. 234–238. Patras, Greece (2008)
4. Možina, M., Žabkar, J., Bratko, I.: Argument based machine learning. Artificial Intelligence 171(10/15), 922–937 (2007)

Recommender System Based on Purely Probabilistic Model from Pooled Sequence Statistics

Javier A. Kreiner[1] and Eitan Abraham[2]

[1] Center for Brain and Mind Sciences (CIMeC),
University of Trento, Rovereto (TN), Italy
javkrei@gmail.com
[2] School of Engineering and Physical Sciences,
Heriot-Watt University, Edinburgh, UK
E.Abraham@hw.ac.uk

Abstract. In this paper we present a method to obtain a recommendation ranking for items in a collection using a marginalization technique to estimate conditional probabilities. The method uses no content-related information and rests on a probabilistic model based on implicitly collected data from past user behaviour. Given a query triplet of items for which a list of recommended results is required, the technique uses estimates for the conditional probabilities of items appearing after the three doublets defined by the triplet. The technique leads to the evaluation of a score function which takes the simple form of a sum of these conditional probabilities. Results show that the approach has good performance with respect to other methods.

1 Introduction

Given the exponential growth of content availability that current technology provides, it is usually impossible for a user to even skim over each item in a collection of e.g. webpages, movies, books or products in order to make a choice. This was recognized early in the internet era and great effort has been devoted to the development of systems that assist the user in this task. The ability to separate the wheat from the chaff is, in fact, the defining element of a number of technology companies, e.g. google or yahoo, and the recommendation of items of interest is of paramount importance for many others like netflix and amazon. In a nutshell, recommender systems attempt to provide a list of elements that are likely to be of relevance to a user. The ranking of elements is based on characteristics of the items and their relationships (content-based information), information about the user in question, and information explicitly or implicitly provided by other users (collaborative-type information) [4].

The ECML/PKDD Discovery Challenge 2011 [1] had as its purpose the improvement of VideoLectures.net's [3] recommender system. The challenge was set up on the tunedit.org platform [2], which provides functionality to easily organize data mining competitions. VideoLectures.net is an open access multimedia repository of video lectures available on the internet. The videos are recordings of lectures given by researchers in diverse areas of science during scientific events such as conferences and workshops. There were two main tasks and a workflow contest during the challenge. This paper

describes a solution for task 2 which obtained second place. This task simulated the situation in which a particular user is known to have seen a set of three videos. Based on this knowledge, the system should recommend 10 videos in descending order of relevance.

The paper is organized as follows. §2 describes the available data. In §3 the main idea of the solution is introduced. §4 contains details about the implementation. §5 shows the performance of the method proposed. Finally, in §6 appear some remarks and conclusions.

2 Data and evaluation

The contestants were provided with a number of datasets containing information about the lectures and past viewing behaviour to construct a recommender system. The datasets contained two types of information. The first type was information about the content-related features of each lecture such as the author and category to which the lecture belongs. The second type was statistical information extracted from the viewing sequences of site users. Given the lack of explicit profiles, the users were identified by a cookie left in their browsers. Each sequence was determined by the stream of videos seen with a uniquely cookie-identified browser. The viewing sequences were not actually provided. Instead, aggregated information about user behaviour was given. The solution presented here disregards content-related data, and uses only statistics obtained from pooled viewing sequences.

In what follows, we introduce notation that will allow us to refer to the statistics used by the method. Let (v_1, v_2, v_3) be a triplet of three different lectures' id's, and t the id of a 'target' lecture , that is, a lecture seen after the triplet. Thus, we define the following:

1. **pairs frequencies**: the number of distinct sequences in which a pair of lectures was viewed together (not necessarily consecutively and regardless of order). We denote the co-viewing frequency of v_1 and v_2 by $f(v_1, v_2)$.
2. **triplets frequencies**: given a triplet of video lectures, the number of viewing sequences in which the three videos that define the triplet appear. $f(v_1, v_2, v_3)$ denotes this frequency.
3. **triplets' targets frequencies**: given a triplet of video lectures, the number of viewing sequences in which a given target video lecture has been seen *after* viewing the three videos in the triplet (available only for the ten targets most frequently viewed after each triplet). The notation used in this case is: $f(v_1, v_2, v_3; t)$. A semicolon to separate the target from the rest because this frequency is counted differently to the previous two.

An example follows that clarifies the last one of these definitions (adapted from [1]). Consider the following viewing sequence:

$$v_1 \to v_7 \to v_2 \to v_1 \to v_4 \to v_5 \to v_6 \to v_3$$

The first operation is to remove duplicates, after which the sequence becomes:

$$v_1 \to v_7 \to v_2 \to v_4 \to v_5 \to v_6 \to v_3$$

Suppose that we want to obtain the target viewing frequencies for the triplet (v_1, v_4, v_5). Given that only v_6 and v_3 appear after all of v_1, v_4 and v_5, we increment the two frequencies $f(v_1, v_4, v_5; v_3)$ and $f(v_1, v_4, v_5; v_6)$ by one.

Now suppose that there is another sequence:

$$v_4 \to v_5 \to v_8 \to v_1 \to v_6 \to v_3 \to v_7$$

Then, given that v_6, v_3 and v_7 appear after all of v_1, v_4 and v_5 we increment the three frequencies $f(v_1, v_4, v_5; v_3)$, $f(v_1, v_4, v_5; v_6)$ and $f(v_1, v_4, v_5; v_7)$ by one.

Hence, if these were the only two sequences containing (v_1, v_4, v_5) we would have:

$$f(v_1, v_4, v_5; v_3) = 2$$
$$f(v_1, v_4, v_5; v_6) = 2$$
$$f(v_1, v_4, v_5; v_7) = 1$$

The whole triplets' targets frequencies dataset was divided by the organizers in two parts. The first part, consisting of the target frequency information for 109044 triplets, was provided to the contestants to train their models. For the rest of the triplets, 60274, the target frequency information was retained by the organizers as a test set to score the submitted solutions. As is customarily done in data mining competitions, part of this test set was used to rank the contestants in the competition leaderboard, while the complete test set was used to construct the final ranking. The performance measure used was the Mean Average R-Precision (MARp, see [1]).

3 Approach used in the solution

Consider a given triplet (v_1, v_2, v_3) and a target t for that triplet available in the training set. Then let:

$$p(t|v_1, v_2, v_3) = \frac{f(v_1, v_2, v_3; t)}{f(v_1, v_2, v_3)} \tag{1}$$

denote the conditional probability of seeing t given that v_1, v_2, v_3 have been seen previously in any order. In a similar way, if we had available $f(v_1, v_2; t)$, the number of sequences in which t appears after (v_1, v_2), we could calculate

$$p(t|v_1, v_2) = \frac{f(v_1, v_2; t)}{f(v_1, v_2)} \tag{2}$$

i.e. the conditional probability of seeing t after having seen the doublet (v_1, v_2).

The solution is based on the observation that it is possible to estimate this conditional probability of seeing a video after having seen a doublet of videos, based on the triplets' information available in the training set:

$$p(t|v_1, v_2) \approx \hat{p}(t|v_1, v_2) := \frac{\sum_v f(v_1, v_2, v; t)}{\sum_v f(v_1, v_2, v)} \tag{3}$$

As a matter of consistency with the numerator as regards the counting of sequences, we chose the denominator as a sum over the triplets in the training set. Consequently, this is an approximation for the following reasons: (a) there is an unknown overlap between the set of sequences counted by $f(v_1, v_2, v; t)$ and $f(v_1, v_2, v'; t)$, and also between the set of sequences counted by $f(v_1, v_2, v)$ and $f(v_1, v_2, v')$ for two different videos v and v', (b) there is no information available for target videos that are not in the top ten most frequently seen videos for each triplet, and (c) there are triplets that have been retained for the test set for which there is no target information. In any case, this "marginalization" over the third video would seem to provide a reasonable estimate.

Consider now a query triplet (q_1, q_2, q_3). The task is to identify, in descending order of viewing frequency, the ten videos most frequently viewed after having seen q_1, q_2, q_3 in any order. Using conditional probability estimates $\hat{p}(t|q_1, q_2)$, $\hat{p}(t|q_1, q_3)$, $\hat{p}(t|q_2, q_3)$, the ranking is constructed based on some function of them, i.e. let

$$\text{score}(t) = F(\hat{p}(t|q_1, q_2), \hat{p}(t|q_1, q_3), \hat{p}(t|q_2, q_3)) \tag{4}$$

be the score assigned to video t. It is reasonable to postulate some restrictions for F. Namely, that it should be increasing, or at least non-decreasing, in each of its arguments, and also that it should obtain the same value for any permutation of its arguments. A number of options were tested for F, among them the product of the arguments. In the end, the sum of the conditional probabilities happened to give a very good result, i.e.

$$\begin{aligned} &F(\hat{p}(t|q_1, q_2), \hat{p}(t|q_1, q_3), \hat{p}(t|q_2, q_3)) = \\ &= \hat{p}(t|q_1, q_2) + \hat{p}(t|q_1, q_3) + \hat{p}(t|q_2, q_3) \end{aligned} \tag{5}$$

Since the original submission of these results we found, however, that there is another option that provides a superior ranking. This is an entropy-like function defined as,

$$\begin{aligned} &F(\hat{p}(t|q_1, q_2), \hat{p}(t|q_1, q_3), \hat{p}(t|q_2, q_3)) = \\ &= -\sum_{1 \le i < j \le 3} \hat{p}(t|q_i, q_j) \log(\hat{p}(t|q_i, q_j) \end{aligned} \tag{6}$$

which also enjoys the same permutation symmetry.

4 Implementation details

In order to render the method computationally viable, as a first step, the target conditional probability estimates given a doublet are calculated and stored in an index. In the index, each doublet points to a list of targets that appeared after the doublet with the corresponding conditional probability estimate (see Equation (3)). This is done by traversing the triplets' training set once and accumulating the frequencies for each doublet and target.

Once this index is constructed, given a query triplet (q_1, q_2, q_3), consider the three doublets (q_1, q_2), (q_1, q_3), and (q_2, q_3). The index contains for each of these doublets a list of target videos with corresponding conditional probabilty estimates. Let L_1, L_2 and L_3 be the list of targets in the index for (q_1, q_2), (q_1, q_3), and (q_2, q_3), respectively.

Additionally, let L be the list of videos appearing simultaneously in L_1, L_2 and L_3, i.e. $L = L_1 \cap L_2 \cap L_3$. For these targets, which would seem particularly relevant for the query, we can readily use the scoring function F to rank them. In the case that the number of results thus obtained is at least ten (the number of required recommendations for the task), the top ten sorted recommendations are written to an output file and we are done with this query triplet.

For some query triplets it may happen that there are less than ten targets in the intersection list L (in the query file this happened for 20688 of the total of 60274 query triplets). If this happens, consider the targets t that are in the intersection of exactly two of the lists i.e.

$$t \in L' := (L_1 \cap L_2) \cup (L_1 \cap L_3) \cup (L_2 \cap L_3) \setminus L$$

for these t's two of the probability estimates are available. They are ranked, again using the function F, and appended after the previous recommendations.

If at this point the recommendation list still has less than ten results (3372 of the 60274 queries), consider the targets that appear exactly in one of the three lists, i.e.

$$t \in L'' := L_1 \cup L_2 \cup L_3 \setminus ((L_1 \cap L_2) \cup (L_1 \cap L_3) \cup (L_2 \cap L_3))$$

For these targets one probability estimate is available. Once again, they are ranked using F and added to the recommendation list.

There are still some query triplets for which a list of at least ten recommendations cannot be obtained (412 of 60274 queries), for these the conditional probability estimates of targets appearing after single videos $\hat{p}(t|q_1)$, $\hat{p}(t|q_2)$, and $\hat{p}(t|q_3)$ are calculated "marginalizing" over two videos:

$$p(t|q) \approx \hat{p}(t|q) := \frac{\sum_{v,w} f(q,v,w|t)}{\sum_{v,w} f(q,v,w)} \tag{7}$$

Now, the score for ranking given to a target for query triplet (q_1, q_2, q_3) is:

$$\hat{p}(t|q_1) + \hat{p}(t|q_2) + \hat{p}(t|q_3) \tag{8}$$

Finally, there are some triplets in the query file for which after carrying out the previous steps still less than 10 recommendations are obtained (29 of 60274), in this case the video pairs co-viewing frequencies are used to generate the remaining recommended video lectures.

The method described above obtained a score of 0.60749 on the leaderboard and a score of 0.61134 on the complete test set. The final solution submitted, which obtained a score of 0.60791 on the leaderboard and a score of 0.61172 on the complete test set, included a coefficient per doublet that was fitted and incorporated into the scoring function. The rationale behind these coefficients was that they might adjust for some of the inaccuracies in calculating the conditional probability estimates discussed above.

Consider a model introducing these coefficients:

$$F(p(t|q_1,q_2), p(t|q_1,q_3), p(t|q_2,q_3)) =$$

$$= \lambda_{q_1,q_2} \times p(t|q_1,q_2) + \lambda_{q_1,q_3} \times p(t|q_1,q_3) + \lambda_{q_2,q_3} \times p(t|q_2,q_3) \quad (9)$$

To fit these coefficients a greedy grid search method was used. Accordingly, each coefficient was initialized to 1. Given a doublet for which the coefficient needs to be fitted, a greedy search was conducted over a range (range used for the solution: $[0.5-1.5]$) and the coefficient value was selected for which the evaluation metric (MARp) over the available training triplets that contained the doublet was maximal. After greedy fitting these coefficients sequentially for each doublet and calculating the recommendations for the test set using Equation (9) for the score function, the leaderboard score improved to 0.60791. The Wilcoxon signed rank test was used to assess whether the difference was statistically significant. The p-value obtained was 0.01462, which does not lend strong support to the hypothesis that the means are different.

5 Results

Table 1 contains the scores for the leaderboard and complete test sets obtained using the methods described. The table also shows the score obtained when using a solution based on single videos conditional probability estimates only (see Equation (7)). Additionally, the scores obtained by the winner solution and the third place solution are included for comparison.

Again, the Wilcoxon signed rank test was used to assess statistical significance of the difference in scores between the original method using the simple sum as ranking function and the one using an entropy-like ranking function. The test yielded a p-value in the order of 10^{-16}, confirming that the difference in scores is statistically significant.

Table 1. Scores for the three methods described

Method	Leaderboard Score	Complete Test Set Score
Singles Cond. Probs.	0.41844	0.42057
Doublets Cond. Probs.	0.60749	0.61134
Doublets Cond. Probs. with Coeffs.	0.60791	0.61172
Doublets Cond. Probs. entropy-like F	0.60910	0.61285
Winner Solution (D'yakonov Alexander)	0.62102	0.62415
Third Place Solution (Vladimir Nikulin)	0.58727	0.59063

6 Summary and Conclusions

The method presented here uses a purely probabilistic approach to construct a recommender system from pooled viewing sequences statistical data. To do this we introduced an approximate marginalization technique leading to an estimate of the conditional probabilities of viewing target videos given that a doublet has been seen. These are in turn combined using a fully symmetric function to calculate a ranking score.

Introduction of doublet-dependent coefficients did not improve the performance in a statistically significant amount according to the Wilcoxon signed rank test. On the

other hand, replacing a simple sum by an entropy-like function for the ranking function yielded a statistically significant higher performance, again as assessed by the same statistical test.

The technique is straightforward, intuitively sound, being based on a simple insight, and easy to implement. Furthermore, it displayed better performance compared to other methods, obtaining second place on task 2 of the ECML/PKDD Discovery Challenge 2011.

References

1. ECML/PKDD Discovery Challenge 2011, http://tunedit.org/challenge/vlnetchallenge
2. tunedIT.org website, http://www.tunedit.org
3. VideoLectures.net website, http://www.videolectures.net
4. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17, 734–749 (2005)

OpenStudy: Recommendations of the Following Ten Lectures After Viewing a Set of Three Given Lectures

Vladimir Nikulin

Department of Mathematics
University of Queensland
vnikulin.uq@gmail.com

Abstract. We propose to use two lectures (out of the given triplet of three lectures) in order to define a direction of prediction, which includes the set of predicted lectures accompanied by the corresponding frequencies. The relevance of the whole predicted set is calculated according to the remaining third lecture. Further improvements were achieved with homogeneous ensembles, based on the random sampling known, also, as bagging. The experimental results were obtained online during the VideoLectures.Net ECML/PKDD 2011 Discovery Challenge (Track N2).

1 Introduction

VideoLectures.Net is a free and open access multimedia repository of video lectures, mainly of research and educational character. The lectures are given by distinguished scholars and scientists at the most important and prominent events like conferences, summer schools, workshops and science promotional events from many fields of science. The website is aimed at promoting science, exchanging ideas and fostering knowledge sharing by providing high quality didactic contents not only to the scientific community but also to the general public. All lectures, accompanying documents, information and links are systematically selected and classified through the editorial process taking into account also users' comments[1].

The tasks of the challenge were focused on making recommendations for video lectures, based on historical data from the VideoLectures.Net website[2].

According to [1], Open Social Learning Systems open new prospects for millions of self-motivated learners to access online a high quality materials. It is estimated that there will be 100 million students qualified to enter universities over next decade. Universities have responded to this need with Open Education Resources: thousands of free, high quality online courses, developed by hundreds of faculty, used by millions worldwide. Unfortunately, online courseware does not offer a supporting learning experience or the engagement needed to keep students motivated.

However, students today care deeply about their education. Four out of five stress about their grades. To create a successful online learning experience for those students,

[1] http://tunedit.org/challenge/VLNetChallenge
[2] http://videolectures.net/

there are two main issues that need to be addressed: 1) creation of online study materials (content), and 2) an engaging online interaction experience (community).

The core problem (and the main subject of this study) lies around the second issue: how to give the right direction to a generation of learners, who live on the Internet, in the wide space of the available research/educational resources.

Recommender systems attempt to profile user preferences over items, and model the relation between users and items. The task of recommender systems is to recommend items that fit a users tastes, in order to help the user in selecting/purchasing items from an overwhelming set of choices [2]. Such systems have great importance in applications such as e-commerce, subscription based services, information filtering, etc. Recommender systems providing personalized suggestions greatly increase the likelihood of a customer making a purchase compared to unpersonalized ones. Personalized recommendations are especially important in markets where the variety of choices is large, the taste of the customer is important, and last but not least the price of the items is modest. Typical areas of such services are mostly related to art (esp. books, lectures, movies, music), fashion, food and restaurants, gaming and humor.

The most of the methods presented and discussed in [2] were motivated by the famous Netflix Cup. Those methods, where matrix factorization is the most common, cannot be applied in our case directly, because the structure of the data is different. In our case we are dealing not with a specific users, but with abstract users who had seen in the past a triplet as a set of three given lectures, where an exact sequence of lectures is not given. We shall consider in Section 4 one suitable modification (as a prospective direction) of the gradient-based matrix factorization (GMF) [6] as an example of stochastic gradient descent algorithm.

Traditional data mining techniques such as association rules were tried with good results at the early stages of the development of recommender systems [3]. Frequent item sets, discovered as part of association rule mining, represent the least restrictive type of navigational patterns, since they focus on the presence of items rather than the order in which they occur within user session [4]. Frequency-based methods are the primary tool in the following below Sections 3.1 - 3.6. Note, also, that Markov decision processes provide a more advanced model for recommender systems (in the case if the sequence of the states is given). According to the Markov Chain Model, we are dealing with a finite space of possible states, and, using a maximum-likelihood estimate (applied to the historical data) as a transition function, we can formulate a prediction [5].

Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class [7]. In Section 3.6 we consider method of random resampling: it is supposed that using the hundreds of predictors (base learners) based on the subset of the whole training set we shall reduce the random factors. According to the principles of homogeneous ensembling, the final predictor represents an average of base predictors. As a reference, we mention random forests as a well-known example of successful homogeneous ensemble. However, the construction of random forests is based on another method, which is linked to the features but not to the samples.

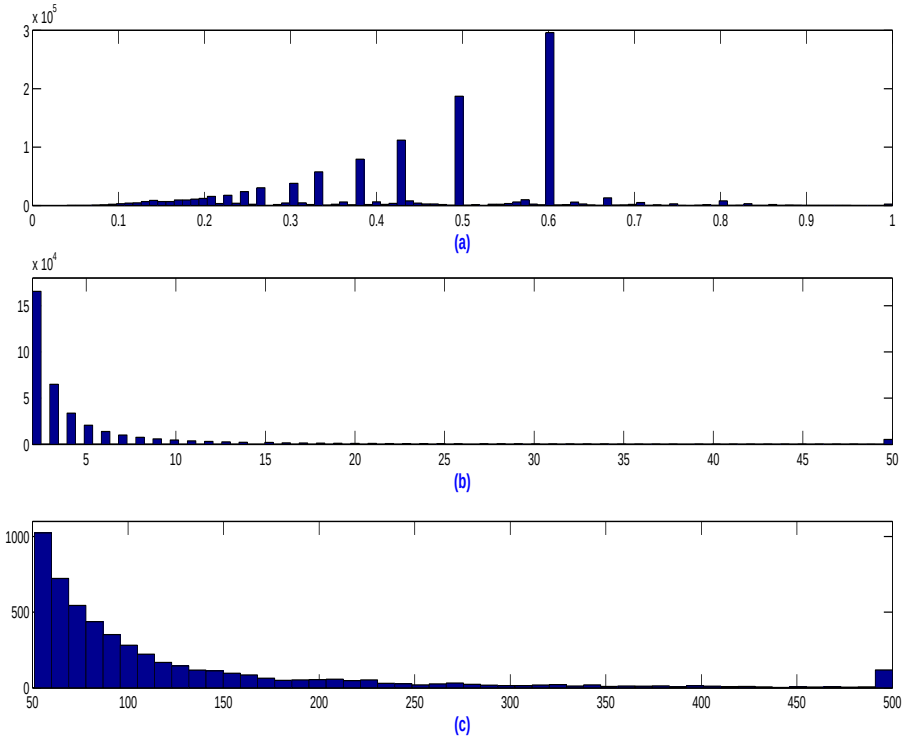


Fig. 1. Histograms of (a) triplet' frequencies; (b) pair' frequencies truncated to the level of 50; (c) pair' frequencies bigger than 50, and truncated to the level of 500, see for more details Section 2.

2 Data and some definitions

The training database includes two sets named 1) pairs P and 2) triplets T with two parts left and right, where the left part contains input triplets and the corresponding numbers of views, the right part contains output lectures and the corresponding numbers of views.

2.1 Pair data

Let us denote by I_P set of indexes corresponding to the pairs data. Any element of I_P represents a set (without order) of two lectures $I = \{a, b\}$, where $I \in I_P$. We shall understand under $P_I = P_{ab}$ number of times lectures a and b were viewed/seen together.

2.2 Triplet data

Let us denote by I_T set of indexes corresponding to the triplets data. The element with index $I \in T$ has two values (one for the left, and one for the right parts): $\tau_I = \{a, b, c\}$ is a triplet or a set of three lectures a, b and c. Under T_I we shall understand the number of times those three lectures were seen together. Further, L_I is a set of single lectures, which were seen after τ_I. Under $T_I(\ell), \ell \in L_I$, we shall understand the number of times lecture ℓ was seen after the triplet τ_I.

2.3 Graphical illustrations

Figure 1(a) illustrates histogram of empirical probabilities or frequencies

$$r_\tau(\ell) = \frac{T_I(\ell)}{T_I}, \ell \in L_I, I \in I_T, \tag{1}$$

where we replaced I by τ in the left side of (1), because there is unique correspondence between I and τ.

Figure 1(b-c) illustrates histogram of frequencies

$$P_I, I \in I_P,$$

where all the values in Figure 1(b) were truncated to 50, Figure 1(c) illustrates histogram of the values P_I bigger than 50, which were truncated to 500.

3 Methods

3.1 Predictions with couples

The task of the Challenge was to make predictions according to the test dataset V, which has the same structure as T (left part). In more details, the task was to make a recommendation of the ten the most appropriate lectures after viewing the given triplet.

Remark 1. As a particular and very important feature of this Challenge, we note the absence of the same triplets in both training T (left part) and test V sets. At the same time, we can report a very significant proportion of the same sets of *two* lectures (*couples*) in both training and test sets.

In total, we found $n_c = 34756$ couples, with number of record (which were extracted from the right part of T) per couple ranging from 1 to $m_c = 4020$. Note that any *triplet* may be considered as a set of three *couples*. There are $N_V = 60274$ triplets in the test set V, and 1) we did not find any related couples in the training set only in 116 cases, 2) we found one couple in 829 cases , 3) we found two couples in 4705 cases, and 4) we found all three couples in an absolute majority of 54624 cases.

Remark 2. Any lecture is identified by the index, where the biggest index is $n_L = 13251$. However, not all n_L lectures were used. We had assumed that the predicted lectures should be found in the right part of T, where we identified only $n_s = 5209$ different lectures.

Some preliminary definitions We shall explain how the system works in the terms of the n_s secondary indexes, because the transformation to the original n_L indexes is a trivial one. Our database was organised as follows. Squared matrix A with sizes $n_s \times n_s$ contains n_c different addresses of the matrix B with sizes $n_c \times m_c$.

Firstly, we shall find three couples $\alpha_{ij}, j = 1, \ldots, 3$, for any triplet $\tau_i, i = 1, \ldots, N_V$, in the test dataset V. Then, for any couple α_{ij} we shall find (according to the matrix A) the corresponding address $\beta(\alpha_{ij})$ (in the matrix B) and the number of records $n(\alpha_{ij})$, where $1 \leq \beta \leq n_c, 1 \leq n \leq m_c$.

Under the element of matrix B we shall understand predicted/recommended lecture ℓ and the corresponding frequency

$$\{\ell, r_\tau(\ell)\}, \tag{2}$$

where r_τ is defined in (1).

Remark 3. The main advantage of the above method is its speed: the algorithm will go through the whole test set V and will output required solution for the Track N2 within 5 min.

An update process Now, we shall describe the most critical step of the computational process. Any particular triplet τ from the test dataset is to be considered in an identical manner, so we omit index i in order to simplify notations.

Suppose that initially all the ratings are set to zero $s(\ell) = 0, \ell = 1, \ldots, n_s$, where $s(\ell)$ is the rating of the corresponding lecture ℓ which will be used for the final ranking as an output of this model.

This is the most important update formula

$$s(B_{\beta k}(1)) += B_{\beta k}(2), k = 1, \ldots, n(\alpha_j), j = 1, \ldots, 3, \tag{3}$$

where $B_{\beta k}(1)$ is the lecture index, and $B_{\beta k}(2)$ is the corresponding frequency defined in (2).

After computation of the vector s according to (3), we shall sort it in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest s (from the top to the bottom) are to be submitted as a solution.

Remark 4. In the case if the number of positive values in the vector s is smaller compared to the required 10, we shall generate remaining indexes at random assuming that the indexes are different compared to 1) the indexes of the input triplet τ_i plus 2) those indexes which were selected already.

The method, as described above produced the score on the Leaderboard **0.49568**, where the detailed definition (with numerical examples) of the Competition score is available from the web-site of the PKDD 2011 Contest.

3.2 Predictions with singles

Essentially, predictions with *singles* (single lectures) work similarly as predictions with couples. However, there are some differences, which could be treated as simplifications.

We had found that the maximum number of the records corresponding to the single lecture is $m_s = 77798$. Accordingly, the matrix $\widehat{B}$ (as a replacement to the matrix B in the previous Section 3.1) has sizes $n_s \times m_s$.

The model works in the following way: by definition, any triplet represents a set of three lectures $\ell_j, j = 1, \ldots, 3$. We shall find the number of records $1 \le n(\ell) \le m_s$, where $1 \le \ell \le n_s$.

An update process Again, initial ratings of lectures are set to zero: $s(\ell) = 0, \ell = 1, \ldots, n_s$.

This is the main update formula

$$s(\widehat{B}_{\ell_j k}(1)) += \widehat{B}_{\ell_j k}(2), k = 1, \ldots, n(\ell_j), j = 1, \ldots, 3. \tag{4}$$

After computation of the vector s according to (4), we shall sort it in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest s (from the top to the bottom) are to be submitted as a solution.

The method, as described above produced the score on the Leaderboard **0.33278**.

3.3 Predictions with pairs

Definition 1. *We shall call that the lectures a and b are P-linked if $P_{ab} \ge 1$. According to the symmetric matrix P, we define set $H(a)$ of all P-linked lectures to the lecture a.*

An update process As before, initial ratings of lectures are set to zero: $s(\ell) = 0, \ell = 1, \ldots, n_s$. Then, we shall apply an update formula

$$s(d) += P(\ell_j, d), d \in H(\ell_j), j = 1, \ldots, 3, \tag{5}$$

where an interpretation/definition of the lectures ℓ_j is the same as in (4).

After computation of the vector s according to (5), we shall sort it in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest s (from the top to the bottom) are to be submitted as a solution.

The method, as described above produced the score on the Leaderboard **0.12677**.

Remark 5. The solution, as described in this section, was recommended by the Organisers on the forum as "*simple pairs solution*".

Note, also, that during our numerous experiments we made a very interesting observation/discovery.

Remark 6. Statistics defined in (3-5) represent a sum of frequencies. It is very interesting to note that the results will be significantly poorer if we shall apply an average of frequencies as an alternative to the sums.

3.4 Predictions with weighted couples

According to the above three sections, predictions with the couples produced best results. We decided to go further and to take into account remaining third lectures ϕ and ψ in both training and test sets.

Motivation: in the case if remaining ("extra") lectures ϕ and ψ are closer (have bigger number of the joint views according to the pairs data), the predicted direction, corresponding to the related *couple*, must be given bigger weight.

As it was discussed in Remark 1, lectures ϕ and ψ are different by definition. In other words, the corresponding ("similar") triplets in the training and test datasets may be represented as

$$\alpha_j \cup \phi_j,\ \alpha_j \cup \psi_j,$$

where $\phi_j \neq \psi_j, j = 1, \ldots, 3$.

An update process As before, initial ratings of lectures are set to zero: $s(\ell) = 0, \ell = 1, \ldots, n_s$. Then, we can re-write (3) in this way

$$s(B_{\beta k}(1)) += w(P(\phi_j, \psi_j))B_{\beta k}(2), k = 1, \ldots, n(\alpha_j), j = 1, \ldots, 3, \tag{6}$$

where w is an increasing weight function. In our final submission we used very simple linear function: $w(x) = 0.01 \cdot x + 0.005$.

After computation of the vector s according to (6), we sort it in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest s (from the top to the bottom) are to be submitted as a solution.

The method with weighted couples, as described above, produced very significant improvement on the Leaderboard **0.58145**.

3.5 Predictions with weighted singles

This section may be regarded as an extension of Section 3.2. In some sense, prediction with weighted singles is similar to the prediction with weighted couples, Section 3.4. However, there are some differences. In the case of singles, we are defining direction of the prediction according to the single lectures. Accordingly, we have two other ("extra") lectures, which should be compared properly with two lectures in the corresponding triplet of the training data.

An update process Updates were conducted according to

$$s(\widehat{B}_{\ell_j k}(1)) += w(\phi_{1j}, \phi_{2j}; \psi_{1j}, \psi_{2j}) \cdot \widehat{B}_{\ell_j k}(2), k = 1, \ldots, n(\ell_j), j = 1, \ldots, 3, \tag{7}$$

where $w(\phi_{1j}, \phi_{2j}; \psi_{1j}, \psi_{2j})$

$$= 0.0005(P(\phi_{1j}, \psi_{1j})P(\phi_{2j}, \psi_{2j}) + P(\phi_{1j}, \psi_{2j})P(\phi_{2j}, \psi_{1j})) + 0.01.$$

The motivation behind the above formula is a very simple: we must ensure that any "extra" lecture from the test triplet is close to at least one "extra" lecture from the train triplet.

After computation of the vector s according to (7), we sort it in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest s (from the top to the bottom) are to be submitted as a solution.

The method, as described in this section, produced the score **0.4529** on the Leaderboard.

3.6 Resampling method (the final recommender)

In this section computation of the single ranking vector s was based on 75% of randomly selected samples. In an absolute majority of all 60274 test instances, the number of positive components of the vector s defined in (6) is greater than 100. So we shall consider this case only.

Let us denote vector of secondary ratings as z, which is set to zero at the beginning of the whole resampling process. We conducted 200 random samplings (global iterations). After any global iteration, only 100 top lectures (components of the vector z) received increments ranging from 1 to 100 votes (bigger for better performance). The method which we used within any global iteration (base learner) is described in Section 3.4.

After completion of all 200 global iterations, we sorted vector z in a decreasing order, and the arguments (indexes of the lectures) corresponding to the ten biggest z (from the top to the bottom) were submitted as a solution.

The final model with resampling, as described above, produced the following score on the Leaderboard **0.58727**. This solution was used as a final.

Table 1. Distances (8) between five solutions described in Sections 3.1 - 3.4 and 3.6.

N	Method	Score	1	2	3	4	5
1	couples	0.49568	0	0.2605	0.2137	0.6394	0.6517
2	singles	0.33278	0.2605	0	0.5832	0.4269	0.4327
3	pairs	0.12677	0.2137	0.5832	0	0.1565	0.1664
4	wgt-couples	0.58145	0.6394	0.4269	0.1565	0	0.91
5	resampling	0.58727	0.6517	0.4327	0.1664	0.91	0

3.7 Statistical comparison of different solutions

The distances in the above Table 1 were computer using simplified version of the PKDD 2011 Contest evaluation method. Any solution represents an integer matrix of $N_V \times 10$, $\mathcal{T} = 10N_V$ integer indexes in total. By comparing two matrices, we shall find the number of common indexes (intersection) in any row. The total number of all intersections will give us a numerator $\mathcal{R}$, and the required distance represents a ratio

$$\mathcal{D} = \frac{\mathcal{R}}{\mathcal{T}}. \tag{8}$$

3.8 Computation time

A Linux multiprocessor workstation with speed 3.2GHz and 16GB RAM was used for the most of the computations, which were conducted according to the specially developed codes in C. The computation of the final solution, as described in Section 3.6, was conducted overnight and took about 12 hours.

4 Gradient-based matrix factorization

The main idea behind an approach of this section is to factorize the transition matrix $\mathbf{X}$ of a Markov chain (or matrix of frequencies) [9].

By definition, $\mathbf{X}$ is a squared symmetrical matrix with size n_s, and any element of $\mathbf{X}$ reflects similarity between the corresponding lectures (the bigger value indicates the higher level of similarity).

Further, we shall consider factorization of the matrix $\mathbf{X}$

$$\mathbf{X} \sim \mathbf{G}\mathbf{G}', \tag{9}$$

where factor matrix $\mathbf{G}$ has sizes $n_s \times k$.

We shall compute matrix $\mathbf{X}$ according to an update formula, which is very similar to (4). Initially, all the values of the matrix $\mathbf{X}$ are set to zero. The update process will be conducted according to the formula

$$x(\widehat{B}_{\ell k}(1), \ell) += \widehat{B}_{\ell k}(2),\ \ x(\ell, \widehat{B}_{\ell k}(1)) = x(\widehat{B}_{\ell k}(1), \ell), \tag{10}$$

where $k = 1, \ldots, n(\ell), \ell = 1, \ldots, n_s$.

In this section, boldface capital letters denote matrices or vector-columns, while normal letters denote elements of matrices. Also, it will be convenient for us to use notation $x_{ab} = x(a, b)$.

We now describe the procedure for undertaking the matrix factorization (9). The matrix factorization represents a gradient-based optimisation process with the objective to minimise the following squared loss function:

$$\mathcal{L}(\mathbf{A}) = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} e_{ab}^2, \tag{11}$$

where $e_{ab} = x_{ab} - \sum_{f=1}^{k} g_{af} g_{fb}$.

The above target function (11) includes in total kn regulation parameters and may be unstable if we minimise it without taking into account the mutual dependence between elements of the factor matrix $\mathbf{G}$.

As a solution to the problem, we can go consequently through all the differences e_{ab}, minimising them as a function of the particular parameters which are involved in the definition of e_{ab}. Compared to usual gradient-based optimisation, in our optimisation model we are dealing with two sets of parameters, and we should mix uniformly updates of these parameters, because these parameters are dependent.

Algorithm 1: Gradient-based matrix factorization.

1. **Input**: $\mathbf{X}$ - similarity matrix.
2. Select M - number of global iterations; $k \geq 1$ - number of factors; $\lambda > 0$ - learning rate.
3. Initial similarity matrix $\mathbf{G}$ is generated randomly.
4. **Global** cycle: repeat M times the following steps 5 - 15:
5. **external**-cycle: for $a = 1$ to $n_s - 1$ repeat steps 6 - 15:
6. **internal**-cycle: for $b = a + 1$ to n_s repeat steps 7 - 15:
7. compute prediction $S = \sum_{f=1}^{k} g_{af} g_{fb}$;
8. compute error of prediction: $\Delta = x_{ab} - S$;
9. internal factors-cycle: for $f = 1$ to k repeat steps 10 - 15:
10. compute $\alpha = g_{af} g_{fb}$;
11. update $g_{af} \Leftarrow g_{af} + \lambda \cdot \Delta \cdot g_{fb}$ (see (12a));
12. $\Delta \Leftarrow \Delta + \alpha - g_{af} g_{fb}$;
13. compute $\alpha = g_{af} g_{fb}$;
14. update $g_{fb} \Leftarrow g_{fb} + \lambda \cdot \Delta \cdot g_{af}$ (see (12b));
15. $\Delta \Leftarrow \Delta + \alpha - g_{af} g_{fb}$;
16. **Output**: $\mathbf{G}$ - matrix of latent factors.

The following partial derivatives are necessary for Algorithm 1:

$$\frac{\partial e_{ab}^2}{\partial g_{af}} = -2 e_{ab} g_{fb}, \tag{12a}$$

$$\frac{\partial e_{ab}^2}{\partial g_{fb}} = -2 e_{ab} g_{af}, \tag{12b}$$

where $a = 1, \ldots, n_s - 1, b = a + 1, \ldots, n_s$.

Remark 7. The content within this section represents rather a direction for a prospective work. We are thinking that transformation by the logit function of the values of the matrix $\mathbf{X}$ will work better with Algorithm 1.

4.1 Ranking of the lectures with the matrix of latent factors

As an outcome, Algorithm 1 produces the matrix $\mathbf{G}$ of latent factors. Accordingly, we can characterize any lecture by the corresponding vector-row of k numerical values, and can compute a proximity measure with the given three lecture (triplet) from the test set V. The smaller value of the distance indicates higher preference.

5 Concluding remarks

We fully agree with [10] that the superiority of new algorithms should always be demonstrated on an independent validation data. In this sense, an importance of the data mining contests is unquestionable. The rapid popularity growth of the data mining

challenges demonstrates with confidence that it is the best known way to evaluate different models and systems.

In general terms, we are satisfied with our results. However, due to the lack of available time, we did not find an efficient way how to construct heterogeneous ensembles. For example, the outcomes of the methods of weighted singles and couples are very different. Nevertheless, both results are competitive, and it is very important to find out how to exploit the differences in order to produce more advanced solution.

Also, we did not explore in details gradient-based matrix factorization, which is presented in Section 4. We can expect that this method may lead (after proper adjustment and modification) to the solution, which will be different and competitive at the same time.

Acknowledgments

This work was supported by a grant from the Australian Research Council. Also, we are grateful to the Organisers of the PKDD 2011 data mining Contest for this stimulating opportunity. Many thanks to the reviewers for very helpful comments and advices, including some very important references.

References

1. Ram A.,Ai H.,Ram P. and Sahay S. (2011) Open Social Learning Communities. *In International Conference on Web Intelligence, Mining and Semantics*, Sogndal, Norway.
2. Takacs G., Pilaszy I., Nemeth B. and Tikk D. (2009) Scalable Collaborative Filtering Approaches for Large Recommender Systems. *Journal of Machine Learning Research*, **10**, 623-656.
3. Agrawal R. and Srikant R. (1994) Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 32p.
4. Mobasher B., Dai H., Luo T. and Nakagawa M. (2002) Using Sequential and Non-Sequential Patterns in PredictiveWeb Usage Mining Tasks. *ICDM 2002*, 4p.
5. Shani G., Heckerman D. and Brafman R. (2005) An MDP-Based Recommender System. *Journal of Machine Learning Research*, **6**, 1265-1295.
6. Nikulin V., Huang T.-H., Ng S.-K., Rathnayake S. and McLachlan G.J. (2011) A Very Fast Algorithm for Matrix Factorisation. *Statistics and Probability Letters*, **81**, 773-782.
7. Breiman L. (1996) Bagging Predictors. *Machine Learning*, **24**, 123-140.
8. Breiman L. (2001) Random Forests. *Machine Learning*, **45**, 5-32.
9. Rendle S., Freudenthaler C. and Schmidt-Thieme L. (2010) Factorizing Personalized Markov Chains for Next-Basket Recommendation. *WWW 2010, North Carolina, USA*, 811-820.
10. Jelizarow M., Guillemot V., Tenenhaus A., Strimmer K. and Boulesteix A.-L. (2010) Over-optimism in bioinformatics: an illustration. *Bioinformatics*, **26**(16), 1990-1998.

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

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

and event attribute information, the training data includes information about pairs and 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

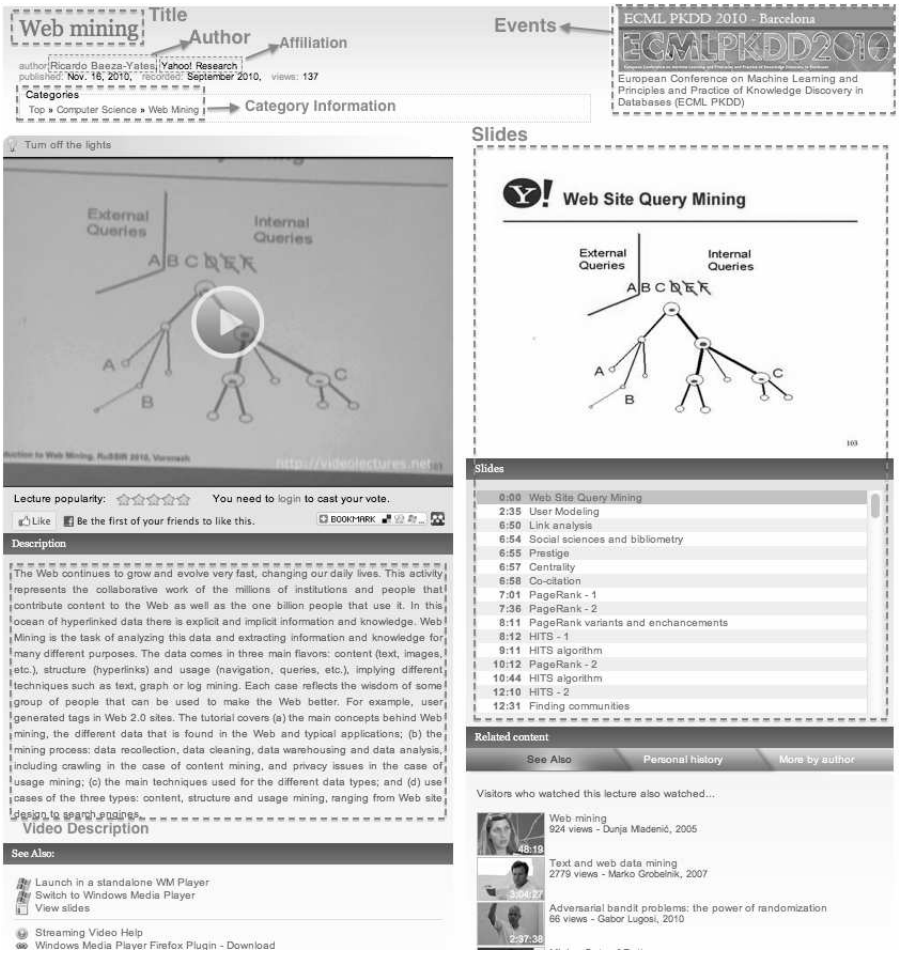


Fig. 1. Lecture video attributes

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 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

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

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 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 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

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 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)

Lightweight Approach to the Cold Start Problem in the Video Lecture Recommendation

Leo Iaquinta and Giovanni Semeraro

University of Bari "Aldo Moro", Bari, Italy
{iaquinta, semeraro}@di.uniba.it

Abstract. In this paper we present our participation as SWAPTeam at the ECML/PKDD 2011 - Discovery challenge for the task on the cold start problem focused on making recommendations for new video lectures. The main idea is to use a content-based approach because it is less sensitive to the cold start problem that is commonly associated with pure collaborative filtering recommenders. The strategy for the integration by hybridization and the scalability performance affect the developed components.

1 Introduction

In this paper we present our participation as SWAPTeam[1] at the ECML/PKDD 2011 - Discovery challenge for the task on the cold start problem focused on making recommendations for new video lectures, based on historical data from the VideoLectures.Net website.

Recommender systems (RSs) usually suggest items of interest to users by the exploitation of explicit and implicit feedbacks and preferences, usage patterns, and user or item attributes. The past behavior is supposed to be useful to make reliable predictions, thus past data is used in the training of RSs to achieve accurate prediction models. A design challenge becomes from the dynamism of the real systems because new items and new users are continuously added without a previous known behavior.

Also VideoLectures.Net exploits a RS to guide users during the access to its large multimedia repository of video lectures. Beside the editorial effort to select and classify lectures, accompanying documents, information and links, the Discovery challenge is organized in order to improve the website's current RS, inter alia, to deal with the cold start problem.

The main idea underlying our participation is to use a content-based approach because it is less sensitive to the cold start problem that is commonly associated with pure collaborative filtering recommenders. The adopted solution exploits almost all the provided data and the actual integration with VideoLectures.Net RS can be potentially performed by a hybrid approach. Moreover, the scalability performance is considered as a primary requirement and, thus, a lightweight solution is pursued.

The rest of the paper is structured as follows: Section 2 recalls some common knowledge about the cold start problem, Section 3 sketches some features of the dataset, Section 4 illustrates the proposed solution and Section 5 closes the paper with some conclusions and future work.

[1] http://www.di.uniba.it/~swap/index.php

2 Cold Start Problem

The cold start problem is commonly associated with pure collaborative filtering RSs. Particularly, the item-based collaborative filtering techniques assume that items are similar when they are similarly rated and therefore the recommendations concern items with the highest correlations according the usage evidence. As drawback, new items cannot be recommended during the cold start because they do not provide an adequate usage evidence.

The cold start problem concerns performance issues when new items (or new users) should be handled by the system. The cold start can be considered as a sub problem of the coverage one [7], indeed it measures the system coverage over a specific set of items or users. Therefore, although the prediction accuracy of a RS, especially for a collaborative filtering one, often grows with the amount of data, the coverage problem of some algorithms appears with recommendations of high quality only for a portion of the items even if the system has gathered a huge amount of data.

Focusing on cold start for items, there are various heuristics to pick out the cold items. For instance, cold items can be items with no ratings or usage evidence, or items that exist in the systems for less than a certain amount of time (e.g., a day), or items that have less than a predefined evidence amount (e.g., less than 10 ratings) [6]. The correct selection of cold items allows to process them in a different way.

The prediction about cold items requires different approaches by comparing the performance for the predictions about hot items. This may be desirable due to other considerations such as novelty and serendipity. Thus evaluating the system accuracy on cold items it may be wise to consider that there is a trade-off with the entire system accuracy [7].

3 Dataset

The main entities of the dataset are the lectures. They are described by a set of attributes and of relationships. The attributes are of various kind: for instance, *type* can have one value in a predefined set (lecture, keynote, tutorial, invited talk and so on); *views* attribute has a numeric value; *rec_date* and *pub_date* have a date value; *name* and *description* are unstructured text, usually in the language of the lecture. The relationships link the lectures with 519 context events, 8,092 authors, and 348 categories. Each of these entities has its own attributes and relationships to describe taxonomies of events and categories.

Almost all this amount of data can be exploited to obtain features for a content-based recommendation approach. The used features are briefly introduced in Section 4.2. The lectures are divided into 6,983 for the training and 1,122 for the testing as cold items.

In addition, the dataset contains records about pairs of lectures viewed together (not necessarily consecutively) with at least two distinct cookie-identified browsers. This kind of data has a collaborative flavour and it is actually the only information about the past behavior. The user identification is missing, thus any user personalization is eliminated. User queries and feedbacks are also missing.

4 Proposed Approach

4.1 Content-based Technique by Hybrid Approach

To overcome the cold start problem of the collaborative approaches, a common solution is to hybridize them with other techniques that do not suffer of the same problem [1]. For instance, a content-based approach can be used to bridge the gap from existing items to new ones: item attributes are used to infer similarities among items.

Content-based techniques also have a start-up problem because they must accumulate enough usage evidence to build a reliable classifier, but in the task on the cold start problem of the ECML/PKDD 2011 - Discovery challenge it is not an issue.

Furthermore, relative to collaborative filtering, content-based techniques are limited by the features that are explicitly associated with the items that they recommend. For instance, a content-based movie recommendation is usually based on the movie metadata, since the movie itself is opaque to the system. In the task on the cold start problem of the ECML/PKDD 2011 - Discovery challenge, this general problem is solved by the editorial effort of VideoLectures.Net to select and classify lectures. In addition, as sketched in Section 3, almost all provided data can be exploited to obtain content-based features.

The hybridization strategy can be flexible in order to apply different approaches to specific classes of items (or users) and, therefore, switch to a specific technique for the selected cold items. A switching approach [1] is a simple hybridization strategy to implement different techniques with sensitivity on the item-level without any further cost beside the cold item selection.

4.2 Steps towards Solution

The solution is obtained mainly by three steps: the data pre-processing, the model learning, and the recommendation.

Data pre-processing step starts with the loading of CSV files of the dataset by the Super CSV library[2] to obtain an in-memory object-oriented representation.
In addition, a set of Lucene[3] indexes are created to store textual metadata (title, description and slide title) in order to exploit the term frequency vectors to efficiently compute document similarities. Since the metadata is inherently multi-lingual, a single index is created for each language and textual metadata is added to the proper index according to the detected language. The language detection is performed by naive Bayesian filters that exploit language profiles learned from Wikipedia[4]. The textual metadata is also preprocessed to remove stop words and to reduce inflected words to their stem: these sub-steps are strongly language-dependent, thus specific linguistic knowledges can improve the process effectiveness.
The event names are filtered by regular expressions to introduce an event similarity metric smarter than a simple string matching.

[2] `http://supercsv.sourceforge.net/`
[3] `http://lucene.apache.org/`
[4] `http://code.google.com/p/language-detection/`

An in-memory complete representation of category taxonomy is also created to compute the category similarity as graph-based minimum path between pairs of categories.
The main output of this step is a set of 20 numeric values describing the similarities between lectures of each pair in the training set. Table 1 reports the used features: for each pair of items, they involve the languages, the frequencies of languages (Fig. 5-b), the descriptions, the recording and publication ages, the conferences, the authors and their affiliations, and the categories.

Model learning step uses Weka[5] to build a prediction model for the frequency of a pair of lectures. The available data and the lightweight goal determined the selection of a linear model for the learning problem. Thus the model output is a weighted sum of the attribute values that predicts the pair frequency. The learning process aims to obtain a regression model for the weights from the output of the data pre-processing step.
This step is quite time-consuming and it requires a lot of memory, mainly under the input constraints. Thus the output of the data pre-processing step can be controlled on exploited features and selected items.
Table 1 reports different models learned using all the available pairs: for each model, the table reports the used features with their learned weights, the regression metrics provided by Weka, and the metric values for the recommendation of cold items. Model-1 uses all the available features; Model-2 leaves out the Lucene-based similarity; Model-3 leaves out the features based on recording and publication ages; Model-4 leaves out the conferences; Model-4 leaves out the authors; Model-6 leaves out the categories. Some weights are missing for the fitness of the learning method. The learned weights of a model are stored in a configuration file, with the option to add a boost factor for each weight to easily explore the feature influences beside the learned model. Fig. 1 and Fig. 2 report the values of the evaluation metric (Mean Average R-precision - MARp) for the recommendations using Model-1 when a boost factor is changed. The boost factors can be modified also to implement a naive feedback control on recommendations without performing a complete learning step.
Fig. 3 reports the evaluation metric values for the submitted solutions when the boost factors for the learned weight in Model-1 are changed: the submitted solutions always outperform the random baseline (MARp: 0.01949).

Recommendation step uses the in-memory representation of the pre-processing step and the learned weights to predict the frequency of an old item against each selected cold item. The highest values are used to select the 30 cold items for the recommendation.
The in-memory representation and the lightweight prediction model allow to formulate a new recommendation in a reasonably short time.

The in-memory representation of the data pre-processing step is also used to create R[6] scripts to visualize the information in the dataset for an informed selection of

[5] `http://www.cs.waikato.ac.nz/ml/weka/`
[6] `http://www.r-project.org/`

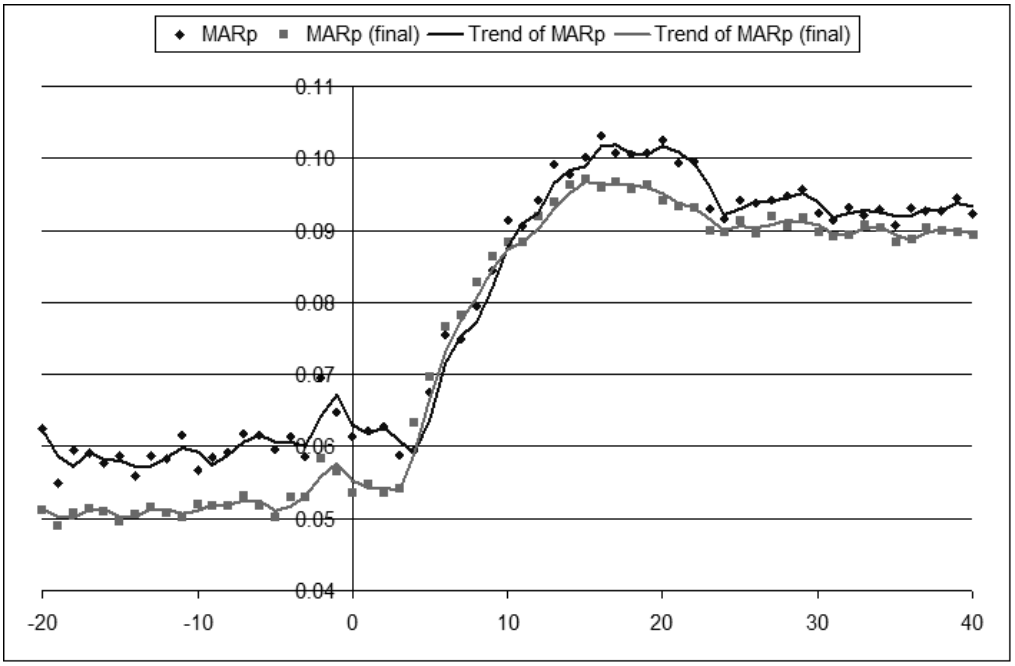


Fig. 1. Recommendation performances changing the boost factor of "categoryBest"

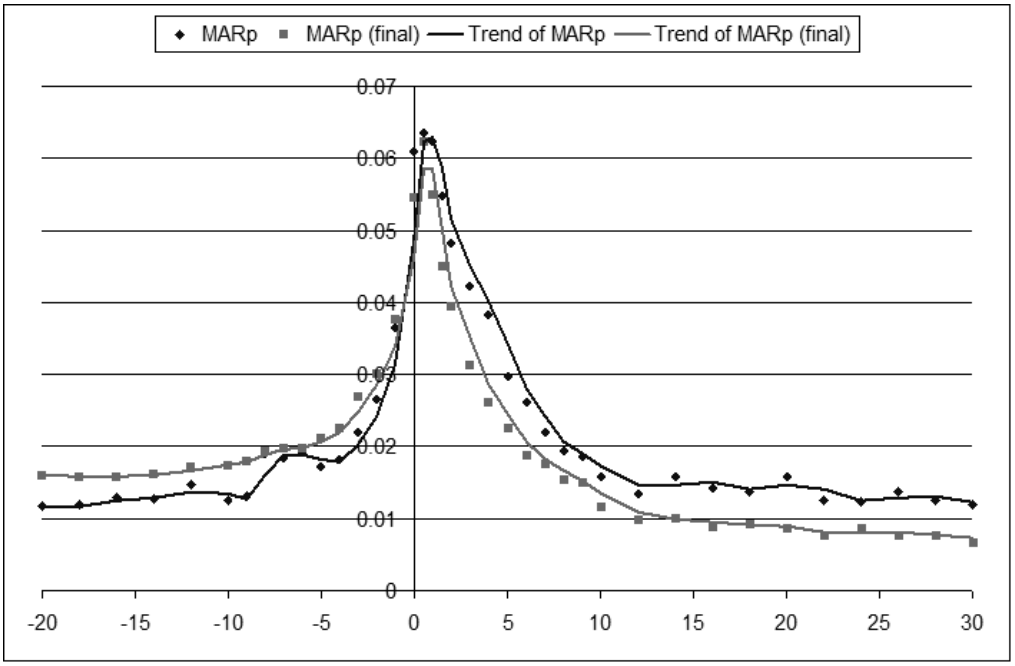


Fig. 2. Recommendation performances changing the boost factor of "deltaRecAge"

Table 1. Learned models

	Model-1		Model-2		Model-3		Model-4		Model-5		Model-6	
sameLang	x	1.2479	x	0.8097	x	1.0349	x	3.2801	x	1.7958	x	1.0363
sameDetectedLang	x	-0.3375	x	-0.2759	x		x		x		x	-0.3374
freqLang	x	2.5217	x	4.6866	x	3.4824	x	-3.8682	x		x	3.9266
freqDetectedLang	x	-0.3458	x	-0.2765	x	-0.4943	x	-0.2544	x	-0.6531	x	-0.3456
description	x	17.9226			x	16.5921	x	22.2335	x	20.9256	x	17.4239
descriptionLen	x	-0.9129	x	1.3709	x	-1.4296	x	-1.0610	x	-1.3196	x	-0.9225
deltaPubAge	x		x	-0.0557			x	0.0173	x	-0.0963	x	-0.0255
deltaRecAge	x	-0.0856	x	-0.0907			x	-0.0776	x	-0.0891	x	-0.0828
pubAgeOlder	x	0.0890	x	0.1568			x	-0.0194	x	0.1543	x	0.1252
pubAgeNewer	x	-0.0417	x	-0.0861			x		x	-0.1626	x	-0.0603
recAgeOlder	x	0.0692	x	0.0709			x	0.0401	x	0.0779	x	0.0730
recAgeNewer	x	0.0282	x	0.0111			x	0.0729	x	0.0512	x	0.0388
sameConference	x	4.4207	x	5.0438	x	4.3148			x	4.5780	x	4.5366
similarConfference	x	3.1643	x	3.2060	x	3.3740			x	3.4212	x	2.9982
atLeastOneSharedAuth	x	3.8986	x	4.0301	x	1.7389	x	4.2799			x	3.4690
sharedAuth	x		x	1.6587	x	4.2296	x				x	0.7031
sharedAffil	x	3.2381	x		x	2.5147	x	4.6232			x	3.0751
categoryBest	x	-2.9007	x	-2.9065	x	-3.6669	x	-3.1974	x	-3.0285		
categoryAvg	x	2.5258	x	2.0846	x	3.5300	x	0.6944	x	2.1317		
Correlation coefficient	0.1796		0.1736		0.1661		0.1579		0.1739		0.1719	
Mean absolute error	5.9355		5.9498		5.8786		5.9772		5.9521		5.8783	
Root mean squared error	23.2987		23.3244		23.3549		23.3868		23.323		23.3315	
Relative absolute error	96.5899		96.8226		95.6633		97.2684		96.8598		95.6583	
Root relative squared error	98.3737		98.482		98.6106		98.7453		98.4762		98.5119	
MARp	0.06220		0.05752		0.05535		0.05990		0.01295		0.06051	
MARp (final)	0.05492		0.04715		0.05306		0.05145		0.01163		0.05295	

the content-bases features. For instance, Fig. 4 shows how the views are temporally distributed considering the recording and publishing ages: the behavior is quite dissimilar for the two time scales, indeed, the oldest recorded lectures are seldom viewed as the cumulative box-plot and density function (the rightmost subgraphs) highlight, conversely the oldest published lectures have the highest density of views. Probably, the user interest for old lectures is weak even if the VideoLectures.Net kindled a lot of attention during the first months. In addition the views of lectures decrease when their recording and publishing ages decrease. Thus recent lectures need some assistance. Fig. 4 supports the idea to exploit age-based features in the model learning, although the temporal distribution of views deserves further investigation for a selective use of pairs in the learning step. Fig. 5 shows how the views of each item are distributed considering its type: the rightmost histogram shows the cumulative views for each type; the uppermost box-plot summarizes the views for each items. Fig. 5 spots how the coldness and hotness are related to the item type. Fig. 6 shows how types and languages are linked by training pairs: the circular areas are proportional to the logarithm of cumulative frequencies for the pairs of lectures viewed together. This kind of information is exploited by the "freqLang" feature.

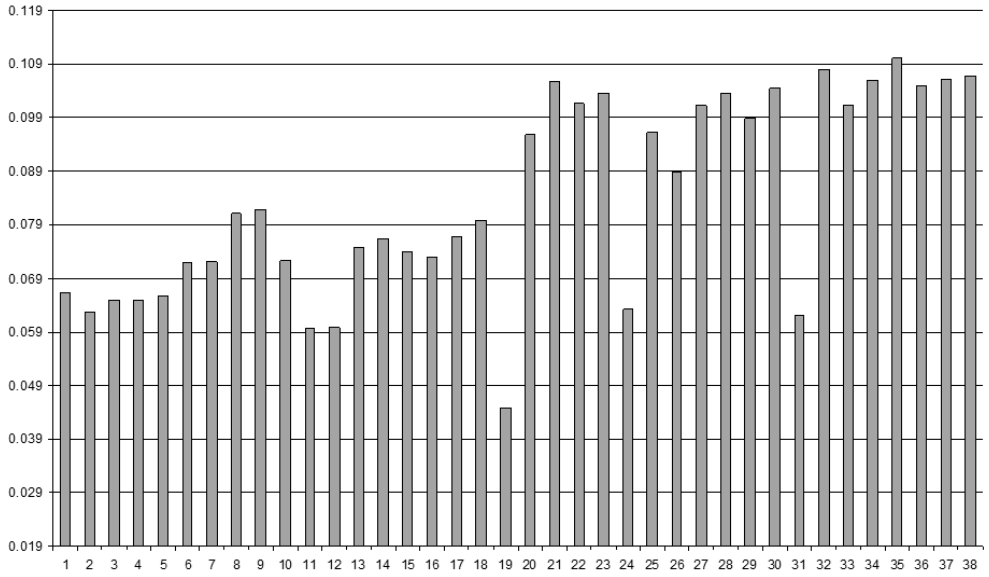


Fig. 3. Mean Average R-precision of submitted solutions

4.3 Scale Problem

With the growth of the dataset, many recommendation algorithms either slow down or require additional resources such as computation power or memory. As RSs are designed to help users to navigate in large collections of items, one of the goals of the designers of such systems is to scale up to real datasets. As such, it is often the case that algorithms trade other properties, such as accuracy or coverage, for providing rapid results for huge datasets [2]. The trade-off can be achieved by changing some parameters, such as the complexity of the model, or the sample size. For real systems it is important to measure the compromises that scalability dictates [7].

RSs are expected in many cases to provide recommendation on-line, thus it is also important to measure how fast does the system provides recommendation [3, 5]. Common measurement are the number of recommendations that the system can provide per second (the throughput of the system) and the required time for making a recommendation (the latency or response time).

The developed Java components allow to complete the recommendation task for the 5,704 lectures in almost 85 seconds on a notebook with an Intel Core 2 at 2.0 GHz as CPU and 2GB of RAM, i.e., each new recommendation about 30 cold items over the selected 1,122 ones is provided in almost 15 milliseconds. Reasonably, a production server allows to reduce further the response time for new recommendations and a cache specifically devised for the recommendations allows to increase the throughput.

Moreover, the learning step performed by Weka is the most time-consuming one and it requires a lot of memory. Although the step is designed to be performed off-line, the

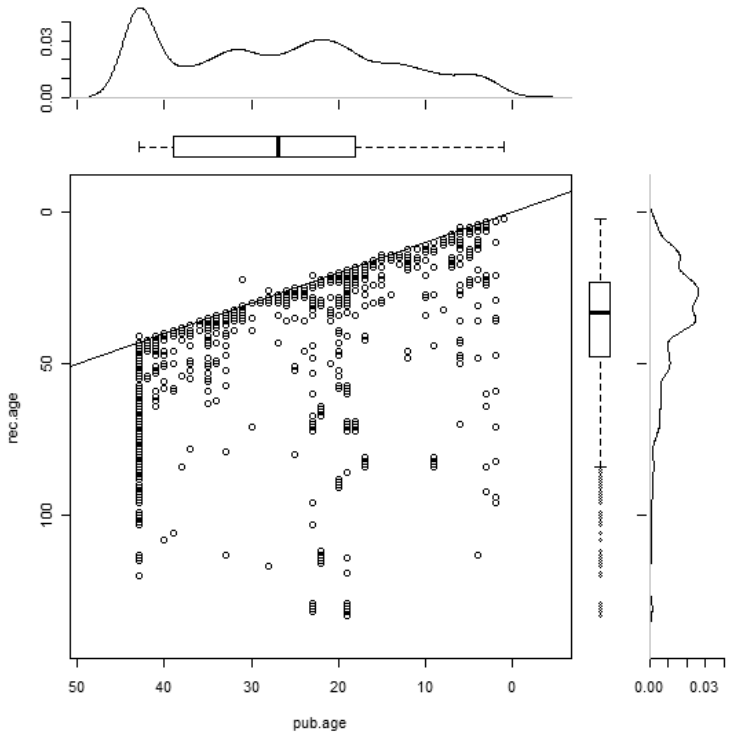


Fig. 4. Temporal distribution of views

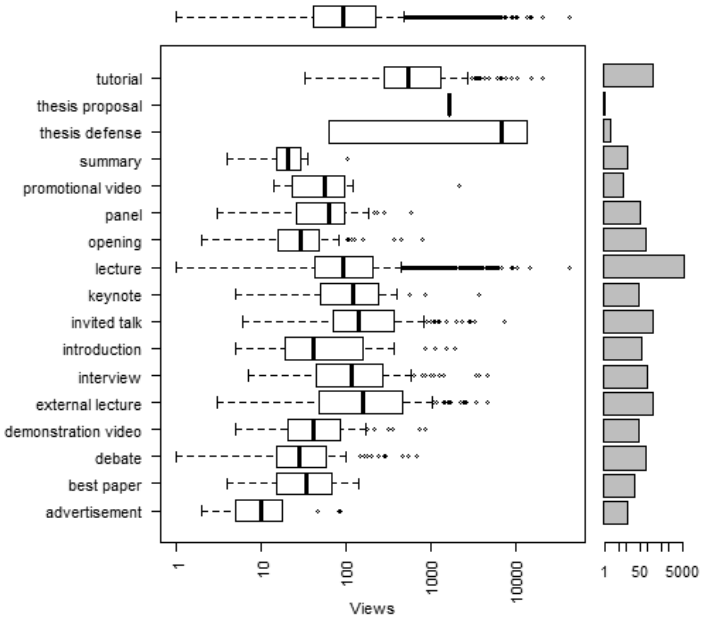


Fig. 5. Distribution of views considering item type

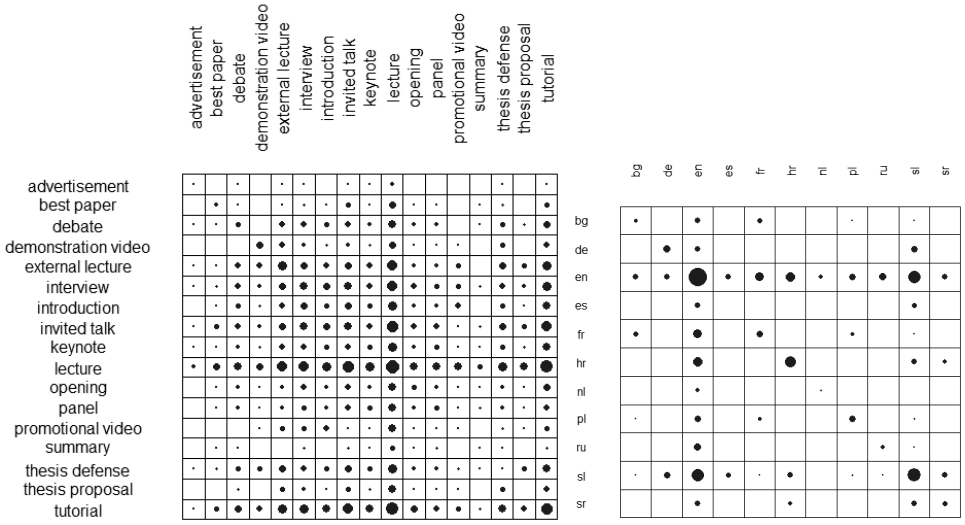


Fig. 6. Types and languages in lecture pairs

time and space requirements can be reduced by exploiting few features or less previous data.

5 Conclusions

We have described the steps to achieve the submitted solution that outperforms the random baseline. The in vitro evaluation of a solution to the cold start problem is an arduous task, since the common assumption about the reliability of past data to provide predictions is weakened. For instance, Fig. 7 shows how many of the old items used in the evaluation of submitted solutions have few associated cold items. The lack of such links becomes from the real data and it warrants the need for some strategy to deal with cold items. In additions, Fig. 7 shows that the average frequency of the considered pairs of old and cold lectures increases when the users view an increasing number of cold items for the same old item: the transition from cold to hot seems to be on the highest levels used for the evaluation metric. The evaluation levels (5, 10, 15, 20, 25, 30) are shown in Fig. 7 as grey vertical lines.

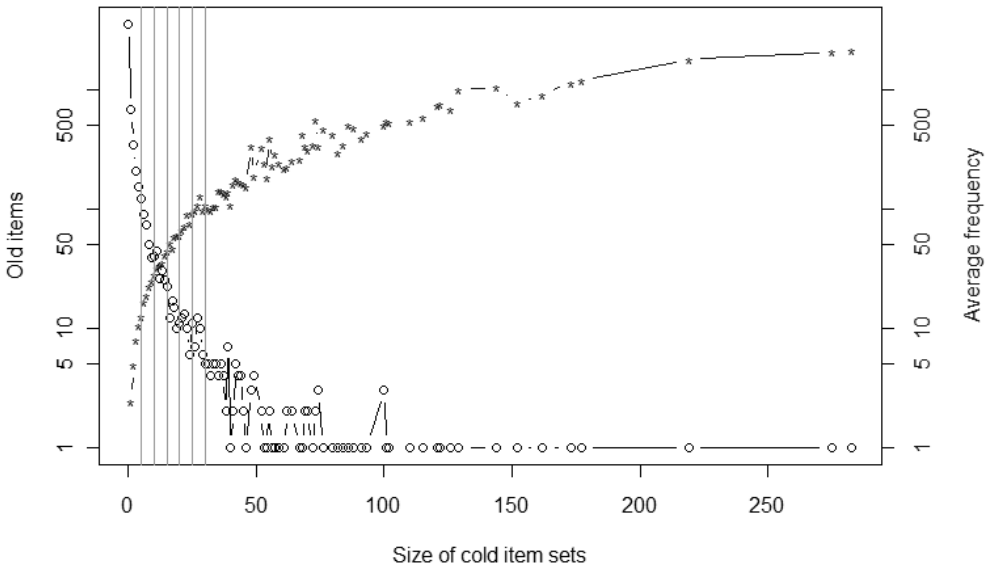


Fig. 7. Number of old items (o) and their pair average frequencies (*) on the size of cold item sets used in the evaluation

The idea of integrating a content-based approach allows to provide also serendipitous recommendations alongside classical ones [4]. Indeed the content-based item similarity

can be used to obtain a hybrid RS that exploits the "Anomalies and exceptions" approach [8] to spot potential serendipitous items as further trade-off with the entire system accuracy.

Finally, the scalability performance is considered as a primary requirement and a lightweight solution is pursued. The preliminary performance for the notebook execution is quite promising and some future directions for improving latency and throughput are sketched. Also a feasible integration strategy is depicted.

Acknowledgments. This research was partially funded by MIUR (Ministero dell' Università e della Ricerca) under the contract "Fondo per le Agevolazioni alla Ricerca", DM19410 "Laboratorio di Bioinformatica per la Biodiversità Molecolare" (2007-2011).

References

1. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12, 331–370 (2002)
2. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: Proc. of the 16th int. conf. on World Wide Web (WWW '07). pp. 271–280. ACM (2007)
3. Herlocker, J., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Information Retrieval 5, 287–310 (2002)
4. Iaquinta, L., de Gemmis, M., Lops, P., Semeraro, G., Filannino, M., Molino, P.: Introducing serendipity in a content-based recommender system. In: Xhafa, F., Herrera, F., Abraham, A., Köppen, M., Bénitez, J.M. (eds.) Proc. of the 8th int. conf. on Hybrid Intelligent Systems (HIS-2008). pp. 168–173. IEEE Computer Society (2008)
5. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th int. conf. on World Wide Web (WWW '01). pp. 285–295. ACM (2001)
6. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proc. of the 25th ACM SIGIR conf. on Research and development in information retrieval (SIGIR '02). pp. 253–260. ACM (2002)
7. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 257–297. Springer (2011)
8. Toms, E.G.: Serendipitous information retrieval. In: DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries (2000)

Recommender System Based on Random Walks and Text Retrieval Approaches

Max Chevalier, Taoufiq Dkaki, Damien Dudognon, Josiane Mothe

Institut de Recherche en Informatique de Toulouse
Université de Toulouse, France
FirstName.Surname@irit.fr

Abstract. This paper presents the approaches IRIT developed for the VLNetChallenge regarding recommender systems in the context of video lectures. The first task aims at recommending newly acquired lectures after viewing an "old" lecture. We use random walk algorithms based on a graph composed of author, category, event, and lecture nodes and associated relationships. The second task aims at recommending 10 lectures from three lectures extracted from a sequence of lectures. We use the categories associated to lectures in addition to the lecture pairs (lectures viewed in a same session).

1 Introduction

IRIT participated to the two tasks of the VLNetChallenge.

Regarding the cold start task, which aims at recommending newly acquired lectures after viewing an "old" lecture, we first built a graph from the data collection. Graph nodes are lectures and associated meta-data (authors, events and categories). Graph links correspond to the various types of relationships (links between lectures, between events and between categories as well as cross-type links). Relationships were weighted differently according to the nature of the links. The resulting graph was used in random walk algorithms. The best results on the test collection have been obtained when the graph weights are significantly more important for the lecture pairs and the authors-lectures relationships than for the remaining relationships.

Regarding the pooling lecture task, we first considered the lecture contents only; this method showed poor results. We then consider the lecture categories. Since many lectures are not linked to categories, we first defined a way to deal with this problem. Then, we use the frequency of lecture visits, lecture pairs and the categories they belong to.

2 Data preparation

To begin with, we uploaded the CSV data provided to the participants in a PostgreSQL database [15]. For each lecture, we extracted the categories, events and authors associated with it.

We also indexed lectures using the Solr search engine [14]. We used as content the *name*, *description* and *slide_titles* fields of each lecture. Indexing is based on a "bag of words" approach. To build the Solr index, the stopwords were not removed and we did not use any stemming heuristic similar to the Porter Stemmer [8]. Avoiding pre-processing steps allows us

to store all the lectures in the same index, regardless of their language. The retrieval model used in Solr combines Boolean Model [7] and Vector Space Model [11]. The documents are first selected by Boolean Model and then are scored using Vector Space Model. The scoring function implemented in Solr is derived from the VSM score, based on the Cosine similarity [10].

Solr was used in the two tasks. In the cold start task, Solr was used to build two matrices that reflect the lecture similarities based on content. For the first one, we used MoreLikeThis from Solr to calculate the similarities between each lecture pairs. For a given document, the MoreLikeThis module generates a query based on the representative terms of the document. These terms are selected depending on several parameters which are: their length, their frequency in the document and their frequency in the overall collection. The second matrix was built differently: for each lecture, we calculate its similarities with all the other lectures, considering its title as a query; lectures were favored if recent.

In the pooled sequences task, Solr was used to retrieve the most similar lectures from a given lecture.

3 Cold start task

The cold start task aims at predicting "which of the newly acquired lectures at the site should be recommended after viewing some of the 'older' lectures" [12].

To complete this task, we first built a graph from the data in which nodes and relationships are typed. In addition we weighted some of the relationships. Then we applied two random-walk models to compute document similarities and predict which new lectures should be recommended. Section 3.1 explains the way the graph is built and section 3.2 explains the way it is used.

3.1 Graph generation

From the data, we built a graph $G=\{N,R\}$ where N is a set of nodes and R a set of relationships between nodes.

The set of nodes N is defined as: $N=\{A, C, E, L\}$ where:

- A is a set of author nodes,
- C a set of category nodes,
- E a set of event nodes, and
- L is a set of lecture nodes.

The set of relationships R is defined as:

$R=\{CR, ER_{ei,ej}, AR_{li,aj}, DR_{li,cj}, TR_{li,ej}, LR_{li,lj}\}$ where:

- CR is a relationship defined between two categories.

$CR(c_i,c_j)$ = 1 if categories c_i and c_j have a hierarchical relationship in the database;

= 0 otherwise.

- ER is a relationship between two events. As for CR, $ER(e_i,e_j)$ is either 0 or 1, based on the hierarchical relationship defined between events e_i and e_j using parent_id attribute.

- AR is a relationship between a lecture and an author.
- DR is a relationship between a lecture and a category.
- PR is a relationship between a lecture and an event.

For those three relationships, when the items are associated in the data set, the relationship is weighted 1; 0 otherwise.

- LR is a relationship between two lectures. We defined two types of LR relationships. They can be either content based similarities or deduced from pairs of lectures. Lecture pairs were provided to participants; the deduced LR_P relationships were weighted considering the frequency of each pair and the number of views associated to its related lectures. Lecture similarities were calculated as described in section 2 and conduced to weighted LR_S relationships. LR_P and LR_S relationships were fused considering a linear combination, such as:

$LR(l_i, l_j) = \beta * LR_{P(l_i,l_j)} + \gamma * LR\_S(l_i, l_j)$

where l_i and l_j are two lectures. In the experiments, β=1.5 and γ=0.05. These values have been obtained through manual tuning.

Finally, each type of relationships receives a relative weight. For example, $AR(l_i, a_j)$ receives a relative weight of 3 between l_i and a_j if the lecture and the author are linked. Figure 1 depicts the various types of relationships that link nodes.

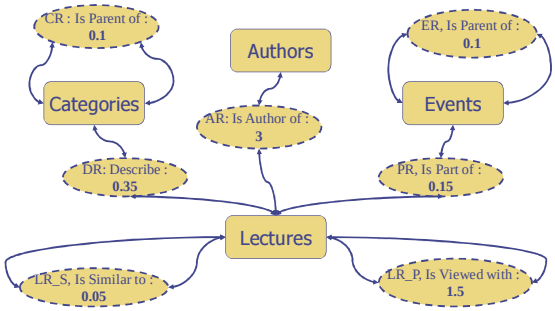


Fig. 1. Nodes and relationships between nodes.

3.2 Random walks

We considered two random walk algorithms: Katz [6] and Random-Forest based Algorithm [5] that consider route accessibility and relative forest accessibility [4]. More details on these methods are presented by Fouss *et al.* [5]. In this latter paper, more methods are also discussed.

Katz. The method proposed by Katz to compute similarities takes into account both direct and indirect links between items [6]. The similarity matrix is defined as:

$$K = \alpha A + \alpha^2 A^2 + \cdots + \alpha^n A^n + \cdots = (I - \alpha A)^{-1} - I \quad (1)$$

where A is the adjacency matrix, I the identity and α constant.
A is the adjacency matrix generated from the complete graph (rows and columns of the matrix are the nodes of the various types) and thus represents direct links between the graph's nodes. A^n represents the indirect links through paths of length n. Both direct and indirect links are taken into account but a coefficient of attenuation is used: α^n represents the attenuation in im-

portance of the links of length *n*, *K* exists provided that the attenuation coefficient α is less than the inverse of the spectral radius of *A*. In our experiments, we use α =0.05. This value should have been tuned; but we did not for time reasons.

Random-forest based algorithm (RFA). In RFA, the similarity matrix *S* between the nodes of a graph G is based on relative forest accessibility. Let $\mathcal{F}(G) = \mathcal{F}$ be the set of all spanning forests of graph *G*. A spanning forest is any subgraph of *G* that is cycle free and includes every vertex of *G*. For any two nodes *i* and *j* of *G*, $\mathcal{F}_{ij}$ denote the subset of $\mathcal{F}$ where *i* and *j* belong to the same tree. The relative forest accessibility of *i* and *j* is defined as $s_{ij} = \varepsilon(\mathcal{F}_{ij})/\varepsilon(\mathcal{F})$. ε is the weight function defined in [4]. For unweighted graphs $\varepsilon(\mathcal{F}_{ij})/\varepsilon(\mathcal{F}) = |\mathcal{F}_{ij}|/|\mathcal{F}|$

[4] demonstrates $(I + L)^{-1}$ exists for any undirected weighted graphs and that :

$$S = (I + L)^{-1} \qquad (2)$$

where *L* is the laplacian matrix from the adjacency matrix *A* generated from the complete graph *G* (see section 3.1).

RFA which can be seen as a rough Laplacian regularization is closely related to the similarity measure associated to the pseudo-inverse of graphs Laplacian L^+(see [4] for more details). L^+ is a valid kernel that preserves the Euclidian commute time distance in graphs. We did not experiment the similarity measure based on L^+ in the context of VLNetChallenge for lack of time to solve a technical problem.

3.3 Implementation issues

All experiments were conducted on Linux computers with a 2.9 GHz Intel Core2 Duo processor P9700 and 6 GB of RAM.

The graphs we handled in the context of VLNetChallenge contain around 15 000 nodes. The approaches we explored are then based on inverting matrices ($\mathcal{O}(n^3)$) of size 15 000 x 15 000. Our attempt to use Scilab [16] (with memory stack set to the maximum) was unfruitful and ended with a stack overflow error after more than 20 hours of running time. After shifting to atlas [17] the Automatically Tuned Linear Algebra Software, the running time was about 20 minutes.

3.4 Results

When considering the preliminary results on the training collection (based on 20% of the final collection), our method obtained from 0.1434 to 0.22465, depending both on the random walk method used and on the weight used for the relationships. The best results have been obtained for RFA using the weights presented in bold font in Figure 1. These weights have been obtained through a rough manual-tuning that used the entire training collection.

When considering the final collection, our method is ranked 9 over 58 submissions without nil results or errors. We obtained a score of 0.24044 while the best result is 0.35857. Interesting enough, when considering the approaches better than ours, we can see that the results decrease from the preliminary results to the final results. One hypothesis could be that those approaches over learnt on test data.

4 Pooled sequences

In this task participants "are asked to recommend a ranked list of ten lectures that should be recommended after viewing a set of three lectures" [13].

To complete this task, we followed an empirical approach according to our knowledge mainly acquired in Information Retrieval field. This knowledge has been transposed and adapted to recommender systems. We tried two approaches that are related to the work we presented in [2]: one was based on **lecture content only**; the second one considered the **categories** associated to lectures and **lecture pair frequency**.

4.1 Content-based approach

In this approach, we considered the lecture content only. We used Solr search engine [14] as explained in section 2. For each lecture of a given triplet, we search for the 50 most similar documents. Then we fused the three retrieved document list using CombSum function [8] that consists in the sum of the document's individual scores.

When applied to the training collection, the results were slightly above 0.04. Indeed when analyzing the learning data set, we identify that users read lectures related to various topics to complete their knowledge. This variety of topics cannot be captured with a standard content similarity-based measure. For this reason, we did not continue with this content-only approach. Thanks to the work done in the cold-start task, we decided to particularly study lecture pair frequency (importance of LR_P in section 3.1) and categories.

4.2 Category-based approach

Rather than considering the lecture content only, we concentrated on the categories of the lectures. The first issue to solve was the fact that many lectures were not associated with any category. For those lectures, we first associated them with a category considering the hierarchy of events. Once the lectures are associated with a category, we then consider the lectures that have been visited with one of the lectures of the target triplet within close categories in the category hierarchy.

Association of categories to lectures. Some of the lectures are not associated with any category; for those lectures, we applied two algorithms. First for any lecture that is not in categories_lectures, we browsed the lecture-event hierarchy using a bottom up approach and associated the current lecture to the category or categories associated to the closest event (considering the hierarchy). When such a parent does not exist, we associated the category (or categories) of the most similar lectures or events, based on its content or description.

Frequency of lecture pairs. For each lecture of the current triplet, we search for the 100 most visited lectures with the current lecture. We then calculate the lecture score (3). The score of the retrieved lecture l_i is computed as its frequency times the distance between categories. Indeed, this distance between categories allows the system to identify recommendations close in sibling categories. In that way, we emphasize the selection of information in close categories

in order to simulate the user behavior according to what we have extracted from the training data set analysis.

$$\text{Score}(l_j) = \text{Frequency}(l_j) * \text{similarity}(\text{category}(l_j), \text{category}(l_i)) \quad (3)$$

When a lecture has more than one category, we use the most general category. This treatment is repeated for the three lectures of the triplet and the three lists are fused using CombSum. The distance between categories is inspired from our previous work detailed in [1].

We then ranked the retrieved lectures by decreasing scores. The recommendations are the top 10 lectures. Using this method, it occurs that we obtained less than 10 recommendations. In those cases, we then add lectures to the recommended list.

Completing the recommended list. When less than 10 lectures are recommended using the previous method, we complete the list by considering the lecture content rather than the lecture visits. For each lecture, we search for the 10 most similar lectures. For each lecture, we search for the 100 lectures the **most visited** with the current lecture and calculate the score of the lectures using the same method as previously. When this process fails to complete the list, it is completed with the lectures the most visited thanks to the frequency of lecture views.

4.3 Results

Considering the training set, using our method, we obtained from 0.04453 to 0.18725 (depending on the approach used).

Regarding the complete set, we are ranked 12th with the score of 0.18943. The best score being 0.62415.

The results we obtained show that the visits on lectures has a great importance; more than the content itself.

5 Conclusion

In this paper, we describe the methods we developed for the two tasks defined in VLNet-Challenge. With regard to the cold start task, our method was not over trained. We tried various values for the different parameters. A more systematic tuning could help improving the results. With regard to the pooled sequence task, we identified that content only approach is not sufficient. Furthermore, we think that categories could have been used more. For example, for a given triplet, we could have kept only those retrieved lectures that share a category with any lecture of the triplet.

In the two tasks, we also identified the importance of the frequency of lecture pairs. As a conclusion, we expect that combining various dimensions in recommender systems can improve recommendation quality.

References

1. G. Cabanac, M. Chevalier, C. Chrisment, and C. Julien, An Original Usage-based Metrics for Building a Unified View of Corporate Documents, International Conference on Database and Expert Systems Applications (DEXA), Springer-Verlag, LNCS 4653, pp. 202-212, 2007.
2. L. Candillier, M. Chevalier, D. Dudognon, and J. Mothe, Diversity in Recommender Systems: bridging the gap between users and systems, Centrics, to appear 2011.
3. P. Y. Chebotarev and E. V. Shamis. The matrix-forest theorem and measuring relations in small social groups. Automation and Remote Control, Vol. 58, N°9, pp. 1505–1514, 1997.
4. P. Y. Chebotarev and E. V. Shamis, On proximity measures for graph vertices, Automation and Remote Control, Vol. 59, N°10, pp. 1443–1459, 1998.
5. F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering, 2006.
6. L. Katz. A new status index derived from sociometric analysis. Psychmetrika, 18(1), pp 39–43, 1953.
7. F. W. Lancaster, E. G. Fayen, Information Retrieval On-Line, Melville Publishing Co., Los Angeles, California, 1973.
8. J. H. Lee. Analyses of multiple evidence combination, Proceeding of SIGIR'97, pp. 267–276, 1997.
9. M. F. Porter, An algorithm for suffix stripping, Program, pp. 130–137, 1980.
10. G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.
11. G. Salton, A. Wong, and C. Yang, A vector space model for automatic indexing, Commu- nications of the ACM, 18(11), pp. 613–620, 1975.
12. http://tunedit.org/challenge/VLNetChallenge/task_1, June 2011.
13. http://tunedit.org/challenge/VLNetChallenge/task_2, June 2011.
14. http://wiki.apache.org/solr/, June 2011.
15. http://www.postgresql.org/, 2011.
16. http://www.scilab.org, August 2011.
17. http://math-atlas.sourceforge.net, August 2011

Joint Features Regression for Cold-Start Recommendation on VideoLectures.Net

Gokhan Capan, Ozgur Yilmazel

Department of Computer Engineering
Anadolu University, Eskişehir, Turkey
{gcapan, oyilmazel}@anadolu.edu.tr

Abstract. Recommender systems are popular information filtering systems used in various domains. Cold-start problem is a key challenge in a recommender system. In *new-item/existing-user* case of the cold-start problem, which is recommendation of a recently-arrived item to a user with historical data, finding links between existing items with recently-arrived items is critical. Using VideoLectures.net Cold-Start Recommendation Challenge data, this paper includes a linear regression model to predict future co-viewing count between an existing item and a recently-arrived, not-yet-viewed item.

1 Introduction

A recommender system produces user-specific subsets of a global item set, in which users are expected to be interested. These recommendations are computed by predicting user-item scores that are not known yet, generally based on:

- Implicit or explicit ratings of similar users on items that user has not rated yet.
- Content similarity between items that user has rated high and has not rated yet.

According to the algorithms listed above, a recommender system may either use a collaborative, content based, or hybrid approach to produce recommendations.

Although collaborative filtering systems are very successful when there are sufficient historical user-item data available, they cannot produce recommendations in any of the cold-start cases, which are recommending new items (an item that nobody has rated yet) to users, or recommending items to new users (a user with no historical ratings data).

Recommending existing items is not in this paper's scope, we will focus on new-item cases.

Finding *existing item-new item* links using *existing item-existing item* links provide scores to be used to discover next high-rated item after high-rating an existing item, which is an important signal for recommendation. Our method represents two items as one vector of a joint feature set, which is constructed by computing relationships between some of their content features (title, categories, authors, … in videolectures.net domain). Then we apply linear regression to predict which item would be consumed after current item most probably. This is a ranked list of candidate successor items for each existing item.

The rest of the paper is organized as follows: Section 2 is the related work, Section 3 describes the methods we have used and experiments we have done, Section 4 concludes the paper.

2 Related Work

There have been many collaborative filtering algorithms studied. These algortihms produce recommendations using:

- Neighborhood based methods [1, 2, 3]
- Latent Factor Models [4, 5, 6]
- Matrix Factorization based methods [7]

Neighborhood methods are the traditional collaborative filtering models, which used to be the dominant collaborative filtering method before matrix factorization techniques. Typically, a neighborhood based collaborative filtering algorithm finds nearest neighbors of items or users, according to historical rating data [8]. Using the neighborhood, the algorithm tries to predict unknown user-item ratings.

Latent factor models are based on representing both users and items in the same feature space, latent factors. Latent Semantic Models for Collaborative Filtering [9] is an example.

Matrix factorization for performing collaborative filtering (may be included in latent factor models) is based on factorizing ratings data, which is a user-item matrix. Singular Value Decomposition based recommenders are the popular factorization based methods [7].

The pure collaborative methods for recommender systems are not able to solve cold-start problems for new-item case. Generally, hybrid models are used to solve cold-start problems [10, 11]. There are some other approaches that fill the user-item matrix by generating ratings (with a bot, for example) [12].

Menon and Elkan's method for Dyadic Prediction (Recommendation and link prediction are examples of dyadic prediction) [13] introduces a log-linear model for discovering latent factors, where a dyad may be a user-item pair (recommendation), or either item-item or user-user pairs (link predicition). Their approach takes side information (different from just unique identifiers) into account, thus providing a solution to cold start problem.

Chu and Park's bilinear regression method for recommendation on dynamic content [14] also benefits from the static features of users (gender, for example) and items (bag of words, category, ...). Their method let them recommend very recent items to users, showing that it may be considered as a solution to cold start problem for the new item case.

Again, Park and Chu's pairwise preference regression method specifically addresses the cold start problem [15]. Their method represents a joint feature space for user/item pairs via outer products.

Park and Chu's pairwise preference regression method is the inspiring method for our solution for predicting links between existing items and new items. However, there are differences. Our solution focuses on constructing a joint feature space for item-item pairs. This feature space is not bilinear. Instead, it is constructed using relationships between two items on several content features. A transformed feature may be either numerical or categorical. For example, cosine similarity between titles of two items is a candidate numerical joint feature (title relationship) for the final dataset, whereas the language relationship (the language code if they are in the same language) is a categorical joint feature.

3 Methods and Experimental Study

3.1 VideoLectures.Net Data

VideoLectures.Net[1] is a repository for video lectures from scientists in various events. The algorithm is applied to the dataset provided by VideoLectures.Net Recommendation Challenge [16]. Listed below are some properties of the dataset:

1. **authors**: Contains data on authors registered on VideoLectures.Net
   attributes: id, name, gender, email, homepage
2. **author_lectures**: Information on which author authored what lecture (There is a many-to-many relationship between authors and lectures)
   attributes: author_id, lecture_id
3. **categories**: Information on categories in scientific taxonomy.
   attributes: id, name, parent_id
4. **categories_lectures**: Information on what lecture is categorized under which category. (There is a many-to-many relationship between categories and lectures
   attributes: category_id, lecture_id
5. **lectures_train**: contains a subset of lectures with publication date prior to 1.7.2009
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description, slide_titles, views
6. **lectures_test**: contains a subset of lectures published after 1.7.2009
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description, slide_titlesll
7. **pairs**: records about a pair of lectures viewed together with at least two distinct cookie identified browsers.
   attributes: lecture1_id, lecture2_id, frequency
8. **events**: contains information on events, which are basically a set of lectures grouped together.
   attributes: id, type, language, parent_id, rec_date, pub_date, name, description
9. **task1_query**: Contains lecture ids from the subset of lectures_train, for which a ranked list of 30 recommended lectures from lectrues_test is expected.
   attributes: id

3.2 Methods and Experiments

To estimate the model that predicts co-viewing counts of *existing video-new video* pairs, we first transform the features of video pairs into one joint feature space. We find relationships between two videos on several content features, and use these relationships as resulting features. Attributes of the transformed data are:

- **type**: If two videos are in the same type (lecture, keynote, ...) the common type, 0 otherwise. This feature is categorical with 16 distinct values
- **language**: If two videos are in the same language the language code, 0 otherwise. This feature is categorical with 10 distinct values.

[1]http://www.videolectures.net

- **parent**: The parent property is hierarchical, that is, parents of videos (the events) also have parents. We have detected all parents of two lectures to the deepest level.
  The value of this feature is simply the jaccard similarity of the resulting sets.
- **title**: After the indexing process, total idf (inverse document frequency) of the common terms is the value for this feature. The reason we chose inverse document frequencies over term frequencies (or simply 0 or 1) is to make significant words more discriminative.
- **categories**: In the dataset, categories are also defined in a hierarchy. We have created category indexes for lectures to the deepest level. Because the top categories are very common for all videos, we applied the same strategy as we did while computing the value for the title feature. The total idf of the common categories of two lectures is the resulting value.
- **authors**: To increase the effect of 'same author's different lectures' if she has fewer lectures in the web site, we applied the total idf strategy again, for authors feature.
- **description**: We computed the value in the same way as we did for the title.
- **co-viewing count**: Score is the target value of the predictors defined so far, which is frequency of lecture pairs viewed together.

Each example in the transformed data representing a video pair contains a set of features, and a target variable (co-viewing count). The task of estimating a function to predict future co-viewing counts is a supervised learning, specifically regression problem. We have used linear regression with the simplest estimator; ordinary least squares. Linear regression with an intercept term estimates a function in the following format:

$$y = ß_0 + ß_1 x_1 + ß_2 x_2 + \ldots + ß_k x_k \quad (1)$$

where y is the target variable to be predicted, x_i's are regressor variables (features), and $ß_i$'s are the parameters to be learned. We want to estimate the parameters so that the resulting function minimizes the error, which is the difference between the observed (actual) and estimated y values. Ordinary least squares is the approach of minimizing the sum of squared errors. Finally, one can apply the estimated regression function to an arbitrary video pair to predict the number of times the videos will be watched together. Before applying the function, she need to transform the video-pair into our joint features space.

We used R programming language [17] to estimate the linear model. Because language and type attributes are categorical and have 10 and 16 distinct values, respectively; the resulting size of dimensions is 31, while there are 363880 training example.

To measure the relative importance of regressor variables on predicting the co-viewing count of two videos, we have used the approach Grömping has introduced [18]. Using the metric *last*, we have compared each regressor variable's contribution to accuracy when all other regressors are available. The top five regressor variables are shown in Table 1.

Table 1. Top 5 relatively important regressors

Feature	Importance
title	0.0065
categories	0.0043
parent	0.0040
description	0.0025
authors	0.0022

Finally, we have assigned values of common term frequencies to features where we have used inverse document frequencies previously, and run the same algorithm. The final evaluation results will show that using term frequencies decreases the prediction accuracy.

3.3 Evaluation

To pre-evaluate the model we have estimated, we applied 3-fold cross-validation to data. The cross validated standard error of estimate, which is the square root of the mean of MSE's of 3 folds, is computed as 23.4.

We have also analyzed the residuals, which is the difference between the actual value and estimated value of an observation, when 10% of the observations were used to test the model we have estimated from the remaining 90%. The quartiles of residuals may give an idea about its distribution. Table 2 shows the quartiles with minimum and maximum values of residuals we have computed, Figure 1 is a boxplot of quartiles with a range of 20. The quartiles show that 50% of residuals are between -4.14 and -0.27.

Table 2. Quartiles of residuals

Minimum	1st quartile	2nd quartile (median)	3rd quartile	Maximum
-151	-4.14	-2.22	-0.27	3170

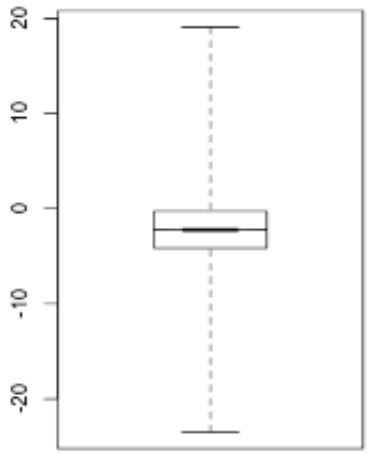


Fig. 1. Boxplot of quartiles of residuals

Goal of VideoLectures.Net Recommender System Challenge Task 1 is finding a ranked list of potential next lectures for each lecture in *task1_query* data, by retrieving a ranked sublist from the test set of lectures. The challenge can be considered as an information retrieval problem, and they have defined an R-precision variants of precision at K and MAP, which are standard evaluation measures used in information retrieval [19].

To find a ranked list of recommended lectures for a given lecture in *task1_query* data, we have paired the lecture with each possible test lecture from *lectures_test* data, computed predicted values, ranked them according to the scores we have computed, and submitted the top ranked 30 candidate lectures. The evaluation score is computed as 0.2492. The experiments also show that choosing term frequencies instead of inverse document frequencies decreases the predicting performance. In this case, the evaluation score is computed as 0.2266. The final evaluation score, 0.2492, can be considered high, ranked 7th among 1656 submitted solutions from 62 active teams of 303 registered teams with 346 members.

4 Conclusion

In this paper, we have described our solution to predict item-to-item link scores (co-viewing counts, in this case) to solve cold-start problem in recommender systems using VideoLectures.Net Recommender System Challenge data. We have used ordinary least squares linear regression to predict scores. The results show that the method of defining joint features and applying regression on transformed data provides us simple and accurate results in relatively small dimensions.

However, we have only tried ordinary least squares linear regression, which may not be the best model for the problem; as a future work other regression methods, especially the non-linear models may be applied to the problem. In addition, more features may be defined, and an efficient feature selection method may be applied to data. We also left this process as a future work.

References

1. Shardanand, U. and Maes, P.: Social Information Filtering: Algorithms of automating “word of mouth”, Proceedings of the SIGCHI Conference on Human Factors in Computer Systems. (1995)
2. Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J.: Item-based Collaborative Filtering Recommender Algorithms, WWW, pp. 285-295 (2001)
3. Herlocker, J. L., Konstan, J. A., Borhcers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering, Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (2002)
4. Billsus, D., Pazzani, M. J.: Learning Collaborative Information Filters, Proceedings of the 15th International Conference on Machine Learning, pp. 46-54. (1998)
5. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm, Information Retrieval, vol. 4 issue 2, pp. 133-151 (2001)
6. Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J.: Application of dimensionality reduction in recommender systems – a case study, ACM WebKDD Workshop. (2000)
7. Sarwar, B. M., Karypis, G., Konstan, J., Riedl, J.: Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. Fifth International Conference on Computer and Information Technology. (2002)
8. Koren, Y. and Bell, R.: Advances in Collaborative Filtering, Recommender Systems Handbook, pp. 145-186. Springer (2011)
9. Hoffman, T.: Latent Semantic Models for Collaborative Filtering. ACM Transactions on Information Systems, vol. 22 issue 1, pp. 89-115. ACM (2004)
10. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M.: Combining content-based and collaborative filters in an online newspaper. ACM SIGIR Workshop on Recommender Systems. (1999)
11. Melville, P., Mooney, R., Nagarajan, R.: Content-boosted Collaborative Filtering for Improved Recommendations. Proc. of the National Conf. on Artificial Intelligence, pp. 187-192. AAAI (2002)
12. Park, S. T., Pennock, D. M., Madani, O., Good, N., and DeCoste, D.: Naïve filterbots for robust cold-start recommendations., Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2006)
13. Menon, A. K., Elkan, C.: Dyadic Prediction Using a Latent Feature Log-Linear Model. CoRR (2010)
14. Chu, W., Park, S.: Personalized Recommendation on Dynamic Content Using Predictive Bilinear Models. Proceedings of the 18th International Conference of World Wide Web. ACM (2009)
15. Chu, W., Park, S.: Pairwise Preference Regression for Cold-Start Recommendation. Proceedings of the Third ACM Conference on Recommender Systems. ACM (2009)
16. 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
17. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, http://www.R-project.org. (2011)
18. Grömping, U: Relative Importance for Linear Regression in R: The Package relaimpo. Journal of Statistical Software, vol. 17, pp. 1-27. (2006)
19. Buckley, C., Voorhees, E. M.: Evaluating evaluation measure stability. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM (2000)