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 the content-bases features. For instance, Fig. 4 shows how the views are temporally distributed

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

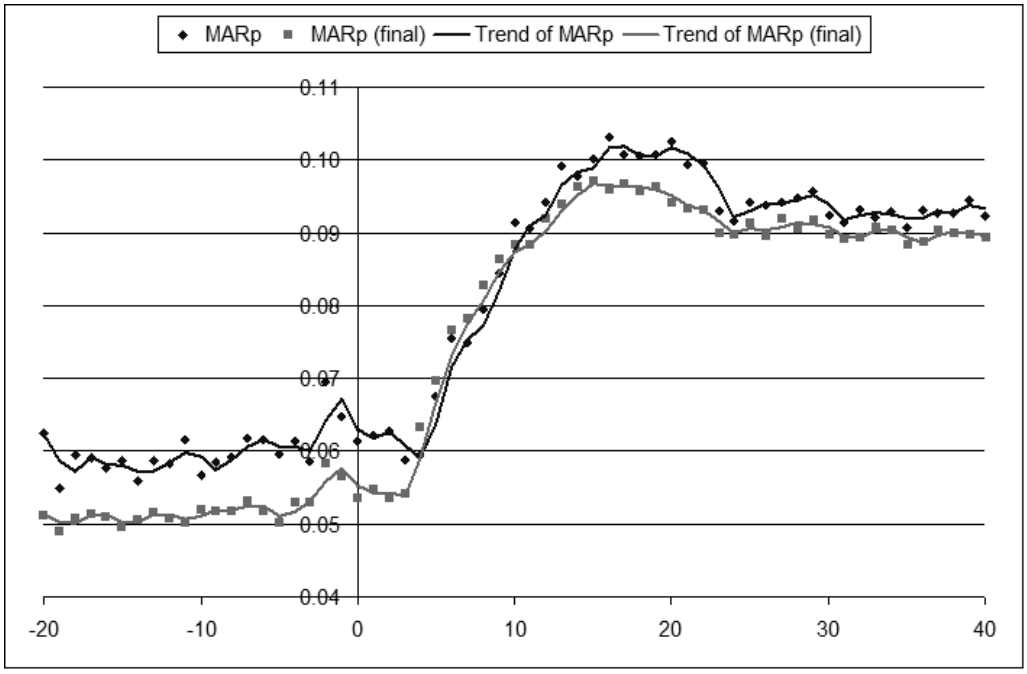

Fig. 1. Recommendation performances changing the boost factor of "categoryBest"

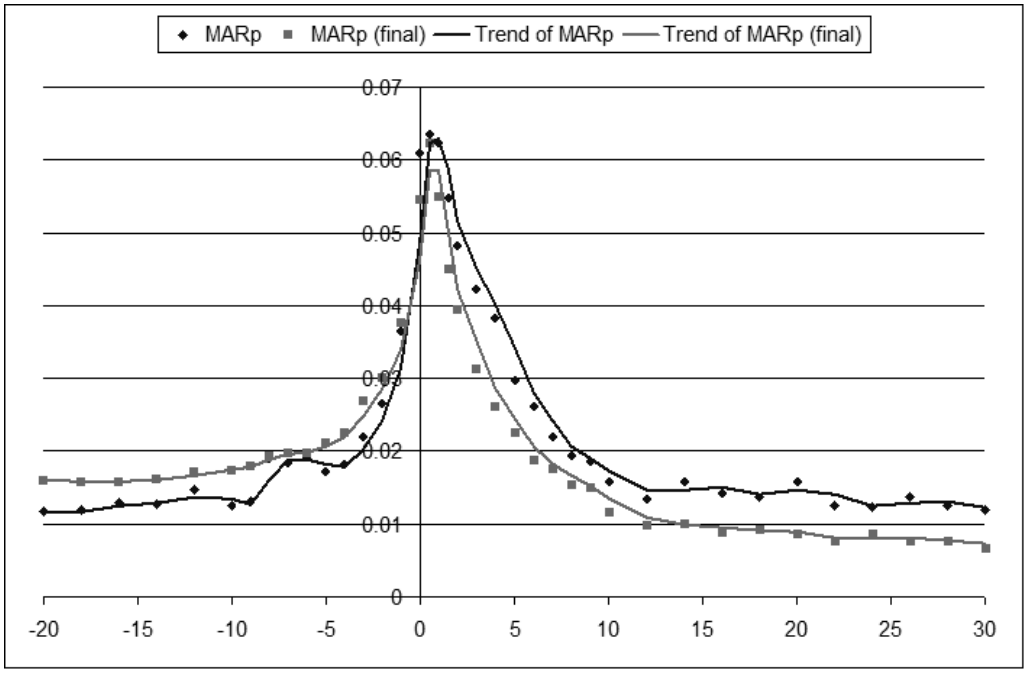

Fig. 2. Recommendation performances changing the boost factor of "deltaRecAge"

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	

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.

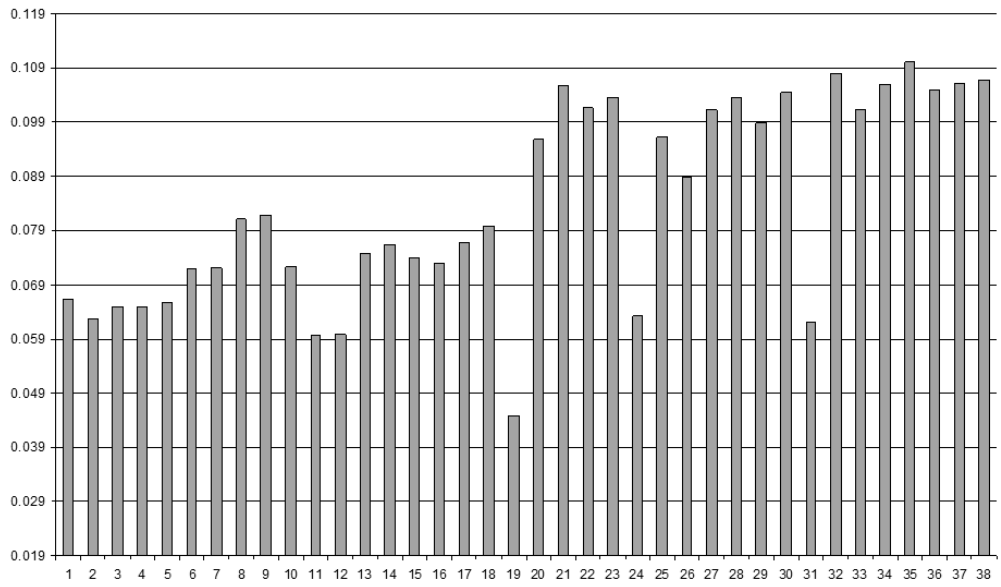

Fig. 3. Mean Average R-precision of submitted solutions

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

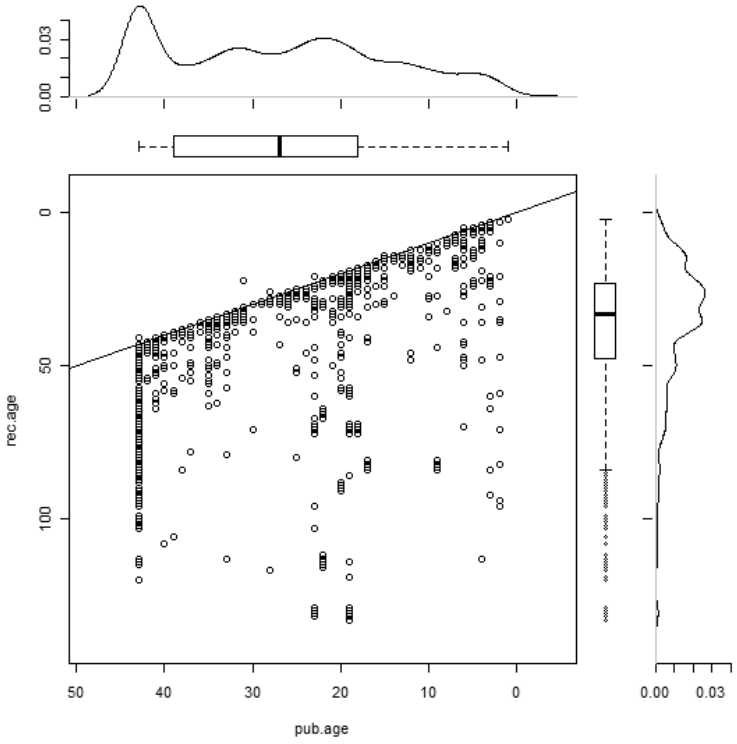

Fig. 4. Temporal distribution of views

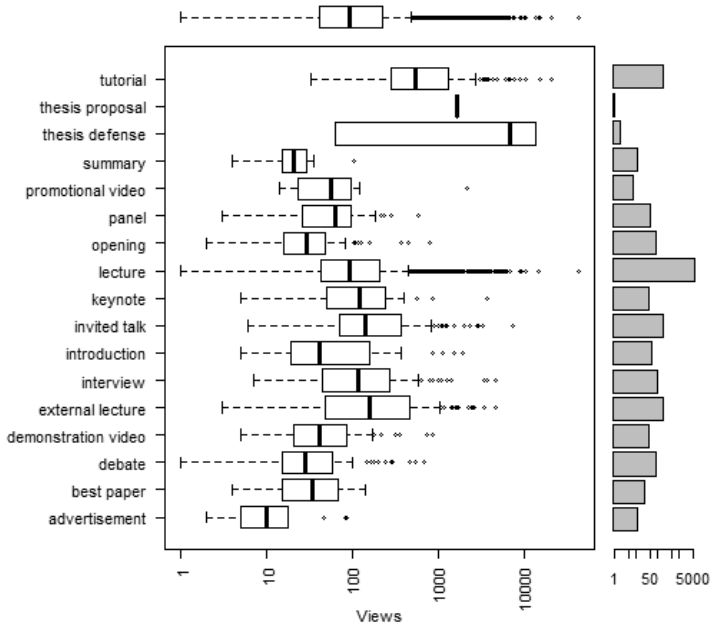

Fig. 5. Distribution of views considering item type

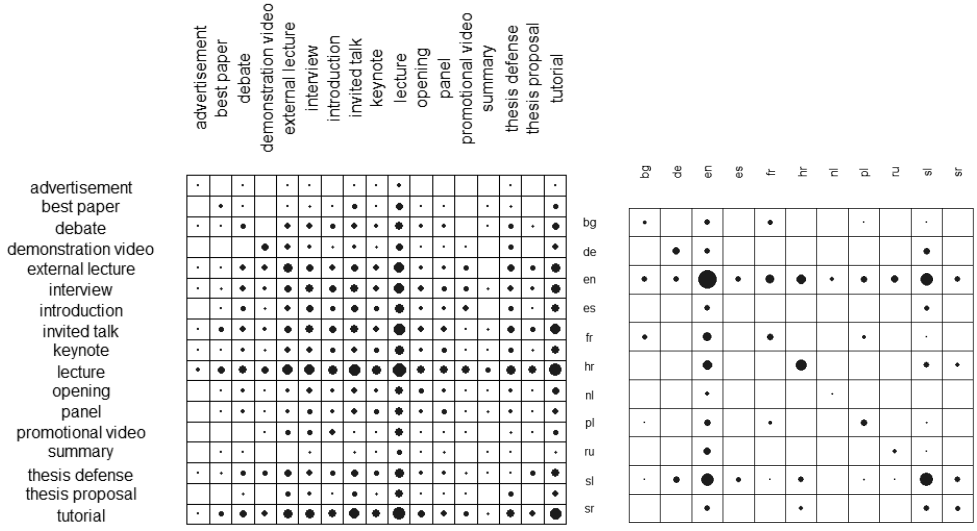

Fig. 6. Types and languages in lecture pairs

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.

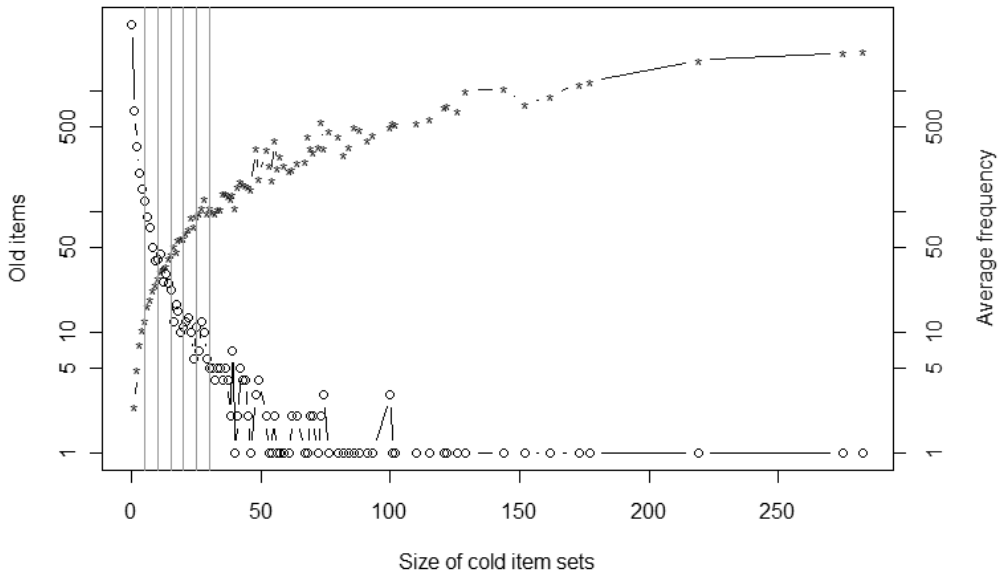

Fig. 7. Number of old items (o) and their pair average frequencies (*) on the size of cold item sets used in the evaluation

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)