Scientific Bulletin of the „Petru Maior" University of Tîrgu Mureş
Vol. 11 (XXVIII) no. 1, 2014
ISSN-L 1841-9267 (Print), ISSN 2285-438X (Online), ISSN 2286-3184 (CD-ROM)

ORGANIZATION AND CONSTRAINTS OF A RECOMMENDER SYSTEM FOR MOOCS

Dumitru RĂDOIU
"Petru Maior" University of Tîrgu Mureş
Nicolae Iorga Street, no.1, 540088, Tîrgu Mureş, Romania
dumitru.radoiu@upm.ro

Abstract

Recommender Systems (RSs) are software tools providing suggestions for users about items likely to be of interest for them.
MOOCs (Massive Open Online Courses) offer a vast amount of learning items to millions of users. Users face a challenging task of finding the suitable learning content, learning path and peer-learners for collaborative activities. Poor selections lead to very low completion rates, typically under 10%.
The paper assumes that a better decision-making process is likely to lead to a higher completion rate and discusses the organization and constraints of a Recommender Systems in a MOOCs context.
As the organization of a Learning Recommender System and the choice of algorithms represent the first two architecture design criteria for meaningful results, attention is given to this factors.
The paper argues that an effective Recommender System must focus on MOOCs specific type of items (learning items) and user behavior in MOOCs context and consequently, a number of particularities and constraints are identified and discussed.
In this paper, the term organization is used to describe how different components work together to form a system and the term architecture is used to describe a blue-print of how to actually build such a system.

Key words: recommender system organization, MOOCs

1. Introduction

We rely on recommendations when making almost all decisions: when hiring somebody, when choosing a book, a movie, a dish or a holiday destination.

The problem of meaningful recommendation becomes critical when the number of items of choice is large and the number of users making poor decisions is over 90%, which seems to be the case in point for MOOCs.

MOOCs (Massive Open Online Courses) are widely seen as a major part of a larger disruptive innovation taking place in higher education [2] and hence the importance to discuss the suitability, the organization and constraints of such a system. In the acronym MOOCs,

Massive relates to the large number of students (users) who simultaneously take courses; the focus of MOOCs being here on scalability and building a learning community.

Open relates to open registration (no prerequisites) and open content (free of charge accessible), although some providers offer their services for profit in exchange for college credits.

Online describes the used communication platform (Web 2.0), providing real time interaction (group collaboration) or automated feedback.

Courses relates to the value proposition, the learning resources (items), most of them self-paced, leading to certification, some (with start/end dates and offered for profit) leading to college credits.

The success of this new form of learning environment is huge; among the most known MOOCs providers, Coursera alone had registered about 2.8 million learners [3] in March 2013 and reached over 5 million in October, the same year [4].

The problem we want to alleviate is related to the low completion rate, typically under 10%. The main reason appears to be a poor selection of the courses (learning items), learning path and lower level of learning community interaction. With limited guidance (usually limited to a short course description), users find only after enrolment if the course is what they need, if it is too difficult or too

basic, too long or not connected to their learning path.

With an overwhelming variety of choices (providers, topics, courses, communities) the "one size fits all" behavior of most search engines is not supporting a qualified decision either.

The paper starts from the hypothesis that a customized Recommender System, based on user attributes, user behavior and item attributes can generate meaningful recommendations and can improve the completion rate.

A Recommender System focuses on a specific type of item (in our case learning items) and consequently, its organization, interface and algorithms used to generate the recommendations must be customized to provide recommendations for that specific type of item.

For the case in point, the Recommender Systems is imagined as a support for users (life-long learners) who lack sufficient knowledge, time or understanding of the context to evaluate the suitability of an item to their profile or learning path.

A RSs stores user attributes, context and behavior when interacting with items and employs specific algorithms to predict the most suitable next item for the user.

For instance, the book recommender system that assists users to select next book to read:

Fig. 1: "What should I read" recommender service: http://whatshouldireadnext.com/

We start by decomposing the recommendation system and discussing its parts:

What data is available and can be usefully exploited by a recommender system in a MOOCs environment?

In a MOOCs environment we may consider useful data the course description (item attributes), user profile (user attributes), user behavior (user system transactional data), and user context (e.g. time zone for collaborative activities).

What recommendations/ suggestions are meaningful for MOOCs users?

The Learning Recommender System discussed in this paper is an application which federates the learning resources (items) and helps students (users) to:

Identify suitable items (i.e. learning resources)

Recommend learning paths (a sequence of items)

Recommend peer-users for collaborative learning activities

What are the most suitable algorithms considering the type of input data, its dimension (millions of users, thousands of items) and expected recommendations?

The discussion here is related to the selection between two algorithms (collaborative filtering, content-based filtering) or a combination of them (hybrid)

What is the organization of a recommender system to acquire, maintain and update data?

How can we measure how good is the recommender system?

One straight forward way is to compare the completion rate of a group using the system with a witness group, not using the system.

2. Recommender System Concepts

Definitions:

Recommender Systems (RSs) are information processing systems that actively gather users-items transactional data with the goal to provide recommendations/suggestions for users about items likely to be of interest for them.

Data used by RSs refers to three kinds of objects: items, users, and transactions, i.e., relations between users and items

Items is the term used to denote what the system recommends to users. Items may be characterized by their complexity, their value or utility to the user.

Users are the persons using the items. Users may have very diverse characteristics and goals and therefore can be modelled by their attributes and by their behavior pattern data, for example, site browsing patterns.

Transaction is a recorded interaction between a user and an item.

When users interact with items, the transactional data is represented as a utility matrix (U Matrix).

Each user-item past interaction could be memorized through a value representing either the degree of preference of that user for that item (weight or rating) or a Boolean value (e.g. seen/not seen).

In the following example, the interaction between users and items is rated by the user on a scale from 1 to 5. Blanks represent the situation where the user has not rated the interaction or there was no interaction.

Table 2: A utility matrix (U matrix) representing ratings of items by users on a scale 1 to 5

Item	User 1	User 2	User 3	User 4	User 5
Item 1		5	4	4	
Item 2	4	1	5	4	
Item 3	3			5	4

Item 4				4	5
Item 5		5	5	5	

In practice, the matrix are much larger and much sparser, user data filling only a tiny fraction of it.

Utility matrix can be Boolean or with weights (ratings).

The rows in the utility matrix can be seen as vectors representing the profiles of items.

The columns represent past activity of users and is usually the starting point in finding similarities linking an active user to a group.

The goal of any recommendation system is to predict the blanks in the utility matrix.

For example, would User 5 like Item 5?

We might design our utility matrix to take into account Item attributes such as tutor name and title, number of lectures, independent work load. In this case we might note that Item 4 and Item 5 have the same tutors and, based on previous rating we might conclude that User 5 will like Item 5 either.

Based only on existing data, though, noting that User 4 rated similarly with User 5 Items 3 and Item 4, we can conclude that User 5 will like also Item 5.

It is not necessary to predict every blank entry in a utility matrix.

There are many types of RSs, some simple, some complex. Some are basic, knowledge poor (e.g. they use very simple and basic data, such as user ratings/evaluations for items), other more complex, using more data (e.g., they use attributes to describe the users and the items) and other even more sophisticated, using context and constraints (e.g. social relations, geo location or activities of the users).

3. Selecting Input Data

What data is available and what data is reliable?

User attributes could be data related to educational level, formal and informal learning, etc.

User behavior data is transactional data generated in the educational setting and could be collected during the process.

Item attributes are the easiest to collect as this data could be entered compulsory when uploading the items.

A more complete taxonomy of data sources for recommendation is provided by [7].

The existing Recommender Systems used in social networking platforms and e-commerce acquire user data both explicit and implicit. The user is asked to be proactive and to directly communicate to the system his/her profile data and she/he needs to do so because they must be perfectly identifiable in an e-commerce transaction (e.g. Amazon) or on a professional network (e.g. Linkedin). The system automatically extracts their main descriptors (user attributes) and data is reliable.

In learning environments users are reluctant to make their private life data available to a centralized system so user attributes will be less reliable.

It follows that the recommender system should use mostly user behavior data and item attributes. What about using item ratings (student feedback)?

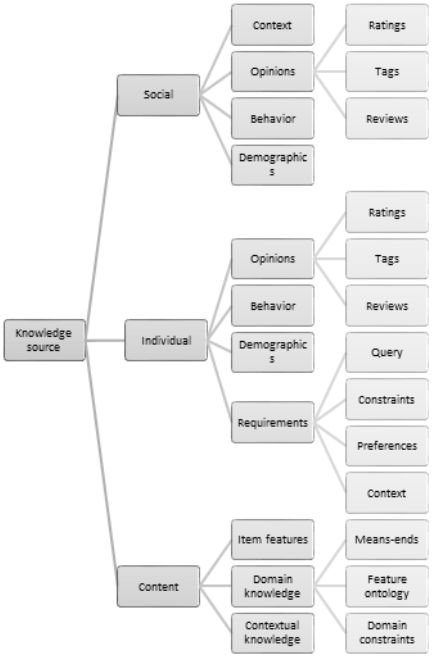

Fig. 2: Taxonomy of knowledge sources in recommendation [7]

Experimental data shows that – when students are asked to fill in feedback forms - response rates are low, unless it is made compulsory, in which case other reliability problems arise. In the end, what is interesting is highly subjective; users are not well-equipped to assess the academic value of teaching. More, they are not experts in the field and are not well-placed to assess the relative merits of a course.

Although it works for instance in movie recommender systems, user ratings does not seem to be reliable enough for recommendations about what to learn next.

To overcome these aspect, for reliable input data, we should probably assume Cranfield evaluation theory and trust that users are well represented by their behavior [5].

4. Basic Recommender System Algorithms

Most simple RCs algorithms use data from users who exhibit similar preferences or behaviors to generate recommendations for an active user. Basic recommendation algorithms are based on statistics and predictive modeling, basically represented by two algorithms or a combination of them:

collaborative filtering, based on the relationship between users and items; Similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items

content-based filtering, based on similarity of items measured through similarity in their properties

hybrid

Collaborative filtering (CF) uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users [1].

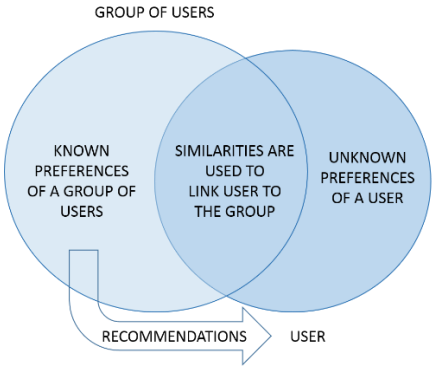

Figure 3 Collaborative filtering

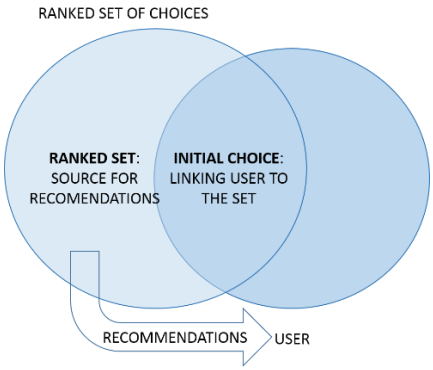

Figure 4 Content-based filtering

The recommendation is based on modelling prior user behavior, explicitly utilizing information generated by the entire user-base. The basic assumption is that if users A and B interact similarly with a set of products and/or services, chances are they will do the same in the future [2]

E.g. Using collaborative filtering, LinkedIn (linkedin.com) generates recommendations for people you might know, jobs you might like, groups you might want to follow, or companies you might be interested in. Amazon (amazon.com) uses content-based recommendation: when you select an item to purchase, Amazon recommends other items based on users' behavior who bought that item.

Collaborative filtering could be used in learning recommender systems (LRS) to recommend learning paths, by using the information from users with a similar profile like yours who started with the same course selection like you. Actually, the recommendation represent the most popular choice of course from the group sharing similar profile. As in the Venn diagram, similarities are "the frame of reference" and differences are the opportunities for recommendation.

Content-based filtering constructs a recommendation on the basis of a user's data or behavior, e.g. browsing history or profile data. In this case a reference table is required, specifying what a user with a certain profile will enjoy studying.

The goal is to create both an item profile consisting of feature-value pairs and a user profile summarizing the preferences of the user, based on their row of the utility matrix.

Similarity of items is determined by measuring the similarity in their properties.

Generate a profile for each item, i.e. item attributes. E.g. for an on-line course we can identify Boolean attributes:

Is there a learning community or automated interaction?

Are there machine-graded assessments or peer-reviewed written assignments?

Is the course self-paced?

Is there college credits offer?

Are there pre-requisites?

But we can think of an infinite number of other additional attributes…

Start/end dates

Credentials of the instructor

The advantage of Boolean attributes is that they can be represented as Boolean vectors and similarity between different courses can be expressed numerically as the cosine of the angle between the vectors.

Table 1

Item	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
Item 1	1	0	1	1	1
Item 2	1	1	0	0	1
Item 3	1	0	0	1	1
Item 4	0	1	1	1	1
Item 5	1	1	1	1	1

Next step is to create vectors with the same components that describe the user's preferences. We can do this using utility matrix which links users to items either via rating or via activity events (e.g. clicking through the course).

Table 2

Item	User 1	User 2	User 3	User 4	User 5
Item 1		1	1	1	
Item 2	1	1	1	1	
Item 3	1			1	1
Item 4				1	1
Item 5		1	1	1	

The vector representing User 5 preferences is the average of

Table 3

Item 3	1	0	0	1	1
Item 4	0	1	1	1	1

Thus we create a user vector with the same components that describe an item vector.

Table 4

Preferred Item	0.5	0.5	0.5	1	1

Recommendations are based on similarity between preferred item vector and the rest of item

vectors representing different course offerings, similarity measured by the cosine of the angle among vectors.

As the users are numbered by millions and the offered items by thousands, dimension reduction of high dimensional data is desired and different methods could be used like locality-sensitive hashing (LSH) and hyper plane tessellations [5,6].

A similar approach can be used when recommending documents or books. Documents could be represented by sets of words, the most used in the text. To measure the similarity of two documents we can use the distance between the sets of words or the cosine of the angle between the sets, treated as vectors.

Hybrid approaches that combine collaborative and content-based filtering approaches.

5. The Organization of a MOOCs Recommender System

Our considerations about input data led us to the conclusion is that recommender system should use mostly user behavior data and item attributes.

It follows that one of the characteristics of the system is that the interface must so designed as to be able to collect transactional data generated by the interaction of the users with the items space. Transactional data should be stored and later used for deeper analysis.

The organization of a recommender system is in principle the one in Fig. 5

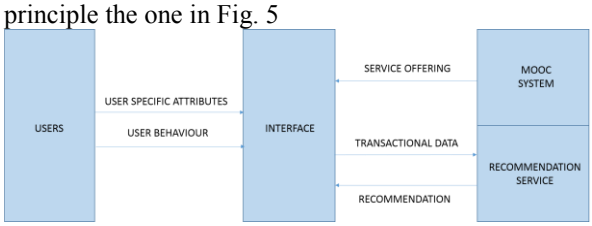

Fig. 5 Organization of a Recommender System for MOOC

6. Open Issues

The architecture of recommender systems and their evaluation on real-world problems is still an active area of research. The paper presents preliminary results in analyzing MOOCs RS and argues that such systems should rely on transactional data rather than ratings and user attributes. This has a direct impact on the organization of the RS in what regards the collection of data.

Selecting the appropriate algorithm to generate recommendations, evaluating the algorithms and data dimension reduction is still under study.

The paper does not address issues related to privacy or system evaluation.

References

[1] Xiaoyuan Su and Taghi M. Khoshgoftaar, Advances in Artificial Intelligence, Volume 2009, Article ID 421425, 19 pages, doi:10.1155/2009/421425

[2] Michael Barber, Katelyn Donnelly, Saad Rizvi, Higher Education and the revolution Ahead, http://www.insidehighered.com/sites/default/server_files/files/FINALEmbargoedAvalanchePaper1303028129.pdf, Accessed May 13, 2014.

[3] "Massive Open Online Courses, Aka MOOCs, Transform Higher Education and Science." Accessed May 13, 2014. http://www.scientificamerican.com/article/massive-open-online-courses-transform-higher-education-and-science/.

[4] Fowler, Geoffrey A. "An Early Report Card on Massive Open Online Courses." Wall Street Journal, October 8, 2013, sec. Special.

[5] Plan, Yaniv, and Roman Vershynin. "Dimension Reduction by Random Hyperplane Tessellations." Discrete & Computational Geometry 51, no. 2 (March 1, 2014): 438–61. doi:10.1007/s00454-013-9561-6.

[6] Andoni, A., and P. Indyk. "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions." In 47th Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS '06, 459–68, 2006. doi:10.1109/FOCS.2006.49.

[7] Recommender Systems Handbook. Accessed May 14, 2014. http://www.springer.com/computer/ai/book/978-0-387-85819-7.