Collaborative Filtering and Inference Rules for Context-Aware Learning Object Recommendation

Daniel Lemire[*], Harold Boley[†], Sean McGrath[‡], Marcel Ball[§]

September 6, 2005

Abstract

Learning objects strive for reusability in e-Learning to reduce cost and allow personalization of content. We argue that learning objects require adapted Information Retrieval systems. In the spirit of the Semantic Web, we discuss the semantic description, discovery, and composition of learning objects using Web-based MP3 objects as examples. As part of our project, we tag learning objects with both objective (e.g., title, date, and author) and subjective (e.g., quality and relevance) metadata. We study the application of collaborative filtering as prototyped in the RACOFI (Rule-Applying Collaborative Filtering) Composer system, which consists of two libraries and their associated engines: a collaborative filtering system and an inference rule system. We developed RACOFI to generate context-aware recommendation lists. Context is handled by multidimensional predictions produced from a database-driven scalable collaborative filtering algorithm. Rules are then applied to the predictions to customize the recommendations according to user profiles. The prototype is available at inDiscover.net.

1 Introduction

With the proliferation of the Internet, demand for on-line learning has grown rapidly. Often used to deliver inexpensive just-in-time information, students now expect similar services from learning institutions. For teachers, these new expectations can be challenging. The design and production of on-line courses is expensive and time-consuming. When providing digital content, it is no longer adequate to merely manually adjust or adapt the course content for students, students should also be empowered to navigate independently (Lundgren-Cayrol *et al.* , 2001).

The Web also faces similar challenges. As the Web becomes ubiquitous, our needs become more sophisticated. For example, while we may have been satisfied in the past with weather reports for a given area, we now want to be able to plan our vacations to other areas and thus, coordinate data coming from weather reports, hotels, and air travel companies. While we generally know how to find web sites on a given topic using search engines, we still can't easily find all air travel companies offering flights from Montréal to Rio next week for under 1000 dollars. It follows that we can't expect our computers to suggest travel packages automatically from data gathered over the Web. We observe that the Web is not a database (Mendelzon, 1998) in that there are no built in common schemas or sophisticated data retrieval mechanism. Yet the Web is the most successful data management tool ever developed. We distinguish two different future challenges for the Web: Information Retrieval and Composition. One approach to the solution to these problems can be found in the *Semantic Web* (Berners-Lee, 1998). Essentially, the Semantic Web adds to the current Web enough metadata so that the Web can be considered machine-parseable (Koivunen & Miller, 2001). In theory, it should render the Information Retrieval problem easier, and one approach to Composition can then be achieved through Artificial Intelligence (AI) planning techniques.

[*]UQAM, Montreal, QC, Canada (Work started while at NRC)
[†]NRC, Fredericton, NB, Canada
[‡]UNB, Fredericton, NB, Canada
[§]UNB, Fredericton, NB, Canada
In International Journal of Interactive Technology and Smart Education, Volume 2, Issue 3, August 2005.

Just like a Web site, digital knowledge and training material should be reusable. For example, once basic arithmetic is achieved, individual reuse this knowledge all their life; a document about arithmetic is also reusable in a similar fashion. In this spirit, the notion of a *learning object* (Downes, 2001; Gibbons *et al.* , 2002) was proposed as a way to enable content reuse in e-Learning. Essentially, a learning object is any component that can be used and reused for learning. A map, a Web site, a piece of software, and a video stream are all examples of learning objects. Thanks to projects such as *eduSource*, learning objects have become tangible (Bhavsar *et al.* , 2003; Bhavsar *et al.* , 2004). For example, KnowledgeAgora[1] offers a learning object taxonomy and a search engine. Users can sell, buy, exchange or offer learning objects. In KnowledgeAgora, unlike the Web at large, Intellectual Property, age range, and education level are explicitly handled, which is important for vital for content reuse.

Object composition in education is already present in fields such as adaptive testing (Raîche, 2000); however, learning objects belong to a more heterogeneous and distributed setting than educators are accustomed to. Few examples of computer supported learning object composition have been reported (Fiaidhi *et al.* , 2004; Fiaidhi, 2004). Learning objects have to be fine-grained enough so that reusability is sensible. That is, we need to have many small objects and we can aggregate to form larger objects, such as courses. The problem in locating learning objects is much harder than simply finding the proper digitalized textbook for a course. An example of the problem we want to solve would be to find all resources related to "Java inheritance" for 3rd year Computer Science students, such that the students are likely to find the content of at least average interest, but such that it was rated above average for accuracy by instructors.

One might think that information retrieval as accomplished on the Web can be applied to learning objects. After all, Google is very efficient at finding content as long as requests can be expressed as a list of keywords. However, one should note that Google works well in part because the Web is made of links in a fundamental way, unlike learning objects. The analogy between the HTML Web and learning objects is not perfect. Learning objects are not necessarily text-based and they are not linked to each other as explicitly as Web sites. Google searching is based on the assumption that a Web page frequently linked to must offer relevant content and returns such results to the searcher. Naturally, course content does include links and relationships, possibly through Dublin Core, but learning objects won't make up a graph the same way the Web is a directed graph of Web pages.

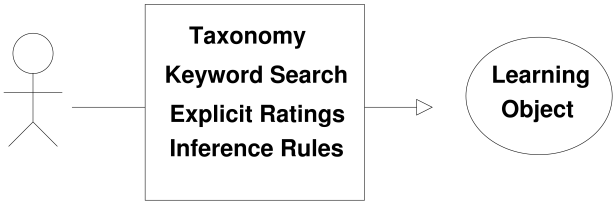

Figure 1: A few of the options a user has for finding learning objects.

Effectively, finding the right resource to express a given idea remains a challenge. When an instructor or student wants to find a particular type of learning object that is of interest to them, they need a way to specify their interest to the system that interfaces with the learning object repository. Fig. 1 shows some of the common methods of doing so: navigating through a taxonomy, performing keyword searches, specifying their tastes/interests through explicit ratings and using a collection of inference rules to filter objects.

Our approach is to focus primarily on explicit ratings and inference rules as a means of helping users discover what they are looking for, to increase discoverability. We found that taxonomies can have low user acceptance in instances where the categories tend to be subjective (such as a taxonomy of musical genres) and that keyword searches across a large set of objects tend to produce far too many results to efficiently sift through. For example, in KnowledgeAgora, a keyword search for "Java" returned 58 learning objects shortly after it was launched and more recently returned 361 records. In the future, it is quite likely that such a reques would return thousands of objects. Such information overload is a serious challenge to content reuse, even for one's own work, and although we can borrow some of the Web techniques, we also need new, adapted search solutions. However, even if we can find an appropriate object for a concept and a given stu-

[1] http://www.knowledgeagora.com/

dent profile, we still have work to do, as a course is not merely a set of objects. We have to find how to "fit" the object into the course. Hence, we are still far from generating personalized courses automatically from databases.

This paper is organized in the following manner. We begin by reviewing the semantic approaches to object filtering and composition, we then present ratings as subjective measures and collaborative filtering as a method to leverage them. The RACOFI architecture (Anderson *et al.* , 2003) is presented as consisting of two components: the SLOPE ONE collaborative filtering algorithm is presented as a good choice for learning object repositories, and the use of inference rules, to customize learning object selection. We conclude with object composition as viewed from the RACOFI model.

2 The Semantic Way

People are already using learning objects with precise descriptive educational metatags. For example, it is possible to find the author of any object in KnowledgeAgora. However, among the difficulties with such a semantic approach is the need for common ontologies so that we can exchange objects across institutions. For example, if we are interested in all free learning objects available, we must all agree on what a "free" object is. Can you modify a free object and redistribute it? If we want "rock" music, does "punk" music qualify? This problem cannot entirely go away (Downes, 2003): we will always have to be satisfied with links between objects and possible semantics, that is, not having a unique ontology.

In order to enable detailed searches on sets of learning objects, metadata specifications and standards have been proposed including IMS's LRM (*Instructional Management System's Learning Resource Metadata*) and IEEE LOM (*IEEE Learning Object Metadata*). These standards are rather detailed: SCORM contains about 60 fields upon which a user can complete a search including date, author, and keywords. Even if we suppose that users are willing to take the time needed to fill out many of these fields to make their searches narrow, and that the object has been carefully tagged, we can still end up with thousands of records (Downes, 2002). It is not clear that adding more and more fields will lead to better results: content creators may omit some information, make mistakes or even enter misleading data on purpose. For example, a resource target age might end up too often to be entered as 18-99 whether or not it is accurate. The problem will only intensify as learning object repositories grow in size.

Because we believe it will be hardly possible to have a (single) common ontology to define learning object semantics, because metadata records are complex, and because learning objects will become even more numerous, we will need to "predict" both the student and course creator needs, using heuristics. If we look back, we see that the same thing happened with the Web. More people find what they are looking for using Google than by using Yahoo's or DMOZ's category systems (taxonomies). It may be telling that, in 2004, Google removed from its main page a link to its copy of DMOZ's taxonomy. Google's strength is that it leverages knowledge about the use made of a resource, that is, the number of links a Web site received. We need a similar approach for learning objects.

One vision of the semantic *way* is to imagine that each learning object is coupled with an XML file describing the object in an objective and exhaustive way. We rather think that each object will be described in different ways by different people, maybe using RDF files. In some cases, like the title of a book, we can expect everyone to agree, but as soon as the granularity of the description goes beyond a certain point, differences will appear. In fact, the IEEE LOM standard already contains subjective fields such as "Interactivity Level", "Semantic Density" and "Difficulty". It is obvious that some people will disagree on the difficulty level of a given learning object. Learning object repositories such as Merlot already have peer review systems for learning objects. Other researchers look at integrating learning object reviews from different judges (Nesbit *et al.* , 2002; Han *et al.* , 2003) as part of communities such as eLera[2].

Once we accept subjectivity as a key ingredient for describing learning objects, it is not hard to imagine each student and course creator might be allowed to annotate learning objects. For example, we can ask students to review and rate a reference document included in an on-line course. We could use the results to determine automatically which students are most likely to benefit from this document (Recker *et al.* , 2003; Weatherley *et al.* , 2002;

[2] `http://elera.net/`

Downes *et al.* , 2004). Hence, from subjectivity, we get to personalized content, which benefits both the student and course creator.

3 What Are *Ratings* and What Can We Do with Them?

In Information Retrieval, we often make the following assumptions.

1. Semantics is at least partly computer-parseable, which most often means that content is text-based or described by text.

2. Users can express their queries in a convenient computer-parseable way such as by a range of dates or a few keywords.

3. It is acceptable for the result of the query to be an unbounded list of (partially) sorted objects.

The Semantic Web and the various metadata standard initiatives for learning objects try to solve the problems occurring when the first of these assumptions is not entirely met. However, even when the semantics is computer-parseable, perhaps because it has been carefully described using XML, other assumptions may not be true. Are users willing to express their requests using up to 60 fields?

For quite some time already, universities have asked students to rate their courses, professors and other elements. If we imagine each course as being a learning object, we will have an objective description of the course given by its title, number, list of pre-requisites and so on, while, on the other hand, we have evaluations made by past students of this same course. All of this data is metadata, "data about the course", but some of it is more subjective than others. So, because we already use subjective metadata in traditional learning, we believe we should also use it in e-Learning, and even go further.

When given the option, users will rate what is interesting to them: sites such as RatingZone, Amazon and Epinions are good examples. At the very least, we can measure implicit ratings: did a user return to this web site and how long did s/he remain on the site. Teaching institutions already measure how much interest courses receive by counting the number of registered students and thus, it seems appropriate to use similar measures with learning objects.

In the learning object context, we propose to predict a user's opinions or a class of users' overall opinions drawing on other users in the system. Hence, if we know what students with a given profile will think of a document, we can decide whether or not to include it in a course. We can also use knowledge about how an object was used in the past to help composition: "Dublin Core", IEEE LOM and IMS's LRM allow us to tag relationships between objects. Ideally, we could even hope to configure courses automatically, the same way we hope to see agents able to surf the web for us to organize our vacations.

The term *collaborative filtering* has its origins with Goldberg et al. (Goldberg *et al.* , 1992). It describes all techniques leveraging incomplete information about tastes and opinions of a set of users. We distinguish three types of applications for collaborative filtering: *gold balling*, *black balling* and *white balling*. In the first of the three, we are simply looking for what the user will prefer (Karypis, 2000). *Black balling* tries to virtually remove objects whereas *white balling* puts the user in charge and allows for thresholds. As an example of *white balling*, one could search for a document such that a given type of student will find it relevant and having an at least average graphical design. In general, we'd like to use both *white balling* and objective meta-data filtering to find, for example, all images representing an eye that have some subjective qualities.

The most widespread type of algorithms in collaborative filtering looks for neighbors to the current user in a set of users and do a weighted average of their opinions (Breese *et al.* , 1998; Pennock & Horvitz, 1999; Resnick *et al.* , 1994; Weiss & Indurkhya, 2001; Herlocker *et al.* , 1999). This approach works well in the case where we have enough users and relatively few objects rated on a single attribute. As the number of users grows, computational cost increases and other algorithms may become preferable for on-line applications. One way is to precompute relations across objects (Lemire & Maclachlan, 2005; Lemire, 2005; Sarwar *et al.* , 2001; Linden *et al.* , 2003). Not much work has been done for the more difficult case where each object is rated on multiple dimensions (using several attributes at once).

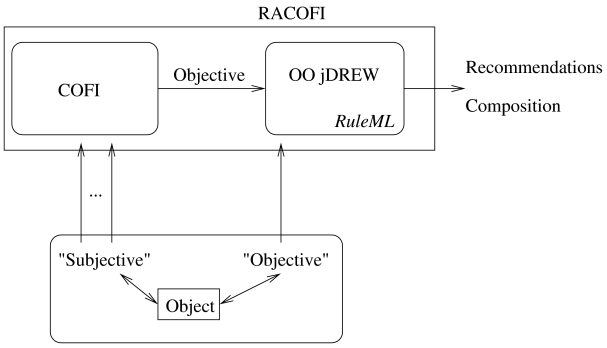

Figure 2: RACOFI architecture diagram.

4 RACOFI

In 2002-2003, the National Research Council of Canada (NRC) and Knowledge Pool Canada collaborated on the Sifter Project[3] in order to produce tools for the retrieval of learning objects in large databases. One of the results of this joint work was RACOFI (*Rule-Applying Collaborative Filtering*), which is a collaboration between Harold Boley and Daniel Lemire with 5 students and other researchers from the NRC.

RACOFI's goals are to process both objective and subjective metadata describing a given object. Hence, RACOFI is built with two software agents: a collaborative filtering library called COFI[4] and a rule engine called OO jDREW[5]. Objective metadata can be processed efficiently using rules written in XML, specifically in RuleML (Boley, 2003)[6]. The RuleML syntax is useful because it is very general and interoperable: it makes it possible to build generic Web services with little programming effort. COFI generates predictions from which OO jDREW then generates recommendations (see Fig. 2).

In order to validate this model, we used it in a Canadian Music recommender site called *RACOFI Music*[7]. A registered user can browse objects, rate them and propose new ones. Ratings can be done along 5 attributes on a scale from 0 to 10: overall impression, lyrics, music, originality, and production. Objective metadata includes title, author, release date, and a link allowing the user to listen to the music. As the user enters more ratings, recommendations typically become more useful. Since we use several attributes, the rule engine applies corresponding weights to each attribute in order to make a recommendation and the user can control these weights. Special rules are also used, for example, to increase predicted ratings to all objects from a given author if the current user liked at least one of the objects from this author. Overall, we found that a rule-based approach made it easy to adapt the system to user expectations.

Since the initial release of *RACOFI Music*, both the collaborative filtering and rule engines have evolved into *RACOFI Composer*, a framework designed for creating RACOFI powered systems, with one of the key features being the support of multi-dimensional ratings and recommendations. The framework is able to "plugin" to existing repositories so that users will be able to rate objects along any number of dimensions and get multiple lists of filtered recommendations.

5 SLOPE ONE Collaborative Filtering Algorithms

We are not particularly interested in using very sophisticated algorithms well fitted to modest data sets. Rather, we believe the challenge is to accommodate very large distributed data sets. Moreover, we believe that simplicity is a key feature for widespread use. Ideally, all aggregated values should be easily understood by the average engineer or IT specialist. We follow closely the work done in the context of Amazon rating systems (Linden *et al.* , 2003), which is based on Item-to-Item Similarity measures. In other words, we are interested in comparing learning objects two by two. However, unlike Amazon, we will not be content to merely discover association rules of the type "the course creator who used this learning object also used this other one". Amazon's goal is simple: sell books, DVDs, CDs, etc. Compared to this, the design of a course is a more complex task with numerous required outcomes.

Collaborative filtering for learning objects implies

- having to manage large sets of objects;

[3] `http://sifter.elg.ca/`
[4] `http://savannah.nongnu.org/projects/cofi/`
[5] `http://www.jdrew.org/oojdrew/`
[6] `http://www.ruleml.org/`
[7] `http://racofi.elg.ca`

- for each object, we have several attributes (multidimensional evaluation);
- an ever growing number of users having used the system or network;
- many different functions and needs depending on the context.

Each one of these characteristics poses scalability issues. Having thousands and thousands of objects is a challenge not often addressed in the literature (Linden *et al.* , 2003). To see the problem, imagine a database made of 100,000 learning objects each of which can be rated on a single subjective attribute. Suppose that we want to analyze relationships between objects. Observe that the number of different pairs of objects is such that even by allocating only 32 bits per pair of object, we still need over 37 MB in storage. It is easily seen that if each object can be rated on various attributes, the number of relationships becomes very large.

To offer fast query times, we propose to *precompute* relations among objects as in (Lemire, 2005; Lemire & Maclachlan, 2005; Sarwar *et al.* , 2001). The problem then becomes almost solely a storage issue. Because the problem is essentially multidimensional, multidimensional database techniques can be used (Codd *et al.* , 1993; Lemire, 2002; Kaser & Lemire, 2003) for highly efficient storage including Iceberg Cubes (Ng *et al.* , 2001).

We now describe the SLOPE ONE algorithm (Lemire & Maclachlan, 2005). Imagine that we have two objects 1 and 2. Suppose that we have simple unidimensional ratings by users Joe, Jill, and Stephen as follows.

	Object 1	Object 2
Joe	2/10	unrated
Jill	3/10	7/10
Stephen	4/10	6/10

We see that user Joe did not rate "Object 2". How can we predict his rating in the simplest possible way? One approach is to observe that on average "Object 2" was rated 3/10 higher than "Object 1" based on users Jill and Stephen, hence we can predict that Joe will rate "Object 2" 5/10. We call the algorithm SLOPE ONE because we take an actual rating x and add or subtract something to it, so that our predictor is of the form $x + b$ (a linear function with slope one) where b is an average difference. In general terms, this means that for any rating given by a user, we have a way to predict ratings on all other objects using average differences in ratings, and those can be precomputed. Suppose now that Joe has rated Objects A, B, and C and we must predict how he would have rated object D. Because Joe has 3 ratings, A, B, and C, we have 3 different predictions for how he would have rated object D. Hence, we must weight these 3 predictions and one easily accessible weight is the number of people who rated both objects as in this example: if 12 people rated both object A and D, 5 people rated both object B and D, and 321 people rated both object C and D, a reasonable formula is $\frac{12p_{\mathrm{A,D}}+5p_{\mathrm{B,D}}+321p_{\mathrm{C,D}}}{12+5+321}$ where $p_{\mathrm{A,D}}, p_{\mathrm{B,D}}, p_{\mathrm{C,D}}$ are the 3 predicted ratings for D given the rating on A, B, C respectively. In (Lemire & Maclachlan, 2005), this algorithm is called WEIGHTED SLOPE ONE. At the implementation level, for each pair of objects, we want to keep track of how many users rated both, and the sum of the differences in ratings. As previously noted, these aggregates are simple and easily updated when new data comes in. In practice, not all pairs of objects are significant: we would only store pairs for which the count, i.e., number of users who rated both, is significant (Ng *et al.* , 2001).

Because SLOPE ONE may appear simplistic, we need to benchmark it to make sure it has a reasonable prediction accuracy. We used the EachMovie and Movielens data sets made available by Compaq Research and the Grouplens Research Group respectively. EachMovie consists of movie ratings collected by a web site run over a few years: ratings range from 0.0 to 1.0 in increments of 0.2, whereas Movielens has ratings from 1 to 5 in increments of 1. EachMovie and Movielens are relatively modest data sets with only slightly more than 1600 and 3000 movies respectively. We selected enough users to have 50,000 ratings as a training set and tested the accuracy of the predictions over a test set made of at least 100,000 ratings (Lemire & Maclachlan, 2005). In our experiments, using the mean absolute error (MAE), we found WEIGHTED SLOPE ONE to be about 2-5% better than the similar but slightly more expensive ITEM-BASED algorithm (Sarwar *et al.* , 2001). While other algorithms are more accurate, we feel that given its simplicity and relative ease of implementation WEIGHTED SLOPE ONE is a good candidate for large learning object data sets.

The approach that we have taken to implement this

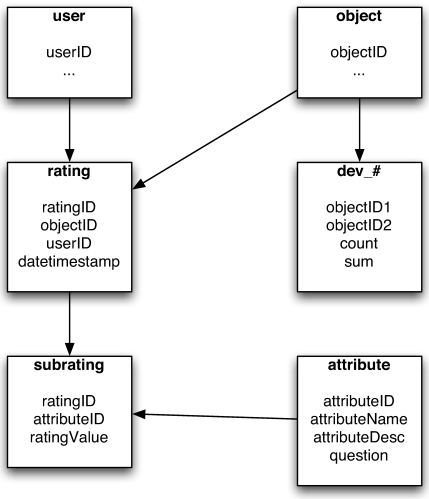

Figure 3: RACOFI Composer relational schema.

WEIGHTED SLOPE ONE algorithm involves the use of a relational database to store relationships between object pairs. These stored relationships are the number of users who have rated both objects and the sum of the rating differences between them, which we will be referring to as *count* and *sum* respectively. We chose to implement the algorithm this way as it allowed us the greatest amount of flexibility, portability and scalability in that our database schema (see Fig. 3) supports the simple addition/subtraction of objects, users and new attributes (or dimensions) to rate the objects against, as well as being able to support very small and very large data sets. Another benefit of this approach is that it keeps redundancy to a minimum.

Using the model outlined by the schema in Fig. 3, we have what is called a *dev* (short for deviation) table for each attribute stored in the database (this is indicated in Fig. 3 by the # symbol, which becomes the values of all *attributeIDs* in the *attribute* table). As can be seen in Algorithm 4, the implementation is simple. Besides using these *dev* tables for the WEIGHTED SLOPE ONE algorithm, we are able to draw several other interesting conclusions by analyzing the *count* and *sum* fields for specific object pairs. One simple conclusion that can be drawn

function predict(user u, object k):
INPUT: user u is modelled by a map from object IDs to rating values.
INPUT: object k is an object ID.
WILL RETURN: predicted rating by u on object k.
denominator $\leftarrow 0$
numerator $\leftarrow 0$
for each object j rated by u **do**
  1. get the count and sum from the dev table
  2. calculate the average difference
  3. denominator = denominator + count
  4. numerator = numerator + count $\times$ (average + the value u rated j)
end for
return numerator / denominator

Figure 4: Algorithm used to implement the WEIGHTED SLOPE ONE scheme.

is; "how many users who rated object *i* also rated object *j*", which is similar to one of the features found on Amazon[8]. In other words, the aggregates we precompute for the WEIGHTED SLOPE ONE algorithm have the potential to be reused for other applications.

6 Inference Rules

One shortcoming of using recommendations based soley on the collaborative filtering predictions is that the recommendations are limited by the subjective metadata that users have rated. By combining the collaborative filtering algorithm with an inference rule system we can use not only the subjective predicitions, but objective metadata about the objects, such as title, author and release date together with information from the user's profile to make the recommendations narrower and more personalized.

On the one hand, we argue that it might be preferable in a distributed and heterogeneous setting to use rule sets rather than to supplement directly the collaborative filtering algorithms with content-specific information using "bag of words" models, for example (Balabanovic & Shoham, 1997; Melville *et al.* , 2002). It is still possible to mine the rules, by e.g. Bayes models, but by feeding them in to a rule engine, system administrators, users and instructors can remain in control. Further, we can address diverse content and are not forced to deal with domain-specific methodologies as part of the system architecture.

[8] `http://www.amazon.com/`

On the other hand, filtering rules can help make the system more scalable: we can quickly eliminate or predict objects of interest using rules and thus, save memory and run-time cost.

One of our main uses of the inference rule system is to enhance the predictions, based upon objective similarities of the learning objects, to achieve better recommendations for the users of the system. For example, our common artist bonus assumes that if a user rated objects from a particular author highly, they are more likely to rate other objects by the same author highly. Based on this, we can add a rule to the system to increase the predicted rating for an attribute of an object if the user rated that attribute of another object with the same author highly:

> **If** the rating of attribute *a* for a rated object *ro* is high (e.g. 5)
> **and** the author of *ro* is the same as object *o*,
> **then** increase the predicted rating of attribute *a* of object *o*.

Another use for the inference rule system is to serve as a tool for filtering the recommendations in the system. For example, we could add a filtering rule to the system so that if the predicted value for the attribute of interest is below a threshold then we would block that object:

> **If** the predicted rating of attribute *a* of object *o* is less than 4
> **and** we are building a recommendation list for attribute *a*
> **then** block object *o*.

Finally, we envision that users can be represented as a set of specific rules which define their profile. Because we can express these rules using the XML application RuleML (Boley, 2003), these profiles can be understood by a wide range of systems regardless of the specific technology or purpose they have. Similarly, metadata about an object can embed RuleML-represented rules that can enforce some logic about the object.

By using information stored in user profiles we can augment our previous rules to make them more powerful. Our first rule could be changed to only apply if the the rules/facts attached to the user profile indicated that it should apply to that user. So, if you consider two users, John and Mary, with John's profile indicating to give the 'common-artist' bonus, and Mary's not indicating that, the rule above could be refined as:

> **If** a user *u* has the 'common-artist' bonus indicated in their profile
> **and** the rating of attribute *a* for a rated object *ro*, is rated highly (e.g. 5) by user *u*
> **and** the author of *ro* is the same as the author of object *o*,
> **then** increase the predicted rating of attribute *a* of object *o* for user *u*.

Based upon the users' profiles and on the first condition of the refined rule, the common-artist bonus would apply only to users that had this specified in the profile, like John, and not to other users, like Mary.

We can also augment our filtering rules to take advantage of the user profiles. For example, if we consider our two users, John and Mary, and they have paid for access to different parts of the system, we would not want to recommend to them types of content that they do not have access to. Therefore, we could augment their user profiles indicating which types of content they are and are not allowed to access, and introduce rules similar to the following one:

> **If** the object *o* is of type *t*
> **and** user *u* does not have access to objects of type *t*
> **then** block object *o* for user *u*.

7 Object Composition

By our definition, learning objects must be reusable, hence the context in which they are used can vary at least slightly: a schema explaining how to set up one's computer monitor can be used in a English/French translation course as well as in an engineering course.

So, we can't abstract out context and relations among objects in such a way that the problem is fundamentally multidimensional: An object is not simply 'good' or 'bad'. In order to better understand how collaborative filtering can be used for object composition, we designed a Web site called inDiscover[9] (see Fig. 5) allowing users

[9] `http://www.indiscover.net`

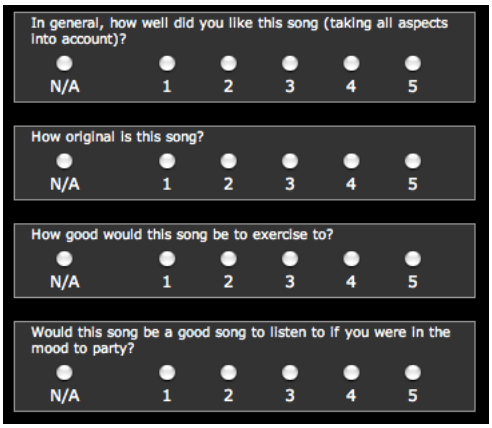

Figure 5: Example of an evaluation form for on-line MP3s. The screen shot comes from the Web site inDiscover prepared as part of the *RACOFI Composer* project.

to manage recommendation lists pointing to music files (MP3 format). The composition of a list for a given context is based on having objects with higher predicted values earlier in the recommendation list. Richer composition is possible in principle such as mood development for recommendation lists that are musical playlists. While the Web site is targeted to independent musicians as a tool to help them increase their visibility, our long-term motive is to learn how to compose objects into meaningful, context-aware lists. In inDiscover, each user can have a recommendation list for each type of context (mood/situation): party, workout, relaxation, and so on.

8 Conclusions

We take for granted that learning object repositories are part of the future of e-Learning because of the great potential for cost reduction and personalized instruction. However, we see several challenges before learning objects can fulfill their potential. We believe that collaborative filtering and rules can do for learning objects what Google did for the Web. The challenge will be to meet the real (collaborative and inferential) needs expressed by course creators and students. We are currently exploring the next steps in MP3 recommendation together with Bell Canada, using inDiscover community building as a commercial example.

9 Acknowledgments

We would like to thank Anna Maclachlan for her advice and contribution to the design of the SLOPE ONE collaborative filtering algorithms. We would like to thank Susan O'Donnell, Murray Crease, and Jo Lumsden for their Human-Interface expertise. We would also like to thank our Group Leader Bruce Spencer for his support and the NRC e-Learning Group for advising us with e-Learning topics.

References

Anderson, M., Ball, M., Boley, H., Greene, S., Howse, N., Lemire, D., & McGrath, S. 2003 (October). RACOFI: A Rule-Applying Collaborative Filtering System. *In: Proceedings of COLA'03*. IEEE/WIC.

Balabanovic, M., & Shoham, Y. 1997. Fab: Content-based, collaborative recommendation. *Communication of the ACM*, **40**(3), 66–72.

Berners-Lee, Tim. 1998. *A roadmap to the Semantic Web*.

Bhavsar, Virendra C., Boley, Harold, & Yang, Lu. 2003. A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments. *Pages 53–72 of: Proc. Business Agents and the Semantic Web (BASeWEB) Workshop*. NRC 45836.

Bhavsar, Virendra C., Boley, Harold, Hirtle, David, Singh, Anurag, Sun, Zhongwei, & Yang, Lu. 2004. A Match-Making System for Learners and Learning Objects. *Learning & Leading with Technology*.

Boley, Harold. 2003. Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms. *In: Proc. Rules and Rule Markup Languages for the Semantic Web (RuleML-2003)*. Sanibel Island, Florida, LNCS 2876, Springer-Verlag.

Breese, J. S., Heckerman, D., & Kadie, C. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *In: Fourteenth Conference on Uncertainty in AI*. Morgan Kaufmann.

Codd, E. F., Codd, S., & Salley, C. 1993. *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*. Tech. rept. E. F. Codd & Associates.

Downes, S., Fournier, H., & Regoui, C. 2004 (May). *Projecting Quality*. MADLaT.

Downes, Stephen. 2001. *Learning Objects: Resources For Distance Education Worldwide*. International Review of Research in Open and Distance Learning.

Downes, Stephen. 2002. *Topic Representation and Learning Object Metadata*.

Downes, Stephen. 2003 (August). *Meaning, Use and Metadata*.

Fiaidhi, J. 2004. RecoSearch: A Model for Collaboratively Filtering Java Learning Objects. *ITL*, **1**(7).

Fiaidhi, J., Passi, K., & Mohammed, S. 2004. Developing a Framework for Learning Objects Search Engine. *In: Internet Computing'04*.

Gibbons, Andrew S., Nelson, Jon, & Richards, Robert. 2002. *The Instructional Use of Learning Objects*. Agency for Instructional Technology. Chap. The Nature and Origin of Instructional Objects.

Goldberg, D., Nichols, D., B.M.Oki, & Terry, D. 1992. Using Collaborative Filtering to Weave and Information Tapestry. *CACM*, **35**(12), 61–70.

Han, K., Kumar, V., & Nesbit, J.C. 2003 (November). Rating learning object quality with bayesian belief networks. *In: E-Learn*.

Herlocker, J., Konstan, J., Borchers, A., & Riedl, J. 1999. An Algorithmic Framework for Performing Collaborative Filtering. *In: Proc. of Research and Development in Information Retrieval*.

Karypis, G. 2000. *Evaluation of Item-Based Top-N Recommendation Algorithms*. Tech. rept. 00-046. University of Minnesota, Department of Computer Science.

Kaser, Owen, & Lemire, Daniel. 2003 (November). Attribute Value Reordering for Efficient Hybrid OLAP. *In: DOLAP*.

Koivunen, Marja-Riitta, & Miller, Eric. 2001. W3C Semantic Web Activity. *In: Semantic Web Kick-off Seminar*.

Lemire, Daniel. 2002 (October). Wavelet-Based Relative Prefix Sum Methods for Range Sum Queries in Data Cubes (Best Paper). *In: CASCON'02*.

Lemire, Daniel. 2005. Scale and Translation Invariant Collaborative Filtering Systems. *Information Retrieval*, **8**(1), 129–150.

Lemire, Daniel, & Maclachlan, Anna. 2005. Slope One Predictors for Online Rating-Based Collaborative Filtering. *In: SIAM Data Mining*.

Linden, Greg, Smith, Brent, & York, Jeremy. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, **7**(1), 76–80.

Lundgren-Cayrol, Karin, Paquette, Gilbert, Miara, Alexis, andJacques Rivard, Fréderick Bergeron, & Rosca, Ioan. 2001. Explor@ Advisory Agent: Tracing the Student's Trail. *In: WebNet'01 Conference*.

Melville, P., Mooney, R. J., & Nagarajan, R. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. *Pages 187–1992 of: AAAI/IAAI*.

Mendelzon, Alberto O. 1998. The Web is not a Database. *Pages 0– of: Workshop on Web Information and Data Management*.

Nesbit, J., Belfer, K., & Vargo, J. 2002. A Convergent Participation Model for Evaluation of Learning Objects. *Canadian Journal of Learning and Technology*, **28**(3).

Ng, Raymong, Wagner, Alan, & Yin, Yu. 2001. Iceberg-cube Computation with PC Clusters. *In: ACM SIGMOD 2001*.

Pennock, D. M., & Horvitz, E. 1999. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach. *In: IJCAI-99*.

Raîche, Gilles. 2000. *La distribution d'échantillonnage de l'estimateur du niveau d'habileté en testing adaptatif en fonction de deux règles d'arrêt : selon l'erreur-type et selon le nombre d'items administrés*. Ph.D. thesis, Université de Montréal.

Recker, M., Walker, A., & Lawless, K. 2003. What do you recommend? Implementation and analyses of collaborative filtering of Web resources for education. *Instructional Science*, **31**(4), 229–316.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. *Pages 175–186 of: Proc. ACM Computer Supported Cooperative Work*.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. 2001. Item-based Collaborative Filtering Recommender Algorithms. *In: WWW10*.

Weatherley, John, Sumner, Tamara, Khoo, Michael, Wright, Michael, & Hoffmann, Marcel. 2002. Partnership reviewing: a cooperative approach for peer review of complex educational resources. *Pages 106–114 of: Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries*. ACM Press.

Weiss, S.M., & Indurkhya, N. 2001. Lightweight Collaborative Filtering Method for Binary Encoded Data. *In: PKDD '01*.