Meta-rules: Improving Adaptation in Recommendation Systems

Vicente Arturo Romero Zaldivar[1] and Daniel Burgos[1,2]
[1]Atos Origin SAE, Albarracin 25, Madrid 28037, Spain
{vicente.romero, daniel.burgos}@atosresearch.eu
www.atosresearch.eu
[2]International University of La Rioja
Gran Via Rey Juan Carlos I 41, 26002 Logroño, La Rioja, Spain
www.unir.net

Abstract

Recommendation Systems are central in current applications to help the user find useful information spread in large amounts of post, videos or social networks. Most Recommendation Systems are more effective when huge amounts of user data are available in order to calculate similarities between users. Educational applications are not popular enough in order to generate large amount of data. In this context, rule-based Recommendation Systems are a better solution. Rules are in most cases written *a priori* by domain experts; they can offer good recommendations with even no application of usage information. However large rule-sets are hard to maintain, reengineer and adapt to user goals and preferences. Meta-rules, rules that generate rules, can generalize a rule-set providing bases for adaptation, reengineering and on the fly generation. In this paper, the authors expose the benefits of meta-rules implemented as part of a meta-rule based Recommendation System. This is an effective solution to provide a personalized recommendation to the learner, and constitutes a new approach in rule-based Recommendation Systems.

1 Introduction

Nowadays, when the amount of information is becoming over-exceeding, Recommendation Systems emerge as the solution to find the small piece of gold in mountains of garbage. In electronic commerce, knowledge management systems, social networks, and other fields and markets, they help users find useful products, lessons or contributions. There are many inputs which can be used as information sources like i.e. user similarities with other users, user profile, and user preferences. All these inputs provide the system with valuable data to suggest the user the best way to follow or the most appropriate choice. Furthermore, people ratings [Rocchio, 1971] are another important source of information for Recommendation Systems.

Other sources of information are i.e. user interests, goals, and objectives, all of them more useful for educational applications. However, educational applications lack of enough amounts of data to establish user similarities in a precise way. In this case, recommendations are based on information stored in a user model which is extended explicitly or implicitly. There are also hybrid approaches which ask some minimum information to the user and the rest is obtained in an implicit way.

For educational applications ***rule-based Recommendation Systems*** have proved as more useful than other systems [Abel *et al.*, 2008]. In general acceptable recommendations can be obtained with a small amount of information. However, when the system achieves a better knowledge of the user, recommendations increase precision since rules evolve in parallel or new ones are included to the rule-set. In general expressing user preferences, goals and interest with rules can be difficult [Anderson *et al.*, 2003] to solve this problem complex and large rule-sets are generated. This solution carries another problem: the size and complexity of the rule-set can be unaffordable. In addition, it would be desirable to generate rules based on data extracted from a database or from the user, increasing this way user adaptability. Such generation could also be on the fly, allowing the rule-set to be up to date.

In this paper we propose a solution to this problem by the introduction of a new abstraction level: ***meta-rules***, which provide foundations for effective adaptive and personalized processes, such as in, e.g. learning. This approach has been implemented in the context of Meta-Mender, our meta-rule-based Recommendation System. In this paper, we also describe the Meta-Mender architecture and implementation to contextualize the meta-rules approach.

2 Background and Related Work

In the field of educational recommendations there are approaches like [El Helou *et al*., 2009] which use a modified page ranking algorithm for the generation of recommendations. The algorithm considers actors, activities and resources as main entities. Here user' activity is used to create a directed graph representing the entities and links between them are generated. Later a rank is assigned to nodes and this information is used to generate recommendations for a user query. This work is valuable for us because in general meta-rules use as input the user's activity for the automatic generation of rules. This input comes as in the referenced paper from actors, activities and resources. The main difference is that in our approach it will be generated a set of rules instead of a directed graph.

In the field of rule-based recommenders there are some relevant reports in the literature [Abel *et al.*, 2008], for example, describes a rule based Recommendation System for online discussion forums for the educational online board Comtella-D. Actually the system is able to call several encapsulated recommenders, collaborative filtering or

content-based recommenders, and the rules decide according to the amount and type of user data which recommender should be called. In doing so, the rules define a meta-recommender which is very interesting but tangential to this work.

An advantage of rule-based recommenders against other approaches is how easy it is to generate explanations for these systems. In many cases, it is almost impossible to explain to the user how a Recommendation System has derived a conclusion. If the user is not sure of a given recommendation and-or prefers to receive some logic-supported arguments which help him to choose a given solution, rules-based systems are the best ones prepared to solve this issue. This problem is usual in automated collaborative filtering systems [Herlocker, 1999], where the lack of explanations decreases the system acceptance and affects user trust.

RACOFI [Anderson *et al.*, 2003] is defined by its authors as a rule-applying collaborative filtering system. This system is a hybrid conformed by a collaborative filtering recommender plus a rule-based recommender. It was designed for recommending Canadian music but its authors argue that the system is content independent. RACOFI uses rules to modify, for example, ratings of items based on item similarity. This means that if a user rates an album as highly original, other albums' originality of the same author will be incremented. This paper is very useful for us because it contains a large set or rules which will be used to prove the synthesis power of a meta-rule approach.

With regards to meta-rules, we have not found anything similar to our proposal. Initially used by LISP [McCarthy *et al.*, 1985] and other languages, the closest concept is rule templates, followed by Open Rules[1] and the Object Oriented RuleML[2] approach of handling rules as data, thus generating entire rules from its component parts. These approaches are very valuable. However, our approach of producing rules using imperative programming comprises these two approaches and, at the same time, it seems to be more powerful. For instance, Meta-Mender can easily generate meta-rules, or other supplementary features, easily. In addition, as it will be shown in the next sections, using meta-rules in Rule-based Recommender Systems is not common, and it provides other benefits like maintainability, reengineering and on-the-fly generation. Finally, the introduction of another level of abstraction is very valuable for adaptation and performance.

For the implementation of a rule-based recommender using a rule-management system can be of great help. In this respect, Meta-Mender makes use of DROOLS [DROOLS] as rule engine. DROOLS is a business rule management system (BRMS) with a forward chaining inference-based rules engine (the so-called rule system). This system makes use of an enhanced implementation of the Rete algorithm [Forgy, 1982]. DROOLS is designed to allow plug-able language implementations. Currently, rules can be written in Java[3], MVEL[4], Python[5], and Groovy[6]. It is also possible to write functions to be executed as the consequence of any rule; this feature has been used to generate rules from meta-rules. It is also possible to assign a priority to rules, which is a way to address the execution order by the rule designer.

[1] http://openrules.com/index.htm
[2] http://ruleml.org/indoo/indoo.html
[3] http://www.sun.com/java/
[4] http://mvel.codehaus.org/
[5] http://www.python.org/
[6] http://groovy.codehaus.org/

3 The Meta-Mender Architecture and Implementation

As aforementioned, the Meta-Mender Recommendation System uses DROOLS as the rule engine. This rule system works as follows, first it is required to feed the engine with a set of rules (or meta-rules in this case); these meta-rules are defined by the professor or by the technical team using the application requirements. To define meta-rules is a more complex task than to define rules. In this respect, future research will be done in order to generate meta-rules automatically. A meta-rule example will be shown in the next subsection. Later some facts should be added to the engine. These facts are extracted commonly from a database, be it relational or ontological. As the facts are inserted, rules antecedents are checked for completeness and once the engine is started, rules that fulfill its antecedents are fired and the corresponding consequences are executed. The order of execution is arbitrary, so rule priority must be stated if the order of execution is important for the final result. In our case the order is important because the output of this iteration will be a file of rules and these files have a structure, so the rules that generate the header must be executed before the rules that generate the body. The output is used for a second iteration from which the final recommendations are obtained. See Figure 1 for a simplified representation of the recommender architecture and Figure 2 for an example of a meta-rules file, this file can generate rules for recommending the next course to follow in a .LRN educational application. See that at the consequence of the meta-rules a function is called, this function receives some information that allows it to write the rules wanted.

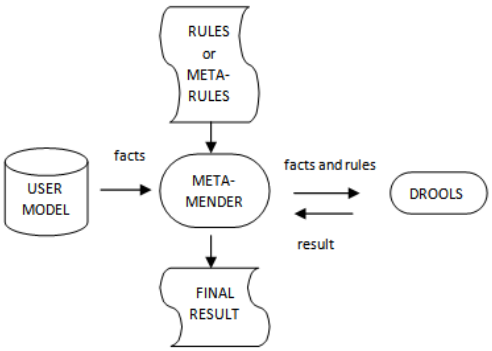

Figure 1. The Meta-Mender Architecture

It is worth mentioning that in DROOLS, rule conditions cannot contain functions, this is a common issue in rule systems the behind reason is performance. The condition is formed by a conjunction of patterns that must be matched by the inference algorithm. These patterns correspond at the end with object instances. So rule conditions are in a way fixed and must be defined statically. The only way to adapt a rule-set to changing conditions is by generating rules dynamically. These changing conditions can be:

- Changes in the application architecture, this includes the addition of a new forum or social network.

- Changes in membership conditions, access restrictions, or similar.
- Changes in model, new classes addition, modification of existing classes, etc.

It is clear that a static defined condition cannot cope with all these changes. So if there is a way for generating rules according to current system conditions and properties, it could save time and effort as long as increase adaptation. See Figure 3 for an example of a rules file obtained from the meta-rules file of Figure 2. In this figure it can be noted that the recommendation list is created as rules consequences are executed. The function `AddRecommendedCourse()` builds this list and the parameter `(10 - $courseLevel)` allows giving more priority to those courses with a lesser level of complexity. This criterion can be personalized to every student.

Meta-rules can handle changing conditions. Different rules can be generated depending on user data, system properties, etc.; also rules priority can be different which implies different recommendations.

```
package service

//global variables definitions
global java.lang.String filePath;

//rules
rule Header
//empty antecedent, executes always
  when
  then
    WriteHeader(filePath);
end

rule CourseSequenceRule
  salience -1 //priority
  when //antecedent
  //classId is an object field
    LRNClassData($classId : classId,
        $className : className)
    LRNStudentData($studentId :
      studentId,
      $studentName : studentName)
    LRNClassPerStudent(classId ==
      $classId, studentId == $studentId)
  then
    WriteClassMembershipRule(filePath,
      $studentName, $className);
end

//function implementation
function void WriteHeader(
       String filePath) {
 ...
}

function void WriteClassMembershipRule(
       String filePath,
       String studentName,
       String className) {
    ...
}
```

Figure 2. A Meta-rule File

Rule output can be related with rule priority since it will be possible to order the results and give the user a list of recommendations ordered by priorities.

4 Practical Implementations of the Meta-Mender Recommendation System

The Meta-Mender recommender is been used at present as part of two projects. The first one, TELMA[7], is focused on the application of Communications and Information Technologies in lifelong training of surgeons of Minimally Invasive Surgery (MIS).

TELMA develops an online learning environment which manages the content and knowledge generated by users in an efficient way. TELMA creates a new training strategy based on knowledge management, cooperative work and communications and information technologies aiming to the improvement of the formation process of the surgeons of MIS.

In order to cover all these needs and achieve its goals, TELMA develops a cooperative and adaptable learning platform. This platform is composed of a training system which integrates a tool for the authoring of didactic multimedia content, a Recommendation System and a social network. The Recommendation System, Meta-Mender, manages the knowledge generated by the users creating the foundations for adaptive learning. The learning platform allows the construction of a complete knowledge base thanks to the reuse and sharing of the knowledge generated for the professionals who access the learning system.

```
package service

global domain.RulesOutData outData;

rule MathI
  when
    UserData($userName : name,
      $userId : id)
    Course($courseLevel : courseLevel,
      $courseName : name, id == 1)
    not (TotalCoursePercentage(userId ==
      $userId, courseId == 1))
  then
    outData.AddRecommendedCourse(
     $courseName, 10 - $courseLevel,
     $userName);
end

rule AlgebraI
  when
    UserData($userName : name,
      $userId : id)
    Course($courseLevel : courseLevel,
      $courseName : name, id == 2)
    not (TotalCoursePercentage(
      userId == $userId, courseId == 2))
    exists (TotalCoursePercentage(
      userId == $userId, courseId == 1))
  then
    outData.AddRecommendedCourse(
      $courseName, 10 - $courseLevel,
      $userName);
end
```

Figure 3. A Rules File Obtained from the Meta-rules File of Figure 2

The second project, GAME·TEL[8], is focused on the creation of a system for the design, development, execution and evaluation of educational games and simulations,

[7] www.ines.org.es/telma
[8] www.ines.org.es/gametel

adapted to student preferences, educational goals, profile, [Burgos *et al.*, 2007].

The games and simulations are conversational adventures which are both usable and understandable. These characteristics benefits to the content creator, the professor in most cases, as long as to the final user, usually the student [Moreno-Ger *et al.*, 2008; Torrente *et al.*, 2008].

The software system is composed of several interconnected modules which allow the integration and intercommunication between games and simulations and several tools widely used for communities of professors. Initially these tools are the learning management systems Moodle and .LRN, and the authoring and learning units execution system LAMS [Burgos *et al.*, 2006; Moreno-Ger *et al.*, 2006]. In this context the Meta-Mender Recommendation System will suggest to users the best learning path according with their preferences, goals and objectives and will help with the game adaptation problem. As an example, the following meta-rule allows the generation of rules for the forums that the user has access to, see Figure 4.

```
rule ForumRecommendationRule
salience -1
when
  TelmaUser($userId : userId, $userName :
    userName, $mainInterest:
    mainInterest)
  TelmaForum($forumId : forumId,
    $mainTopic : mainTopic)
  TelmaForumAccessPerStudent(forumId ==
    $forumId, userId == $userId)
then
  WriteForumRecommendationRules(filePath,
    $userId, $userName, $forumId,
    $mainTopic);
end
```

Figure 4. A Meta-rule from Telma Application

This kind of rules allows for adaptation in case of the addition of a new forum to the application, a fact that can happen at any moment.

```
modify(amount->"0.5";
    comment->"Adjusting originality
    rating (by 0.5) for high ratings
    of other albums by this artist.";
    variable->originality;
    product->?item)
  :-
    rating(itemID->?item2;
    originality->"9.0"!?REST0),
    product(itemID->?item2;
    artist->?artist!?REST1),
    product(itemID->?item;
    artist->?artist!?REST2).
```

Figure 5. A Modify Rule from the RACOFI System

5 Expressive Power of Meta-rules

In this section we provide an example of the expressiveness power of meta-rules. We expose how a large set of rules can be generated from some few meta-rules.

```
tax(amount->"%15";
    comment->"15 percent HST")
  :-
    location(nb).
```

Figure 6. A Tax Rule from the RACOFI System

To this extend, we lean on the set of rules published on [Anderson *et al.*, 2003]. This rule-set contains 20 rules that modify user ratings based on item similarity. An example of these modify rules can be seen on Figure 5.

Other set of rules are the tax rules, there are 20 tax rules in the referenced paper. See Figure 6 for an example of these rules.

Finally the last set of rules is the so-called: `NotOffered` rules. A sample of these rules can be seen on Figure 7. There are 12 such rules in the rule-set.

```
NotOffered(itemID->?itemID)
  :-
    userLevel(beginner),
    product(itemID->?itemID;
      impression->?IMP!?REST),
    $lt(?IMP,7,true).
```

Figure 7. A NotOffered Rule from the RACOFI System

So the RACOFI rule-set defines more than 50 rules. This number is not very high but its maintainability can consume a lot of time. Also it is very hard to confirm that the rule-set is consistent with current data, suppose that it is necessary to modify a tax or a rating, it would be necessary to traverse the affected rules to check that everything is correct. Also on the fly rule generation, in order to increase adaptation, is a feature not easily covered with a static rule-set.

```
rule modifyMetarule
  when
    RatingAmountPair($amount : amount,
      $rating : rating)
    ProductMetadata($metadata : metadata)
  then
    WriteModifyRule(filePath, $metadata,
      $amount, $rating);
end

rule taxMetarule
  salience -1
  when
    TaxData($amount : amount,
      $location : location)
  then
    WriteTaxRule(filePath,
      $amount, $location);
end

rule notOfferedMetarule
  salience -2
  when
    NotOfferedData($maximum : maximum,
      $metadata : metadata, $student
      student, $userLevel : userLevel)
  then
    WriteNotOfferedRule(filePath,
      $maximum, $metadata, $student,
      $userLevel);
end
```

Figure 8. A Meta-rule-set that Generates the RACOFI Rule-set

The meta-rule approach discussed in this paper and implemented in our Meta-Mender Recommendation System is very useful to solve the problems mentioned above. Identifying common rule´s structure it is possible to write a concise meta-rule-set able to generate on the fly any

number of rules. This metadata, because meta-rules usually gets metadata as input, driven rule generation help solving the consistency problem, because rules can be regenerated at any moment after a change in the metadata.

Adaptation is also enhanced because different rules can be generated depending on user goals, needs and interests. As a side effect the development time of a rule-set is drastically reduced as long as a large number of rules can be generated with only one meta-rule.

As an example a set of meta-rules able to generate the whole rule-set of the RACOFI system is shown in Figure 8.

```
function void WriteModifyRule(
  String filePath, String metadata,
  String amount, String rating)
{
   FileWriter output = new
     FileWriter(filePath, true);
   MessageFormat mFormat = new
     MessageFormat("");
   output.write(mFormat.format(
     "modify(amount-> \"{0}\";\n",
     new Object[]{amount}));
   output.write(
     mFormat.format(
     "        variable->{0};\n",
     new Object[]{metadata}));
   output.write("        product->?item)\n");
   output.write("    :-\n");
   output.write(mFormat.format(
     "       rating(itemID->?item2;" +
     "{0}-     >\"{1}\"!?REST0),\n",
     new Object[]{metadata, rating}));
   output.write("       product(" +
     "itemID->?item2;artist-" +
     ">?artist!?REST1),\n");
   output.write("       product(" +
     "itemID->?item;artist-" +
     ">?artist!?REST2).\n");
  output.close();
}
```

Figure 9. The WriteModifyRule Function

In Figure 9 it can be seen a function that generates the modify rule-set of the RACOFI System. The rest of the functions referenced in Figure 8 are similar to this one.

6 Conclusions and Future Work

In this paper, we present a meta-rule based approach for rule generation. Meta-rules are rules that generate rules, and are able to generalize a rule-set providing bases for adaptation, reengineering and on the fly generation. In this paper, the benefits of meta-rules have been exposed. As an implementation example some details of Meta-Mender a meta-rule based Recommendation System have been also presented. The Meta-Mender Recommendation System is a component of, at present, two educational applications in development and test. The concept the Meta-Mender is based on, meta-rules for adaptation starts a new branch in the recommendation field and broadens the scope for new solutions in the field.

Meta-rules are an effective solution to provide a personalized recommendation to the learner, and constitute a new approach in rule-based Recommendation Systems.

Meta-rules constitute a new abstraction level, which provides foundations for effective adaptive and personalized processes, such as in, e.g. learning. This abstraction level is also highly valuable for adaptation. Rules can be different for different users or for the same user at different periods of time. Meta-Mender is a recommendation system that implements this approach.

At present, we use Meta-Mender in two R&D projects. In both, the next step is an evaluation phase with real data from actual users (in fact, two different, separate target groups). These two evaluation processes will provide a first-hand feedback of the implementation of Meta-Mender. Out of these results, we will refine the engine and we will design a visual authoring tool for meta-rules.

Acknowledgments

The research presented in this paper has been partially supported by the following projects of the Plan Avanza, a Spanish, nationally funded R&D programme: FLEXO (www.ines.org.es/flexo, TSI-020301-2009-9), GAMETEL (www.ines.org.es/gametel, TSI-020110-2009-170), TELMA (www.ines.org.es/telma, TSI-020110-2009-85).

References

[Abel *et al.*, 2008] F. Abel, I.I. Bittencourt, N. Henze, D. Krause and J. Vassileva. A Rule-Based Recommender System for Online Discussion Forums. *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Hannover, Germany, 2008.

[Anderson *et al.*, 2003] Michelle Anderson, Marcel Ball, Harold Boley, Stephen Greene, Nancy Howse, Daniel Lemire and Sean McGrath. RACOFI: A Rule-Applying Collaborative Filtering System. *Proceedings of the IEEE/WIC COLA'03*, Halifax, Canada, October, 2003.

[Burgos *et al.*, 2006] D. Burgos, C. Tattersall and R. Koper. Re-purposing existing generic games and simulations for e-learning. *Special issue on Education and pedagogy with Learning objects and Learning designs*. *Computers in Human Behavior*, 2006.

[Burgos *et al.*, 2007] D. Burgos, C. Tattersall and R. Koper. How to represent adaptation in eLearning with IMS Learning Design. *Interactive Learning Environments*, 15(2), 161-170, 2007.

[DROOLS] DROOLS. The Business Logic Integration Platform, http://www.jboss.org/drools.

[El Helou *et al.*, 2009] S. El Helou, C. Salzmann, S. Sire and D. Gillet. The 3A contextual ranking system: simultaneously recommending actors, assets, and group activities. *In Proceedings of the Third ACM Conference on Recommender Systems RecSys '09 ACM*, New York, New York, USA, 373-376, 2009.

[Forgy, 1982] C. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence*, 19, 17–37, 1982.

[Herlocker, 1999] J. L. Herlocker. Position Statement - Explanations in Recommender Systems. *In Proceedings of the CHI' 99 Workshop*, Pittsburgh, USA, 1999.

[McCarthy *et al.*, 1985] J. McCarthy, P. W. Abrahams, D. J. Edwards, T. P. Hart, and M. I. Levin. LISP 1.5 Programmer's Manual, MIT Press, 1985.

[Moreno-Ger *et al.*, 2006] P. Moreno-Ger, I. Martínez-Ortiz, J. Sierra and B. Fernández-Manjón. A Descriptive Markup Approach to Facilitate the Production of e Learning Contents. *In Proceedings of 6th International Conference on Advanced Learning Technologies* (ICALT 2006), 19-21, Kerkrade, The Netherlands, (IEEE Computer Society), 2006.

[Moreno-Ger *et al*., 2008] P. Moreno-Ger, D. Burgos, I. Martínez-Ortiz, J. Sierra and B. Fernández-Manjón. Educational Game Design for Online Education. *Computers in Human Behavior* 24(6), 2530–2540, 2008.

[Rocchio, 1971] J.J. Rocchio. Relevance feedback in information retrieval, in the SMART Retrieval System. *Experiments in Automatic Document Processing*, Englewood Cliffs, NJ. Prentice Hall, Inc., 313-323, 1971.

[Torrente *et al*., 2008] J. Torrente, P. Moreno-Ger, and B. Fernández-Manjón. Learning Models for the Integration of Adaptive Educational Games in Virtual Learning Environments. *In Proceedings of the 3rd International Conference on E-learning and Games*, Nanjing, China. Lecture Notes in Computer Science 5093, 463–474, 2008.