


Learning in Rule-Based Recommendation Systems



SHRINIDHI HUDLI* (WITH ARVIND HUDLI**)

*UCLA. NOW AT APPLE, INC.

**MS RAMAIAH INSTITUTE OF TECHNOLOGY, INDIA

Outline

- ▶ Problem description
- ▶ Rule-based systems
- ▶ Recommendation System approaches
- ▶ Learning
- ▶ Overrides
- ▶ Implementation
- ▶ Discussion

Problem Description

3

- ▶ Learning in rule-based recommendation system
 - ▶ General problem, but specific application to course recommendation systems
- ▶ Each year students choose (elective) courses for their curriculum
 - ▶ Forces in play
 - ▶ Interest – personal
 - ▶ Proposed plan of study
 - ▶ Pre-requisites
 - ▶ Market opportunities
 - ▶ Department requirements
 - ▶ Advisor recommendations
- ▶ Building a course recommendation system that addresses all forces and also learns

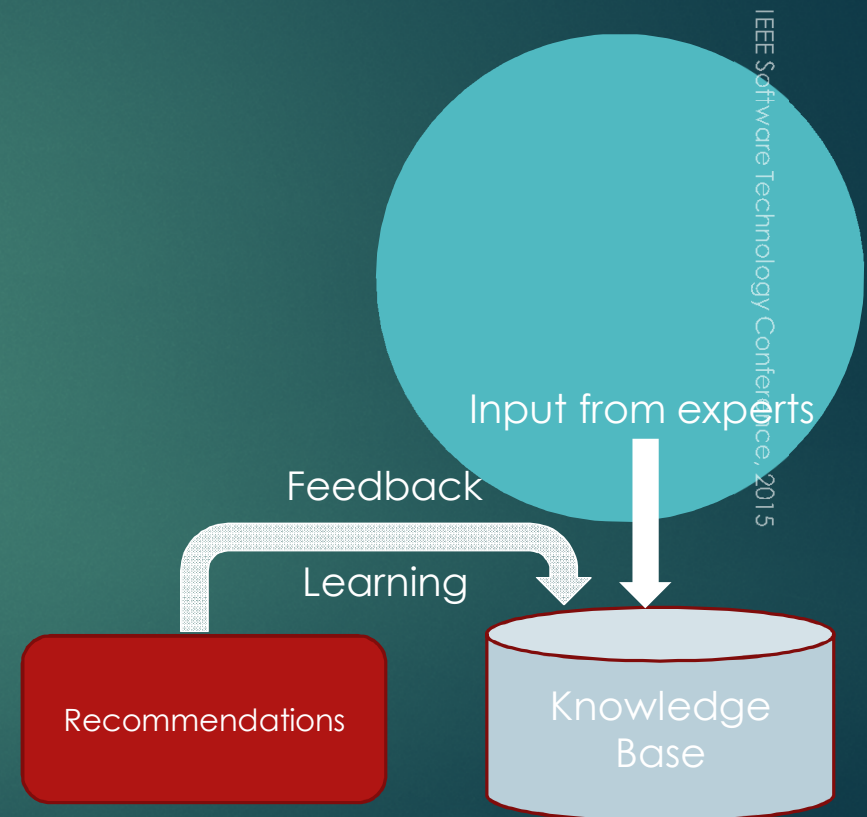
Prior Work

- ▶ Prior course recommendation systems were built on structural relationship between courses and historical data*
 - ▶ Lee and Cho propose a system based on perceived importance of courses, number of electives in department, etc.
 - ▶ T. Denley's Degree Compass, uses students past performance to match degree goals of students
 - ▶ Chu, et.al, propose a data mining and graph theoretic approach to course recommendation
 - ▶ Chen, et.al. describe an approach based on fuzzy item response theory
- ▶ In our view, course selection is not entirely individualistic or autonomous like movie viewing or product purchases, as modeled by previous approaches
 - ▶ Course recommendations are subject to forces mentioned earlier

*References to these in accompanying paper

Solution Approach

- ▶ Use a knowledge-based system that encodes rules addressing forces determining a choice
 - ▶ Build the initial knowledge system based on input from experts
 - ▶ Make course recommendations based on constraints satisfaction
 - ▶ Prioritize and prune recommendations
 - ▶ To make them meaningful
- ▶ Learn based on feedback from recommendations
 - ▶ Reinforce based on positive feedback
 - ▶ Penalize based on negative feedback



Rule-based Systems

- ▶ System infers outcomes – recommendations in our case – based on rules in the knowledge-base
- ▶ Knowledge-base is the heart of rule-based systems
- ▶ Rule-engine's inference process consists of match-select cycle of rules
 - ▶ Built on popular Rete Algorithm
 - ▶ Rules are fired based matched based on constraints
 - ▶ Known facts are matched with rules and chosen in a conflict set
 - ▶ Matching rules are chosen from the conflict set based on priority of rules
 - ▶ We extend standard inference mechanism by making priority dynamic

Rules in Rule-Based systems

- ▶ Rules are of the form
 - ▶ Left Hand Side (LHS) \Rightarrow Right Hand Side (RHS)
 - ▶ LHS is a set of constraints, and RHS is inference if LHS is satisfied
 - ▶ In a course recommendation system, LHS can capture interests, pre-requisites, prior-course grades, conflicts, etc.
- ▶ It is possible that more than one rule is applicable
- ▶ Rule-priorities and conflicts determine which rules will eventually apply
- ▶ In a course-recommendation system, additional rules may have to be applied based on departmental requirements
 - ▶ For example a breadth requirement may force a particular course to be recommended even if some constraints are not satisfied – for example student's interests
- ▶ Our system enhances the inferencing process
 - ▶ Rule-engine match-select
 - ▶ Post-processing match-select based on department's mandatory requirements

Recommendation Systems

- ▶ Recommendations are based on collaborative filtering
 - ▶ A mechanism where inputs from multiple users are used to make predictions for a given user
 - ▶ The user in question's profile and interests are used to filter choices of users with similar profile and interests
- ▶ We further refine and apply item-based collaborative filtering
 - ▶ Recommendation of a given course may imply interest in another related course
 - ▶ We use course groups to either select or eliminate
 - ▶ For example if Software Metrics course is recommended, we will not recommend Quantitative Project Management – if only one course from a group can be chosen

Recommendation System Types

- ▶ Memory-based
 - ▶ User data and actions are remembered and are used to establish collaboration or correlation amongst other users
 - ▶ Similarity measures such as Cosine Similarity, Jaccard coefficient are used to correlate users interests
- ▶ Model-based
 - ▶ More flexible in recommending items a user or user group has never encountered before
 - ▶ Uses probabilistic techniques ala Bayesian Network or Latent Semantic Analysis
- ▶ Association Rules
 - ▶ Recommendations are based on causal relationship between entities
 - ▶ For example a course's pre-requisites and student's prior course work
- ▶ Hybrid approach
 - ▶ We use a hybrid approach and combine memory-based and association rules in our recommendation subsystem

Learning

10

- ▶ In our context of rule-based recommendation system, we use reinforcement learning
 - ▶ Reinforcement learning dynamically changes rule firing priorities
 - ▶ Useful recommendations improve probability of similar future recommendations
 - ▶ Not so useful recommendations – based on user feedback – reduce probability of similar future recommendations
- ▶ The system learns to avoid unnecessary biases to smoothen out probability adjustments
 - ▶ Penalties and rewards change slowly and consider a wider user base

Overrides

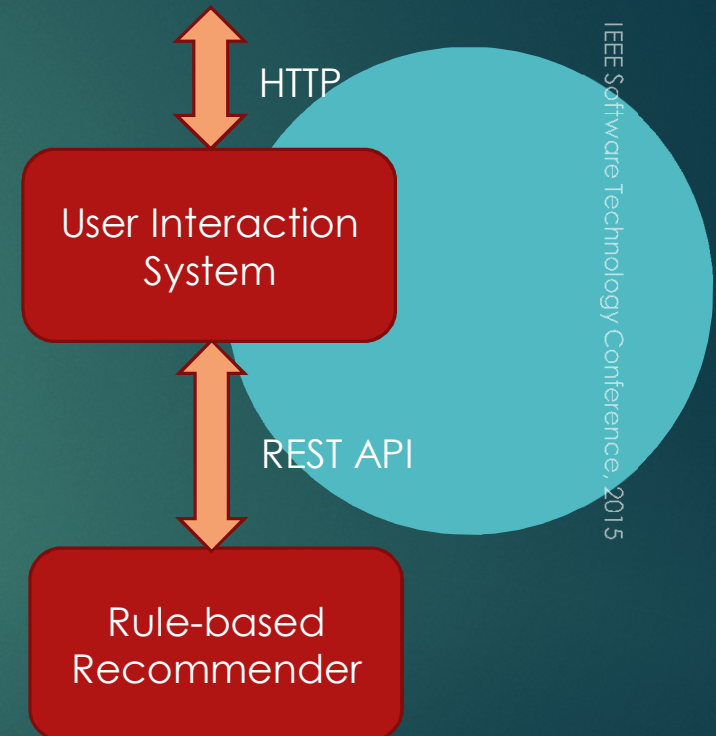
11

- ▶ The system must support overrides from experts and departments
- ▶ For example the department may want students to “pick” courses like Java Programming or Big Data
 - ▶ Based on industry trend
- ▶ User interests or other factors influencing recommendations need to be overridden if such courses are not chosen!
- ▶ We add a post-processing recommender to override recommendations if necessary
 - ▶ Override rules may be written to resolve conflicts and eliminate collaborative filter based recommendation

Implementation


12

- ▶ Web-based Student Course Recommendation System
 - ▶ Separation of user system from recommendation system
 - ▶ User Interaction System built with Python/Flask
 - ▶ Rule-based recommender is built with a REST endpoint, allowing it to be used in different contexts (if needed)
 - ▶ Built in Ruby and a 3rd party rule engine



Rule Engine Choices

13

- ▶ CLIPS
 - ▶ Classic rule engine, first implemented by NASA in 1985
 - ▶ Implemented in C
 - ▶ Not well-suited for web applications
- ▶ PyCLIPS
 - ▶ Python wrapper on CLIPS. Did not choose this either as it runs on Python 2.6 and our web component does not run on Python 2.6
- ▶ JBOSS Drools
 - ▶ Very popular rule engine. DROOLS rules need to be compiled to Java each time there is a change in rule. Less dynamic
- ▶ Ruleby ☒ 
 - ▶ Full functionality of DROOLS and CLIPS and implemented in Ruby. Has a DSL syntax which allows new rules to be dynamically created

Rule Structure and Priorities

14

- ▶ Rules have two parts
 - ▶ Antecedent
 - ▶ Conditions that must be satisfied to be selected. Conditions may be based on both user profile and user input. Certain courses may be offered only for certain majors or students with certain interests
 - ▶ Consequence
 - ▶ Actions that will be performed if the antecedent is satisfied
- ▶ Priority that determines when a rule will be fired
 - ▶ Priorities are generally static, but in our case they are dynamically determined and adjusted based on user feedback
- ▶ Expert or override rules have the lowest priority, always 0
 - ▶ The rules will apply at last and have effect only if overrides are needed

User Input

15

- ▶ User input comes in two forms
 - ▶ User interests
 - ▶ General area interest - high interest in programming languages , low interest in theory
 - ▶ Interest in related and pre-requisite courses
 - ▶ User feedback
 - ▶ User feedback for each course recommendation. This input is used for reinforcement learning

Course Recommendations

16

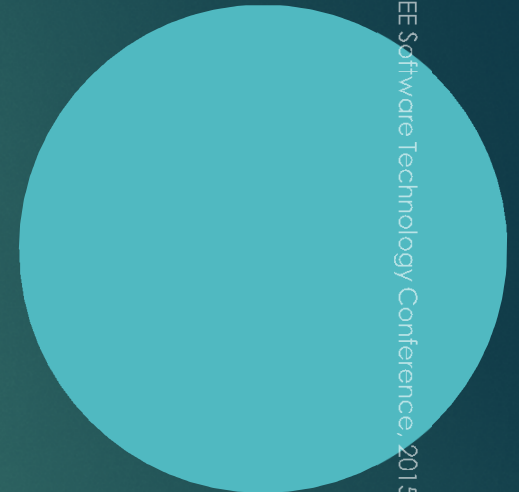
- ▶ Courses may have relations amongst them, such as mutually exclusive group
 - ▶ At most only one course in the group would be recommended
 - ▶ Example: Java Programming, C# Programming
- ▶ Certain courses may be offered only for students at certain level
 - ▶ Example - graduate courses, freshman courses, etc.
- ▶ Course recommendations are grouped based on course groups

Sample Rule

17

```
rule :Java,  
  {:priority => priority("CS516","1","Java")},  
  [Student, :student, m.semester == 5] do |context|  
    s = context[:student]  
    (grade,interest) = s.pre_req("CS315")  
    interests = Array.new  
    interests << s.interest("programmming")  
    interests << s.interest("application")  
    area_interests = interests.max  
    if (interest < 7 || area_interests < 50)  
      skip context[:student]  
    else  
      if !$recommendedCourses.key?(s.name)  
        $recommendedCourses[s.name] = Array.new  
      end  
      elective = Hash.new  
      elective["group"] = "1"  
      elective["code"] = "CS514"  
      elective["name"] = "Java"  
      $recommendedCourses[s.name] << elective if s.interested?("CS516")  
    end  
  end  
end
```

IEEE Software Technology Conference, 2015



Override Rule

18

```
rule :ExpertRecomendation74, {:priority => 0},  
  [Student, :student, m.semester == 7] do |context|  
  s = context[:student]  
  courses = $recommendedCourses[s.name]  
  courses = Array.new if courses == nil  
  already_recommended = false  
  courses.each {|c|  
    already_recommended = true if c["code"] == "CS715"  
  }  
  if already_recommended == false  
    elective = Hash.new  
    elective["group"] = "4"  
    elective["code"] = "CS715"  
    elective["name"] = "Data Mining"  
    s.add_expert_recommendation(elective)  
  end  
end
```


Learning-based adjustment of priorities

- ▶ Rules are fired based on priorities
- ▶ Priorities are adjusted based on reinforcement learning
- ▶ Positive feedback increases priority of corresponding rule
- ▶ Negative feedback decreases priority of corresponding rule
 - ▶ Rewards and penalties can be adjusted with different coefficients

Results

20

- ▶ The course recommendation system was used for undergraduate courses at Visvesvaraya Technological University
- ▶ Used for students in undergraduate computer science curriculum
 - ▶ 600 students
- ▶ Initial results were very encouraging
 - ▶ Further study needed to determine effectiveness of recommendations

Discussion

21

- ▶ Reinforcement learning in a rule-based course recommendation system
 - ▶ Recommender is a hybrid recommender using memory-based and association rules methods
 - ▶ Rule engine has dynamic priority of rules, allowing learning to influence rule matches and selection
 - ▶ Expert rules to override recommendations if necessary, with a static priority of 0, always fired last
- ▶ Future work will incorporate dynamic addition of rules either by added by the user or generated by the system while learning