Learning in Rule-Based Recommendation Systems

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

Problem Description
- Learning in rule-based recommendation system
- Each year students choose (elective) courses for their curriculum
- Forces in play
  - Interest
  - Proposed plan of study
  - Pre-requisites
  - Market opportunities
  - Department requirements
  - Academic opportunities
- Building a course recommendation system that addresses all forces and does learn

IEEE Software Technology Conference, 2015
Prior Work

- Prior course recommendation systems were built on structural relationship between courses and historical data.
- We see Dasgupta propose a framework based on perceived importance of course, number of electives in department, etc.
- T. Denley’s Degree Compass uses student past performance to match degree goals of students.
- Chu, et al. propose a data mining and graph theoretic approach to course recommendation.
- Shen, 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.

Solution Approach

- Use a knowledge-based system that encodes rules addressing forces determining choices.
- Build the initial knowledge system based on expert input.
- Make course recommendations based on constraints satisfaction.
- Review and prune recommendations.
- Learn based on feedback from recommendations.

Knowledge Base

- Knowledge base is the heart of rule-based systems.
- Rule engine’s inference process consists of match-select-cycle of rules.
- System infers outcomes — recommendations in our case — based on rules in the knowledge-base.
- Knowledge base is the heart of rule-based systems.
- Knowledge base is the heart of rule-based systems.
- Rules are fed based matched based on constraints.
- Several rules are matched with rules and chaining conflict.
- We extend standard inference mechanism by making priority dynamic.

Rule-based Systems

- We extend standard inference mechanism by making priority dynamic.
Rules in Rule-Based systems

- Rules are of the form
  - left-hand side (LHS) \( \text{if } \) right-hand side (RHS)
  - LHS is a conjunction, and RHS is a disjunction
  - It is possible that more than one rule is applicable

Rules and conflict detection, which are all eventually applied

In a course recommendation system, additional rules may have to be applied depending on departmental requirements.

Our system enhances the inferring process

Rule-engine match

Post-processing match based on departmental 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.
  - User profile and interests are used to filter choices of users with similar profile and interests.

We further refined and apply item-based collaborative filtering

- Recommendation of a given course may depend on other courses.

We use course groups to either select or eliminate.

- If a course's prerequisite is not met, the course is not recommended.

Association rules

- Recommendations are based on causal relationships between entities.

Hybrid approach

- We use a hybrid approach and combine memory-based and association rules in our recommendation system.
Learning in Rule-based Recommendation Systems

**Learning**
- In our context of rule-based recommendation system, we use reinforcement learning.
- Development scoring dynamically changes rule firing priorities.
- Good recommendations improve probability of similarity recommendation.
- Not so useful recommendations - based on user feedback - reduce probability of similar future recommendations.
- The system learns to add an advisory bias to smoothen out probability adjustments.
- Penalties and rewards change slowly and consider a wider user base.

**Overrides**
- The system must support overrides from experts and departments.
- For example, the department may want students to "pick" courses like Java Programming & SQL.
- User interests or other factors influencing recommendations need to be overridden, such as content not chosen.
- We add a post-processing recommender to override recommendations if necessary.
- Overrules also may relate to market conflicts and alleviate counteracting rule-based recommendations.

**Implementation**
- Web-based Student Course Recommendation System.
- Separation of user system from recommendation system.
- Recommendation system built with Python/Flask.
- Baseline 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**

- **CLIPS**
  - Classic rule engine, implemented by NASA in 1985
  - Implemented in C
  - Not well suited for web applications
  - Python wrapper in CLIPS. Did not choose this as it is more overhead.

- **PyCLIPS**
  - A Python wrapper on CLIPS. Did not choose this as it is more overhead.
  - Implemented in Python 2.4

- **JBoss Drools**
  - Very popular rule engine.
  - Rules need to be compiled each time there is a change.

- **Ruleby**
  - Full functionality of Drools, and CLIPS and implemented in Ruby.

**Rule Structure and Priorities**

- Rules have two parts:
  - Antecedent
    - Conditions that must be satisfied to be selected. Conditions may be based on 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 evaluated based on user feedback.
  - Expert or override rules have the lowest priority, always 0.
  - The user override rule and have effect only if overrides are needed.

**User Input**

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

- Courses may have relations among 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 of certain level
- Example: graduate courses, freshman courses, etc.
- Course recommendations are grouped based on course groups.

Sample Rule

```
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("programming")

interests << s.interest("application")

area_interests = interests.max

if (interest < 7 || area_interests < 50)

skip context[\:student]

else

if !$recommendedCourses.key?

$recommendedCourses[\:student] = Array.new

end

elective = Hash.new

elective[\"group"] = "1"

elective[\"code"] = "CS514"

elective[\"name"] = "Java"

$recommendedCourses[\:student] << elective if s.interested("CS516")

end
```

Override Rule

```
rule :ExpertRecommendation74,

\{priority => 0,

\[Student, :student,

m.semester == 7\] do |context|

s = context[\:student]

courses = $recommendedCourses[\:student]

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