Crossing the Rubicon for An Intelligent Advisor

Răzvan Andonie
Computer Science Department
Central Washington University, Ellensburg, USA
andonie@cwu.edu

J. Edward Russo
Johnson Graduate School of Management
Cornell University, Ithaca, USA
jer9@cornell.edu

Rishi Dean
Sloan School of Management
Massachusetts Institute of Technology, USA
rdean@mit.edu

ABSTRACT

Recommender systems (RS) are being used by an increasing number of e-commerce sites to help consumers find products to purchase. We define here the features that may characterize an "intelligent" RS, based on behavioral science, data mining, and computational intelligence concepts. We present our conclusions from building the WiseUncle Inc. RS, named Rubicon, and give its general description. Rather than being an advisor for a particular application, Rubicon is a generic RS, a platform for generating application specific advisors.

Keywords

Recommender systems, electronic commerce, user interface, user modeling

INTRODUCTION

E-commerce sites use RS to guide potential consumers through the buying process by providing customized information and product recommendations. Based on the customers' individual needs, values, and preferences, the goal of a RS is to find the "best" possible product from a large set of complex options. We shall only mention some online recommender systems that have been used, or are being considered for use: [4, 5, 3, 10]. There are several well-known e-commerce businesses that use, or have used, RS technology in their web sites: Amazon, Travelocity, BMW, MovieFinder, and Dell among them.

Although commercial RS have been available for several years now, we are still at the beginning of using RS on a large scale. In reality, sellers provide an RS to help improve the (long-term) business relationship. This goal gives rise to several desiderata that can be difficult to achieve. The RS should be flexible, scalable, multifunctional, adaptive, and able to solve complex search and decision problems.

The RS interface with the customer should be based on the same consumer psychology knowledge and strategies used in marketing. Behind this "visible" task, a RS can bring valuable information to marketers, making them improve their offer and products (customer profiling, marketing segmentation). For instance, RS can help businesses decide to whom to send a customized offer or promotion.

RS use knowledge to guide consumers through the often overwhelming task of locating suitable products. This knowledge may come from experts (e.g. marketing, product domain) or it can be "mined" knowledge learned from the behavior of other consumers. These two types of knowledge can be used not only during the recommendation process, but also to adaptively improve the system itself.

When optimizing the recommendation, it is possible that the system has to search in a huge admissible solution space for the "best" product. Solving such an optimization during the course of an Internet interaction creates a difficult problem, one that requires devising fast heuristic solutions.

Another knotty problem is related to the formal definition of the "best" recommendation. The optimality of the recommended option generally requires several criteria. The question is how to quantify the importance of such different attributes like color, shape, speed, price, etc?

These considerations and others make us define an *intelligent advisor (IA)* as a RS having the following features:

a.) During each interaction with a customer, it extracts knowledge from the customer that is used to build and update the corresponding customer profile. When the interaction is concluded, the system makes a valid recommendation.

b.) The IA saves the extracted knowledge and the customer

profile can be further "mined" for marketing-relevant knowledge.

c.) The IA - customer interface is based on the psychology of the consumer and the purchase decision process. Therefore, behavioral science techniques should create the fundamentals of an IA, in particular a customer dialog that embraces what/how people think, rather than forcing consumers to feed an optimization algorithm. Thus the IA divorces users from some of the complexity of their decisions.

d.) The IA should be able to improve its functionality by continually learning from its interactions with consumers.

e.) An IA should be robust in the face of data that are uncertain, noisy, sparse, or missing.

f.) An IA should be scalable and able to work in real-time, to meet the requirements of an Internet application.

g.) An IA should know how to draw multiobjective comparisons among products.

h.) An IA should be largely domain-independent, such that with minimum modification effort, one should be able to customize the same platform for other applications (e.g., selling computers, cars, financial services).

How far are current RS from an ideal IA? Some of the existing RS already incorporate some of these requirements. For instance, TalkMine uses the behavior of its users to adapt the knowledge stored in information sources [8]. However, most probably, none of the commercial available RS fulfills all requirements.

The idea of defining an IA came after several years of building a commercial RS, called Rubicon. Rather than being an advisor for a particular application, Rubicon is a generic RS, a platform for generating application-specific advisors. In this paper we look at the difficulties we faced when building Rubicon and the main concepts we used.

DIFFICULTIES IN BUILDING A RS

There are two categories of problems we faced when building our RS. The first is related to the system design, the second to the customer behavior.

Design and Integration

Incorporating complex behavioral data. There are many types of information that can be collected and used: customer knowledge (goals, needs and profile of the user), domain knowledge (product information and business rules specific to a particular vertical application), traffic logs, and expert knowledge. Using expert knowledge alone we can recommend "good" products. Using only customer knowledge, we can recommend products that were sold successfully in the past. The first strategy is much better at dealing with new products, whereas the latter one reflects only the customer experience.

Scalability and real-time performance. Scalability in recommender systems includes both very large problem sizes and real-time latency requirements. For instance, a recommender system connected to a large web site must produce each recommendation within a few tens of milliseconds while serving thousands of consumers simultaneously and searching through potentially billions of possible product configurations.

Noisy, missing, uncertain, and sparse data. The value of a RS lies in the fact that most customers have not deeply considered many of the available products, the product features, or their personal needs. This means that we must often deal with extremely sparse data, such as that resulting from a customer responding "I don't know", "I don't care", or "I don't want to answer this question".

Connecting recommenders to marketers. RS should be connected to the vendor's product database and the marketer's reporting systems. Only products that are currently in stock should be recommended, or products that can be configured in a feasible manner, both from engineering and logical perspectives. The highly volatile nature of real-world products and information systems creates the necessity of adequate database maintenance in the IA.

Domain independence. From the software engineering point of view, building a domain-independent RS platform can be done by separating the generic part from the domain specific knowledge modules.

User Experience

The customer-recommender interface is usually based on a series of interactive questions presented to the customer by the RS, accompanied by multiple-choice options for the customer to input their answers. In this case, a difficult problem is what strategy to follow when selecting questions to present. An intelligent dialog should be personalized. Some randomness should be used when selecting the questions [7]. Too much randomness leads to excessive customer effort, but a small amount of randomness may help to extend the space over which the recommender understands the customer's interests and ensures that all questions are occasionally presented to customers. A reasonable strategy for selecting information from customers is to minimize customer effort while still being able to make accurate predictions [7]. However, this strategy is quite simplistic, and a behavioral science-based investigation is necessary here. What we should measure is not customer effort (measured in the duration of the dialog), but customer satisfaction. Satisfaction quantification results from longer-term statistics on usage and surveying customers.

During the conversation, an IA adopts a five-stage process, described by [2]: **i)** Opening; **ii)** Utilitarian Needs; **iii)** Hedonic Preferences; **iv)** Optional Features / Add-ons; and **v)** Endgame.

Stage **i** frames the buyer (e.g., knowledge of the product category and extent of product search to date) and the main product characteristics (e.g., a desktop PC versus a laptop).

Stages **ii** and **iii** encompass, respectively, the utilitarian and hedonic or emotional needs. The former include the functional uses of the product, such as an automobile's seating capacity or environmental friendliness. Stage **iii**'s hedonic needs, like the image of a car's body style and brand name, are often harder for a buyer to express. Needless to say, extracting such knowledge can be a substantial behavioral challenge in itself. Stage **iv** captures the remaining, minor product specifications, like an automobile's audio speakers or aspects of its interior. The final stage covers such external elements as a PC's warranty or the local availability of reliable repair service for an automobile. These five stages are sufficient to structure the process of a purchase decision for all complex products.

How can a trusted recommender validate itself to consumers through a web client? The following factors contribute to the success of such conversations in Internet-mediated dialogs [9].

The benefits of the conversation should exceed its costs. People use information only if it is perceived as adding benefits or as reducing costs. If (expected) costs exceed (expected) benefits at any point, there is a clear risk of the customer terminating the dialog.

Credibility and trust. The information and advice must be credible, and the source must be trustworthy. An Internet-delivered RS cannot provide the face-to-face cues of trustworthiness that a human can. However, although a RS may have no initial reputation for trust (based on past experience), such an image can be built over time by personal usage, word-of-mouth recommendations, or public endorsements (e.g., by consumer-oriented magazines' endorsement of the system's knowledge and disinterestedness). One alternative is to add a confidence metric, and this has the potential to improve user satisfaction and alter user behavior in the RS [6]. A second alternative is to make the RS adaptive. This would reduce the risk of manipulation: users can detect systems that manipulate prediction and, this has a negative impact on their trust [1].

Intelligence and customization. First, the advisor must know what kinds of information people can validly provide and how to successfully extract that information from buyers. Consumers can usually say what they need or want the product to do and can articulate such personal preferences as style and color. However, they may have difficulty specifying the product features that meet those needs. Second, based on whatever can be learned from the customer, the problem of identifying the optimal product must be solved. Thus, the advisor must first extract the customer's needs and then build an inferential bridge from those needs to the most suitable product.

Control. Customers should be able to request additional or explanatory information. Or as the conversation proceeds, the customer may learn something that requires returning to an earlier point in the dialog and changing a preference stated there. Buyers who feel impatient should be able to request a recommendation at any time, even before the advisor would normally feel comfortable providing one. Finally, the buyer might even like to suspend the conversation and return later. More control in any situation is empowering, and more so in situations where control is expected. Providing satisfactory conversational control is a special challenge to RS.

Feedback. Specific feedback might include (a) how much progress has been made toward identifying the best product, and (b) how much longer the conversation is expected to take. Whatever specific feedback options are provided, however, users do not want to receive feedback only after they have answered every question (as they must in many static surveys).

SYSTEM DESCRIPTION

Rubicon is a generic domain-independent advisor, recommending products from an existing set. Each product is configurable, meaning it is comprised of several components, which may each be described in turn by several attributes. Building a RS depends largely on the knowledge representation model, and we chose a computational intelligence framework. Our RS is a classifier that "learns" to make good recommendations. This classifier is an expert system, able to explicitly expose its acquired knowledge. The main characteristics of Rubicon are:

- The inferential process from the customer's needs to the best product is constructed in two stages, called *Bridges*, one from needs to product attributes, and the second from attributes to the products themselves.

- It can easily be customized for different applications since the interface to the application-specific knowledge domain is separated from the main system.

- The front-end dialog is dynamic and changes based on user responses. It elicits information from users to determine a location in a needs space which is then used to find optimal (sub-optimal) solutions in a products space.

- It accepts imprecise input from users.

- It provides a justification for all recommendations.

- Reversibility: The system can reverse the decision process from effect to cause. This allows forecasting the adoption of new products or services using real customer decision data.

The Rubicon system diagram (Fig. 1) shows the following main modules.

Conversation Engine (CE)

The CE is responsible for dialog management, presenting questions to the user and processing the resulting responses
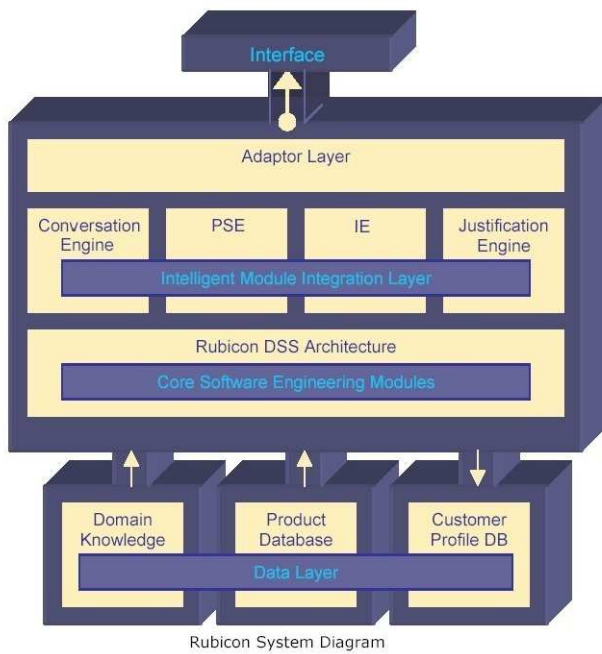
Figure 1: Rubicon High Level System Description

sent to it by the user. Questions and their associated responses are processed to accomplish the following two results: *i)* Propagate the knowledge gained from a response to the subsequent inference mechanisms and *ii)* Determine the next question to pose to the user.

In doing so, the dialog management occurs subject to the following constraints:

- Presents the appropriate questions for the system to confidently determine an intelligent, personalized recommendation.

- Presents questions conforming to users' expectation of a real dialog with respect to flow, organization, and coherence.

- Minimizes the number of questions presented.

- Scalable addition/ subtraction/ modification of questions.

- Allows users and administrators to reproduce particular dialogs.

- Uses proper constructs for further data mining.

From a computational standpoint a rule-based expert system is used to implement the CE's dialog management process. Questions and responses are linked by sets of predetermined rules, and a number of other intermediary constructs. In this way, the questions, responses, and rules can be specified, along with goals (i.e., knowledge to be gathered) independently of knowing the dialog flow in advance. The

system at runtime determines, based on the behavioral and informational goals, which question to present next to the user. When all appropriate questions have been presented, the conversation is determined to be complete. However, the user may intervene at any time to ask for the system's current best recommendation based on the information provided thus far.

Inference Engine (IE)

The purpose of the IE is to map the user's profile of needs (the output of the CE) to the attributes necessary to comprise the appropriate recommendation. Given a set of responses resulting from the dialog, the IE can indicate a set of recommendations, ordered by the degree of their preference. These recommendations are not concrete (physical) product recommendations yet, but a mapping from the user needs space to the space of attributes, yielding generic descriptions of the product, like "RAM Amount" (e.g., standard, large, maximum) and "Network Card type". Collectively this inference is called the *First Bridge*.

The IE is taught by a human expert. However, it can learn incrementally as well: new teaching examples can be added without restarting the teaching process from the beginning. Conditional rules can be extracted to describe the behavior of the IE and justify recommendations, market research and performance improvement. The IE is stable under noisy inputs and user uncertainty. Such "noise" may be produced by "I don't know" answers, or by contradictory answers in the dialog.

To implement the IE, a fuzzy neural net architecture is used, trained to represent the expert knowledge of a particular product domain. For instance, in the case of a personal computer RS, experts develop training patterns to represent the varying needs profiles of customers along with their corresponding feature sets for a recommended PC. The inference process is fast, online.

Product Search Engine (PSE)

The PSE is the *Second Bridge*, a mapping from the space of attributes to the space of (physical) products. It is an optimization module interfacing with the retailer's product database to select the best, valid product configurations that match the criteria specified by the user, such as the minimum cost, the maximum likelihood of success, or a number of other simultaneous criteria. The inputs to the PSE are the levels of the attributes (the output of the IE), the configuration constraints (i.e., incompatibilities among components), and a user's criteria for optimization (e.g., a desired price point). These criteria for the algorithm can be set by the IE and CE and are, therefore, uniquely tailored to a given user. The PSE can sort though billions of options in real time, allowing searches to be completed online. The products with the highest degree of fit are passed to the Justification Engine for further processing.

The response to a question is subsequently used to provide

more information that adds to Rubicon's knowledge base during a user-experience. This, in turn, leads to a recalculation and optimal selection of the next most appropriate question. This question-response model continues until Rubicon is either asked, or has sufficient confidence, to make a recommendation.

The PSE navigates a vast search space, taking into account different optimization criteria. We used a genetic algorithm approach for this (NP-complete) optimization problem. In the initial phases, the PSE operates on abstractions of the real world, and then through an adaptor layer translates these abstractions into concrete items. This information is capable of being read and processed at runtime. The PSE remains independent of constant updating of the "real world" items. This adaptor level is implemented as an XML data bridge.

Justification Engine (JE)

Recommendations are run through the JE to provide a plain English explanation of why the system has provided a specific recommendation. This justification is delivered in the same vernacular as the dialog, personalized to the user, and is present to facilitate user understanding and adoption of the recommendation. The JE takes the set of If-Then rules from the IE and the set of recommended configurations from the PSE and develops a rationale for selecting each product. The value of the JE is that it creates confidence in the recommendation.

Rubicon is implemented using a complimentary modular software approach that encapsulates the individual computational blocks, as well as the necessary software architecture emphasizing a stable and reusable model that is compliant with the J2EE technology standard. The user interface is HTML 4.0 compliant, utilizing DHTML, and combining client-side scripting and styles. It is implemented using XML/XSLT, built for 4th generation web browsers, and rendered via the adapter layer using JSP/servlets.

PRELIMINARY TESTS

Although still under development, Rubicon was sufficiently developed to be submitted to usability testing by two major PC manufacturers. Each test involved about a dozen users and compared three RS. One was the manufacturer's current online RS, one was an attractive competitor, while the third was Rubicon. The results made available revealed that Rubicon was judged clearly superior in both tests. For instance, in one test, when asked which of the three RS the user would "be most likely to use again", nine of eleven respondents chose Rubicon.

Rubicon was tested online by a webhosting services provider. Of 2200 online users who began a conversation, 83% completed it to the point of receiving a recommendation (which was the only result made available to us). This was judged by the host company to be an extraordinary high completion (i.e., non-abandonment) rate.

CONCLUSIONS

Is Rubicon "intelligent"? According to our IA criteria from the Introduction, yes. Furthermore, we believe that these criteria may be a base for classifying RS. We have not presented here other modules of Rubicon, used for prediction, customer profiling, and marketing segmentation, since we have tried to focus on the core system. It was a challenging task to build Rubicon, especially because of its generic character. Making the system largely independent of a specific e-commerce application required greater complexity and abstraction. But do we really need a generic RS? From a user perspective this may be a non-issue. However, for the RS designer and software engineer this is a critical requirement. We should think not only in terms of how to use a RS, but also how to build it and how to adapt it fast for very different application areas.

REFERENCES

1. D. Cosley, S. Lam, I. Albert, J. Konstan, and J. Riedl. Is seeing believing? how recommender systems influence users' opinions. In *Proceedings of CHI 2003 Conference on Human Factors in Computing Systems*, pages 585–592, Fort Lauderdale, FL, 2003.

2. A. Horowitz and J. Russo. Modeling new car customer-salesperson interaction for a knowledge-based system. *Advances in Consumer Research*, 16:392–398, 1989.

3. S.-Y. Hwang, W.-C. Hsiung, and W.-S. Yang. A prototype www literature recommendation system for digital libraries. *Online Information Review*, 27:169–182, 2003.

4. W. Lin, S. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.

5. J. Liu and J. You. Smart shopper: An agent-based web-mining approach to Internet shopping. *IEEE Transactions on Fuzzy Systems*, 11:226–237, 2003.

6. S. McNee, S. Lam, C. Guetzlaff, J. Konstan, and J. Riedl. Confidence displays and training in recommender systems. In *Proceedings of INTERACT '03 IFIP TC13 International Conference on Human-Computer Interaction*, pages 176–183, 2003.

7. A. M. Rashid, I. Albert, D. Cosley, S. Lam, S. McNee, J. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pages 127–134, San Francisco, 2002.

8. L. M. Rocha. Talkmine: a soft computing approach to adaptive knowledge recommendation. In V. Loia and S. Sessa, editors, *Soft Computing Agents: New Trends for Designing Autonomous Systems - Studies in*

Fuzziness and Soft Computing, pages 89–116. Physica-Verlag, Springer, 2001.

9. J. E. Russo. Aiding purchase decisions on the Internet. In *Proceedings of the Winter 2002 SSGRR (Scuola Superiore Guglielmo Reiss Romoli) International Conference on Advances in Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet*, L'Aquila, Italy, 2002.

10. H.-W. Tung and V.-W. Soo. A personalized restaurant recommender agent for mobile e-service. In *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, pages 259–262, 2004.

APPENDIX: THE USER EXPERIENCE WITH THE RUBICON COMPUTER ADVISOR

Instantiated as a computer advisor, Rubicon shows (Fig. 2) its first screen to a customer. Buyer control is offered, inter alia, by the "Why Ask" button, which pops up a justification for the current question. This feedback also exhibits respect for the buyer's right to know why their time and effort are being spent answering this particular question.

Figure 2: Getting acquainted

Ten questions into the conversation, just as it moves to Stage 2, the screen from Fig. 3 appears.

The Progress box on the right now shows a current recommendation in the form of the best type of computer for the particular buyer and the two most suitable alternative types. For each of the three types, a measure of fit to the buyer's needs is displayed. This measure should increase as more needs are elicited through further questioning and the recommended product is further customized to those needs. Below the recommendation is an announcement about the number of facts known, either directly or inferred. And below that is an indicator of how far through the conversation the system expects the buyer is. The measure is not time, but the number of questions already asked and the expected number to be asked before the conversation is finished.

After this screen the next question is "Do you have any problems with your current computer?" If the buyer answers

Figure 3: Extract the customer's needs

"Yes" and identifies "Too Slow" as the only problem, then the screen shown in Fig. 4 appears.

Figure 4: Continue extracting the needs

Note that the fit of the Hobbyist PC to the buyer's needs has moved from 40% to 53% and the number of known facts from 14 to 21. Both changes provide feedback and show the benefit of participating in the dialog, as it progresses toward the best recommendation. If the conversation reaches its natural conclusion (i.e., is not terminated prematurely by the buyer), three products are recommended, in order. This is another illustration of buyer control. People prefer to make the final choice themselves from several options, although they also want to know the advisor's ranking. A demo version of Rubicon can be found at www.wiseuncle.com.