Preference-Based Interfaces for CRPGs

André Matias, Carlos Martinho

Instituto Superior Técnico; andre.matias@ist.utl.pt

Abstract

This work proposes the application of preference handling techniques, in particular preference elicitation, to build an item recommender system to be integrated in the context of a computer role-playing game. This system will be used by in-game vendors, allowing them to learn the player's preferences and then recommending the items more adequate to the player. This paper introduces the problem caused by lack of customized support during vendor interactions, describes how some existing games tackle it, and then addresses the state of the art in preference modeling and preference elicitation techniques necessary to adequately build an item recommender system. Finally, it describes the main ideas behind the architecture of the item recommender system that will be developed.

Keywords: role-playing games, preference elicitation, artificial intelligence

Resumo

Este trabalho propõe a aplicação de técnicas de manipulação de preferências, em particular de extracção de preferências, na construção de um sistema de recomendação de itens a ser integrado no contexto de um *computer role-playing game*. Este sistema será usado por vendedores no contexto do jogo, permitindo que estes aprendam as preferências do jogador e recomendem os itens mais adequados ao jogador. Este documento introduz o problema causado pela falta de suporte personalizado durante interacções com vendedores, descreve como alguns jogos lidam com ele, e foca-se de seguida no estado da arte relacionado com modelação de preferências e técnicas de extracção de preferências, necessário para construir um sistema de recomendação de itens adequado. Por fim, descreve as principais ideias da arquitectura do sistema de recomendações que será desenvolvido.

Palavras-chave: *role-playing games*, extracção de preferências, inteligência artificial

Introduction

In the world of computer role-playing games (CRPG), the player typically takes control of a virtual avatar and engages in exploring fantasy worlds, interacting with a diverse cast of non-player characters (NPC), battling enemies, completing assigned quests and "building" the avatar in a number of possible ways. For this purpose, the player often has the need to acquire resources that provide valuable aid throughout the game, generally offered by in-game vendors, for instance blacksmiths.

When interacting with such vendors the player may be presented with a wide plethora of resource items, having little to no clue as to what to choose to satisfy her needs. Also, the currency used to buy these resources is limited and typically requires some work to be obtained. Thus, the number of items the player can buy is limited and, consequently, she needs to carefully consider about which ones will prove to be more useful and adequate to her preferences. The player may then take a considerable amount of time to make a decision and, even then, may make a "wrong" one, not buying the more adequate items and having her game experience suffer significantly later on.

ISBN: 978-989-20-2190-4

There are several ways of helping the player make a decision which are used in existing CRPGs. Probably the most obvious ones are the use of good interfaces and detailed item descriptions. Another possible help consists in organizing all the items in different categories (Eternal Sonata, 2007) or, alternatively, in different vendors (Phantasy Star Online, 2000). The player then knows that vendor or that category only includes weapon type items, for instance. Another type of help provided are recommendations of items that are new in stock (Tales of Vesperia, 2008) or items that are generally considered important at some point in the game.

However, all these general ways of helping may not be enough to assure the player will choose the items more adequate to her style of play and her preferences. How else, then, could we help the player make a decision? This work proposes the integration of an item recommender system in a CRPG, by applying preference handling techniques in such context, in particular preference elicitation, and thus allowing vendors to learn the player's preferences by building her preference model and then give customized recommendations according to such preferences. Additionally, this preference model may also be used to recommend quests and adapt the storyline to the player's preferences, making the game experience more personal and, hopefully, more enjoyable.

Preferences

Preferences over some domain of possible choices allow establishing an order to these choices so that a more desirable choices precedes a less desirable one. These possible choices will be, from now on, referred to as outcomes. There are some properties to be taken into account when working with preferences. Preference orderings may be characterized by at least two important criteria: first, an ordering is *total* if it is possible to compare any pair of outcomes, or *partial* if it is not; second, an ordering is *strong* if no two outcomes are equally preferred, or weak if such restriction is not applied on the ordering. Also, preference relations are transitive, that is, if *A* is preferred over *B* and *B* over *C*, then *A* is preferred over *C*. See Brafman and Domshlak (2009) for a more formal definition of preferences and relevant properties.

While these concepts may seem reasonably easy to grasp, there are some difficulties one can face when working with preferences. Each outcome may have several *attributes* or *aspects* that affect the preferences of a player regarding such outcome. Also note that not all attributes may be relevant to the player; only the *preference-affecting attributes* play a role here. Attribute preferences, in their turn, are not always quantifiable and easy to formalize. There may also arise a need to consider *trade-offs* and *interdependencies* among different attributes. These are all issues that may arise when modeling preferences (Brafman and Domshlak 2009).

When starting our preference modeling task, we should take into account a metamodel with five key elements. First, we have the *model* itself which specifies what we are really interested to know about, that is, what preferences the player has for each one of the available items. As such, the model may be represented as a simple ordering of such items, from most to least preferred or vice-versa. Naturally, the player will not be asked to order all the items according to her preferences; the player will simply make statements about her preferences and interact with the store (buying or selling items) and, as such, elicited preference information will not be stored as an explicit ordering. Instead, we use a tool that stores this information and implicitly specifies the model in a more compact and convenient way. This tool is the *language*. In order to map the language to the model, an *interpretation function* is then needed. Finally, we have the *algorithms* that implement the *queries* we want to be able to ask about the model. Finding the most preferred item or testing if item *A* is more preferred than item *B* are simple examples of such queries. See Brafman and Domshlak (2009) for more on the preference modeling task metamodel.

Returning to the language element of the metamodel, the use of *value functions* as the language is usually a reasonably acceptable choice. Value functions map an outcome to a real value. The interpretation function then dictates, for instance, that the higher the value of an outcome, the more preferred it is. As it was already stated, preferences may be affected by attributes an outcome has, rather than by the outcome itself. Thus, value functions can map an attribute to a real value instead, achieving a more compact representation. We then need a method to compute the real value of an outcome from the real values of its attributes.

A simple approach to compute the real value of an outcome is by using *additivity*, that is, by simply applying a sum operation to all of an outcome's attribute values and then using the returned result as the outcome's real value. Although this may be enough in several applications, there may arise a need to consider preference dependencies among different attributes. These dependencies are not covered by such a language. It’s here where generalized additive independence (GAI) value functions come into play, allowing for a more general form of additive independence. Instead of applying a sum operation to the real value to each attribute's real value, GAI value functions apply a sum operation to *factors*. Each factor is the real value of a set of combined attributes, in order to capture attribute dependencies among them. If such a set always contains a single attribute, then we obtain our first form of additive value functions. Value functions as the language for preference modeling and preferential independence properties are further studied in Brafman and Domshlak (2009) and Chen and Pu (2004).

Preference Elicitation

In order to build a preference model as complete and accurate as possible, we need to use adequate *preference elicitation* techniques. As such, these techniques allow the system to elicit, from the user, the necessary information to build such a model.

There are at least three main difficulties that may arise during preference elicitation that should be immediately taken into account. First, users may focus on *means objectives* rather than *fundamental decision objectives*. Second, users may not be aware of all preferences until they see them violated. Finally, users may state inconsistent preferences, if the system allows it. In order to avoid such difficulties, the system may try to educate the user about which items (and which attributes) are available, making her gain *preference fluency*. A common and effective way to do this is by showing item examples to the user, which is usually implemented in a *example-critiquing* style of interaction. Using such interaction, when the player interacts with the store, a set of examples are immediately shown. From here, the player may apply a critique to one of these items (for instance, asking for "a sword like this, but with fire property"), getting another list of examples in return. The player may then apply another critique to an item on the list and this goes on until the player chooses what she wants to buy. The minimum number of examples to show so that the target choice is included was studied by Faltings et al. (2004). As for what examples to show, there should be a focus on *diversity* and *similarity*. Diversity among examples promotes gain in preference fluency and similarity (to the ideal item according to the user's current preference model) assures the system shows the user examples of her preference. This implies a trade-off that is further studied in Smyth and McClave (2002), and McSherry (2002).

When the user is not able to find an outcome that satisfies all of her preferences, she will then need to choose a partially satisfying one. There may also be too many possibilities to choose from, making the user not certain of what decision should she take. In both mentioned cases, there may be a need for a *preference revision* process. Preference revision allows the user to change one or more of her stated preferences or the degree to which such preferences were stated. See Pu and Chen (2008) for a more in-depth discussion about preference revision.

It is important to note that, in order for a recommender system to be successful, users should be able to trust it and understand it. After all, we won't follow recommendations we don't understand from someone we don't trust. Thus, recommendations should be sufficiently transparent to the user and easy to understand. This may be achieved by using *explanation interfaces* (Pu and Chen 2008), which provide options to display clear explanations for the displayed recommendations. A traditional method for doing this is by inclusion of a "why" option alongside every recommendation, which, when selected, displays an adequate explanation of why that item was recommended to that user. A more recent, and efficient, way of doing this is by using *organization-based* explanation interfaces, which group recommended items into multiple categories, each one labeled with a title explaining its contained items' similar characteristics.

As a last note to preference elicitation, one should know that it is not always possible, or reasonable, to obtain a full preference specification from the user, due to time and effort needed, among other potential reasons. As such, it is important to know how to work with only a partial specification of the user's preferences. This problem may be approached in several possible manners: with a *maximum-likelihood* approach or a *Bayesian* approach, akin to machine learning algorithms (see Brafman and Domshlak 2009); using *regret-based scores* (see Braziunas and Boutilier 2010); or as a *partially observable Markov decision process* (POMDP) problem (see Boutilier 2002, Doshi and Roy 2008).

Architecture

In the context of this work, an item recommender system was developed and integrated in a CRPG, also developed for the sole purpose of this work. The system will use GAI value functions as the chosen language for modeling preferences, in order to deal with preference dependencies. Preference-affecting attributes and relevant dependencies will be defined from the start. The preference model will be updated when the player interacts with the store (buying or selling items) and when she answers to queries posed by the vendor, choosing one of multiple provided choices during natural conversations with the vendor. The interface will implement an example-critiquing style of interaction and provide an explanation to each set of examples displayed. These examples will be displayed in number and diversity as was studied and briefly mentioned in this paper. In order to work with a partial specification, regret-based scores will be used.

The system will be evaluated in several sessions, each involving only one user playing the CRPG. This CRPG will feature two vendors: a traditional one that does not learn the player's preferences and a second one that uses the item recommender system, builds a preference model and makes recommendations accordingly. Each player will be able to interact with only one of these vendors; the purpose here is to evaluate the game experience with each vendor separately and then compare results. Relevant data will be retrieved with log files during the play session and with a written questionnaire and open discussion after the session.

Bibliographic References

Boutilier, C. (2002), A POMDP Formulation of Preference Elicitation Problems, AAAI/IAAI 2002 Conference on Artificial Intelligence, Edmonton, Canada.

Brafman, R., Domshlak, C. (2009), Preference Handling – An Introductory Tutorial, AI Magazine, Volume 30, nº 1.

Braziunas, D., Boutilier, C. (2010), Assessing regret-based preference elicitation with the UTPREF recommendation system, ACM Conference on Electronic Commerce, ACM Press.

Chen, L., Pu, P. (2004), Survey of Preference Elicitation Methods, EPFL Technical Report IC/2004, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

Doshi, F., Roy, N. (2008), The permutable POMDP: fast solutions to POMDPs for preference elicitation, 7th International Joint Conference on Autonomous Agents and Multiagent Systems, IFAAMAS.

Faltings, B., Pu, P., Torrens, M., Viappiani, P. (2004), Designing Example-Critiquing Interaction, Proceedings of the International Conference on Intelligent User Interface (IUI-2004), ACM Press.

McSherry, D. (2002), Diversity-Conscious Retrieval, Lecture Notes in Computer Science, Volume 2416, Springer Berlin, Berlin, Germany.

Pu, P., Chen, L. (2008), User-Involved Preference Elicitation for Product Search and Recommender Systems, AI Magazine, Volume 29, nº 4.

Smyth, B., McClave, P. (2001), Similarity vs. Diversity, Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR-01, Springer-Verlag, Berlin.