The Friendly Blacksmith
Application of Preference Elicitation Techniques to a Computer Game's Synthetic Vendor

André Matias

IST - Instituto Superior Técnico
andre.matias@ist.utl.pt

Abstract Preference handling techniques have been a subject of increasing study, as they prove as useful tools to support decision making, using a preference model as basis. As a part of preference handling, preference elicitation arises here as the key process of obtaining such a model, with maximal accuracy and minimal effort. The main goal of this work is to apply preference handling techniques, with a focus on preference elicitation, to synthetic vendors in computer games, in particular role-playing games, to assist players in making decisions and hopefully enriching the game experience. This document starts with a description of current role-playing game interfaces, exposing the problem caused by the lack of customized support during vendor interactions, followed by an explanation of the concept of preferences and how to work with them, reviewing relevant preference elicitation techniques. The concepts underneath the recommender system developed to test this work's hypothesis are formally explained, and an overview of its implementation is given. The recommender system's evaluation process is then described in detail and collected data is presented, analyzed and discussed, with overall encouraging results. Finally, concluding remarks are presented and several points for future work are suggested.

1 Introduction

In the world of fantasy 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. Weapons and armor for use by the player's avatar are just two of the many types of resources usually seen in any CRPG. One way of acquiring such resources is typically offered with the possibility of interacting with in-game NPCs with the role of vendors, and buying them from an available collection of resources provided by such vendors, through means of fictional in-game currency. From now on, the user (in particular, the player) and the avatar shall be referred to as "she", while the in-game vendor will be treated as "he".

In spite of what was previously stated, it's not uncommon for the player to be presented with a wide plethora of resource items, having little to no clue as to what to choose to satisfy her needs. Additionally, in a way similar to what's observed in real life, the currency used is not unlimited and requires some kind of work to be gained. Thus, in such situations, the player needs some kind of help to make a decision, in order to buy the items that better suit her personal preferences and the in-game situation, assuring as much as possible that the limited currency in her possession is not wasted in less than useful items.

As is the case of Eternal Sonata[1], some CRPGs facilitate interactions with vendors by organizing the items in several categories and providing extensive descriptions for each item, making the decision process somewhat easier for those who may have a hard time finding

[1] Eternal Sonata, 2007, developed by Tri-Crescendo, published by Namco Bandai Games.

and/or choosing what resources they need. A few CRPGs also recommend new-in-stock (for example, Tales of Vesperia[2]) or generally important items to the player. Even with all this help, the player's personal preferences are typically not taken into account in any interaction with the vendors and many players may find it hard to make good decisions regarding which items they should buy in a large collection of different alternatives. Worse, a few players may even make faulty decisions and consequently have their game experience suffer later due to lack of proper resources in a given situation.

Since a player tends to interact repeatedly with such vendors over the full length of the game, such difficulties may significantly hinder the player's experience. Therefore, creating a model which mirrors, as completely and accurately as possible, the player's current preferences, and using it to guide interactions with a vendor, may help greatly improve the speed and efficiency of acquiring needed resources, hopefully making the game more enjoyable.

This preference model is built actively while the player interacts with the vendor and is maintained, even when the player has finished acquiring her resources, for use in later interactions. Choosing a representation for this preference model, constructing a reasoning that allows an acceptable mapping from this model to the real preferences of the player when guiding interactions with the vendor and, mainly, selecting several elicitation techniques to build and update such model, are the core issues of this work.

Thus work focuses mainly on studying state of the art preference elicitation techniques and building an item recommender system that implements relevant techniques and may be integrated into a game. The purpose of this system is to build, store and manage the player's preference model, allowing the game's vendor to provide her with valuable help in making a decision of what to buy or even allowing the storyline or generally offered quests to depend on said model, allowing for a more customized and, hopefully, a more enjoyable game experience.

Such a system was developed and tested, in the context of a CRPG specifically constructed for this work's purpose, and results were collected, analyzed and interpreted. This contribution marks a first approach to integrating a recommender system in a game and will hopefully be used to further progress this work in the future.

The next section will describe in more detail describing how vendor interfaces are currently being constructed in CRPGs, and introduces the reader to the concept of preference handling and, in particular, preference elicitation, briefly describing relevant state of the art techniques to model and handle preferences. Section 3 will describe the conceptual model for the recommender system developed in the context of this work. Section 4 will describe how the recommender system was evaluated, presenting the attained results. Finally, section 5 will present the concluding remarks and suggest several points to be approached as future work.

2 State of the Art

This section gives an introduction on how vendor interfaces are currently being designed in games, presents the concept of preferences and briefly describes how to work with them and, finally, describes several preference elicitation techniques.

2.1 Synthetic Vendor Interfaces in CRPGs

There is a great effort put in making good game interfaces; after all, a bad interface can be reason enough for a player to lose interest in a game. Vendor interfaces require particular care, as they will be used many times throughout the CRPG experience, and will determine how prepared the player is to endure the upcoming challenges. This is because the CRPG genre usually relies on items that are used to equip the player and, consequently, prepare her for the virtual adventure.

[2] Tales of Vesperia, 2008, developed by Namco Tales Studio, published by Namco Bandai.

Many games approach this topic by providing several tools that facilitate the process of finding and acquiring proper equipment. Some games organize the items into categories (Eternal Sonata) within the same vendor, or throughout different vendors, one for each item type (Phantasy Star Online[3]). A few games also make non-personalized recommendations to the player, highlighting one or more items from the set of items listed or giving this information in some other way. Chaos Rings[4], for instance, highlights new items with a "New" label. In other cases, like in Tales of Vesperia, there is an NPC whose main *raison d'être* throughout the whole game is to recommend the player some of the new and usually best items she can acquire at vendors throughout the virtual world. There is also a general worry of making the item names be descriptive enough so that the player quickly understands their function.

With all this said, one can ask: what more is there to tackle in the improvement of these interfaces? One should remember that the recommendations previously mentioned are non-personalized, i.e., an item is recommended because the game developers thought it to be important at some point in the game or just because it is a new item in stock, among other potential reasons. Thus, the player may still feel overwhelmed by the great number of items or simply not being sure about what she should get. Taking this into account, this work will hopefully provide an answer to the previous question, showing that recommendations based on the player's preferences may significatively minimize the player's time and effort spent and, consequently, maximize her in-game performance and overall satisfaction with the game.

2.2 Modeling Preferences

In a CRPG, every time we browse the items in a store, thoughts similar to "I prefer this sword over that one" will probably come to mind often. If we know the game well enough, we may even be able to order all available items according to our style of gameplay, from most to less desirable or vice-versa. What allows us to compare one item to another and construct such ordering are our preferences.

Semantically, preferences over some domain of possible choices allow establishing an *order* to these choices so that a *more desirable* choice precedes a *less desirable* one. Hence, preference relations establish an *ordering* between the elements of a set of choices. Following the terminology in Brafman and Domshlak (2009) [5] these elements will be, from this point on, referred to as *outcomes.* The aforementioned orderings, in turn, can be characterized by at least two important criteria: firstly, an ordering is *total* if it is possible to compare any pair of outcomes, or *partial* if it is not; secondly, an ordering is *strict/strong* if no two outcomes are equally preferred, or *weak* if such restriction is not applied on the ordering. It is important to note that preference relations are *transitive*, that is, if an outcome A is preferred over an outcome B, and B over an outcome C, then A is preferred over C.

Although the previous concepts seem reasonably easy to grasp, it's not always easy to work with preferences. The easier scenario to handle is when each outcome we intend to order, using preference relations, has only one relevant *aspect* or *attribute* which is, at the same time, easily *quantifiable.* If the attribute is quantified with a value that is, say, a level of preference (the higher the level, the more preferred it will be), one needs only to order the outcomes by their single-attribute values to specify her preferences. However, in real applications, preferences are usually not this easy to construct and work with. Outcomes often have more than one relevant attribute and they're not always easily quantifiable. There may also arise a need to consider *trade-offs* and *interdependencies* between various attributes.

As stated in Brafman and Domshlak (2009) [5], a preference modeling task should be approached by first asking what the model is and what questions or queries can we ask about it. Since preferences establish an order among outcomes, a simple preference model

[3] Phantasy Star Online, 2000, developed by Sonic Team, published by Sega.
[4] Chaos Rings, 2010, developed by Media.Vision, published by Square Enix.

could correspond to such an ordering. As for the queries we could ask about it, they could be, with only a mention to some, finding the most preferred outcome, comparing between two outcomes and order the set of outcomes using all outcome attributes or just a subset of the attributes. There is also a need for algorithms to compute each of the desired queries given the model but they aren't relevant to the current discussion at hand.

We can then describe a metamodel, for preference modeling, with five basic elements: the *model*, which represents an ordering of possible outcomes; the *language*, which stores preference statements in a compact representation that can be mapped to the model; the *interpretation function*, which handles this mapping; the *queries*, which allow us to ask something about the model; and the *algorithms*, which compute answers to the aforementioned queries. Figure 1 gives a graphical representation of how these elements are interconnected, in which directed arcs indicate choice dependence and dotted line directed arcs indicate the mapping of the language into a model.

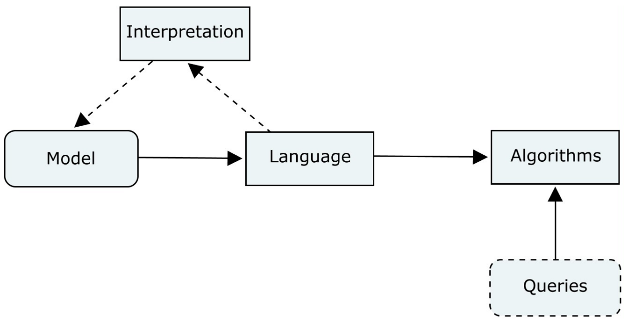

Fig. 1: The metamodel used in preference modeling. Adapted from Brafman and Domshlak (2009) [5].

The language can be either *quantitative* or *qualitative*. A quantitative language is typically more advantageous, due to its compactness and mathematical nature.

Generalized additive independence (GAI) utility functions [8] correspond to the quantitative language used in the context of this work, due to its simple, easy to understand and flexible nature, and its support for dealing with interdependent attributes (see Chen and Pu 2004 [11] on *preferential dependence*). A GAI utility function is defined as:

$$u(x) = \sum_{k \leq K} f_k(x[k]) = \sum_{k \leq K} u_{x[k]},$$

where x represents an outcome consisting of N attributes $X_1, ..., X_N$, K is the number of factors, f_k, with $k = 1, ..., K$, are factors and $x[k]$, with $k = 1, ..., K$, are sets of preferentially-dependent attributes. The attribute subsets associated with each factor need not be disjoint.

The *language* of GAI utility functions assigns a value to each outcome depending on its attribute values, and then the *interpretation function* in use here will dictate that an outcome is preferred over another if it has a higher utility value.

The utility parameters $x[k]$, with $k = 1, ..., K$, are acquired and updated through elicitation, which will be discussed in the following subsection.

2.3 Preference Elicitation

Once noted how preference information can be stored and maintained, one could ask the following questions: How can a recommender system build the user's preference model? How can it elicit, from the user, the necessary information required to build her preference information? The search for an adequate answer to these questions indeed arises as a problem.

Preference acquisition is a problem of growing importance, since people tend to rely more and more on product recommender systems (or product search tools) to find outcomes that best satisfy their needs and preferences [14].

This subsection focuses on two main points: "best-practice" guidelines for recommending outcomes and state of the art techniques to work with partial preference specifications.

2.3.1 Recommendation Guidelines

As the player may not be familiarized with the full set of items in the vendor's store and respective attributes, it is important to make her gain such an understanding to maximize the potential of further preference elicitation. In other words, show *examples* to the user so she can gain a better understanding of the domain of possible outcomes at her disposal. With this said, one may then ask two important questions: *how many examples* should be shown and *what examples* should be shown.

According to Faltings et al. (2004) [12], who investigated the minimum number of examples to show so that the target choice is included even in a situation where the preference model is inaccurate, this number should be given by the following:

$$t = \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{d},$$

where t is the number of shown items so that the target solution is guaranteed to be included, d is the maximum number of stated preferences and ε is a factor that bounds, above or below, the error of the preference function. Pu and Chen (2008) [14] note that, for a number of preferences up to 5, the correct amount of shown items is typically between 5 and 20.

As for the question of what examples to show, the obvious answer would be: the best examples according to the user's preference model. However, as it was previously stated, users may not be fully aware of all the preferences in context, and, as such, the preference model may not yet be sufficiently complete and accurate. Also, users may be uncertain about their own preferences and then try to construct them as they browse through the shown examples and apply critiques to them. Because of these two facts, showing only the best examples may prove itself insufficient to guarantee optimality. The system should then guide the user to develop a preference model that is as complete and accurate as possible, assuring, however, that the initiative to state more preferences remains on the user's side. Pu and Chen (2008) [14] call *suggestions* to examples chosen to stimulate preferences and describe the *diversity strategy* as a possible solution for this problem.

The *diversity strategy* basically scores each outcome taking into account how similar it is to the ideal outcome and how different it is from the already selected recommendations.

Define the score $s(x,Y)$ of an outcome x given an already selected set of recommendations Y and the diversity degree $d(x,Y)$:

$$s(x,Y) = \alpha r(x) + (1-\alpha)d(x,Y),$$

$$d(x,Y) = 1 - \sum_{y\in Y} sim(x,y),$$

$$r(x) = sim(x,x^{*}).$$

The similarity measure $sim(x,y)$ quantifies the similarity between outcomes x and y. The rank $r(x)$ simply calculates $sim(x,y)$, where y is the ideal outcome. The α constant determines the trade-off between similarity and diversity; the higher α is, the more priority is given to similarity and less priority is given to diversity. The α can vary between recommendation cycles, for example having higher values when the player is already expected to have a reasonable knowledge of her own preferences [14]. Finally, score $s(x,Y)$ of an outcome x decides how appropriate it is to recommend that outcome, given the current recommendation set Y.

Even after a set of example items is recommended by the system, the user may still not be able to find an outcome that sufficiently satisfies all of her preferences and may need to choose a partially satisfying one. Also, the shown set of outcomes may have too many possibilities, being difficult for the user to make a decision. The set should then be narrowed down so the user can make a decision more easily. The process of *preference revision* [14] arises as a possible solution to the described issues: it allows the user to change one or more of her previous statements regarding preferred attributes and/or change the degree to which they were previously and statedly preferred.

Lastly, in order for a product recommender system to be successful, users should be able to put trust in it and understand it. After all, we won't choose to follow recommendations we don't understand from someone we don't trust. Thus, a recommender system should be able to convince the users that its recommendations are best suited for them, explaining for that purpose the reasons behind such recommendation choices. This way, recommendations become more transparent and users can feel more confident about them. *Explanation interfaces* [15] are traditionally implemented either through the inclusion of a "why" component associated with each recommendation or through grouping recommendations into multiple labeled categories.

2.3.2 Working with Partial Specifications

It is not always possible, or reasonable, to get a full preference specification because it may require considerable effort and time from the user and, in other cases, it may not even fit into the context. For instance, a vendor asking a player to state all of her preferences the first time they meet doesn't seem a very natural interaction and the player may not even be fully aware of her preferences yet. Thus, it is important to know how to work with only partial preference specifications.

This problem is usually approached either as a computational learning theory problem involving algorithms used in statistical machine learning (such as the Bayesian approach [10]) or as a partially observable Markov decision process (POMDP) problem [2].

Here, the *minimax regret* decision criterion [1,3,4] is presented, accounting for its encouraging results in a recent study and testing in the UTPREF system as it is described in Braziunas and Boutilier (2010) [9].

Minimax Regret works with partial specifications by using a set of utility functions U, each of which imprecisely represents the player's preferences, and choosing an outcome taking all of them into account. Through elicitation these utility functions are bound to converge to the player's true utility function.

Let us formally define *minimax regret* in stages. The *pairwise regret* of choosing outcome x with respect to outcome x' over the utility function set U is:

$$R(x, x', U) = \max_{u \in U} u(x') - u(x).$$

The *maximum regret* of choosing outcome x is:

$$MR(x, U) = \max_{x' \in X} R(x, x', U).$$

The *minimax regret* of utility set U is:

$$MMR(U) = \min_{x \in X} MR(x, U).$$

Basically, the *minimax regret* is the minimum regret that can be achieved, by choosing a *minimax-optimal* outcome x^* instead of its adversarial witness x^w, given the utility function set U.

The use of *minimax regret* is simple to understand, an effective driver of preference elicitation (as discussed on the next section), and does not require a probabilistic prior like many statistical approaches. For a more thorough and general description of the advantages (and disadvantages) of *minimax regret*, refer to Boutilier et al. (2006) [4].

3 Conceptual Model

The developed recommender system consists mainly of five modules: the *Main* module which stores the full preference model and is responsible for communicating with the game; the *Minimax Regret* module which is responsible for deciding which item is most appropriate for a recommendation; the *Elicitation Strategy* module which will request more information from the player when deemed necessary; the *Current Solution Strategy* module which decides what to ask the player in order to maximize information gain; and finally, the *Recommendation Strategy* module, responsible for constructing a list of recommendations based in the item selected in *Minimax Regret*.

See figure 2 for a graphical overview of the system's modules and, inter-module interactions, and interaction with the game.

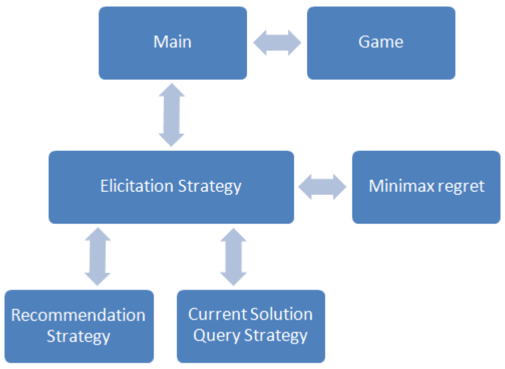

Fig. 2: Recommender system model overview and its interaction with a game.

The recommender system's preference model uses GAI utility functions, in order to capture attribute dependencies, as they were described in subsection 2.2. Utility parameters are updated through replies to queries (see ahead) and whenever an item is purchased.

The *Minimax Regret* module uses, as its name implies, the *minimax regret* decision criterion as it was described in subsection 2.3.2, to choose the best outcome to recommend.

The *Elicitation Strategy* module decides when to make recommendations, using the results computed in the *Minimax Regret* module. The process can be summarized as follows:

1. Compute *minimax regret* MMR.
2. Repeat until $MMR < RegretThreshold$:
   (a) Ask query Q.
   (b) Update utility parameters on U according to the response given to Q.
   (c) Recompute *minimax regret* MMR with the updated U.

The *Current Solution Query Strategy* module implements the adopted query strategy with the same name (CS in short), first described in Boutilier et al. (2005, 2006) [3,4] and recently tested in the UTPREF recommender system, with encouraging results [9].

The CS strategy adopted here uses *bound queries*. In a *bound query*, the player is directly asked if one of her utility parameters lies above a certain value q. A positive response raises the respective utility parameters with values lower than q to value q, for every utility function in U. A negative response lowers the respective utility parameters with values higher than q to value q, for every utility function in U. The query is formulated using the current solution, that is, the current *minimax-optimal* outcome and its adversarial witness as basis. Refer to Boutilier et al. (2006) [4] for the complete formal definition.

Finally, *Recommendation Strategy* module uses the results computed in *Minimax Regret* and decides which recommendations should be made. The adopted strategy chooses outcomes with least *maximum regret* for the first half of recommendations, and uses the *diversity strategy* (see subsection 2.3.1) to select the rest. The number of recommendations to show is decided by the game.

4 Evaluation

The developed recommender system was evaluated in a CRPG made specifically for this work's purpose, with a total of 17 participants between ages 23 and 34 involved. Each testing session involved playing through two different scenarios, A and B. Scenario A used the developed recommender system as it was described in the previous section; Scenario B uses a modified recommender system that randomly generates queries and recommends only the outcomes with most *maximum regret*. Each scenario corresponds to a full run of the CRPG.

4.1 Testing Session

Before each testing session, players were simply informed that they were going to play a CRPG, in which they would be interacting with an intelligent vendor.

The CRPG (for either scenario) starts with an introductory phase, where the player is given access to a few weapons to stimulate her to accurately express her preferences later on. As soon as the player meets the vendor, initial weapons are taken away, she is given the option to choose one of three described heroes as a role model (each of which technically translates to a set of utility functions that will be used to initialize preference information) and from there she has to start acquiring and upgrading her equipment on her own. Introductory phase aside, the CRPG is made of three quests and before each of them the player has the option to interact with the vendor. After the three quests are completed, the vendor gives the player a final, powerful weapon, which is technically the *minimax-optimal* outcome, and then asks the player to rate it.

The key interface between player consists on the following options: display all available weapons, display only the recommendations, ask the player a query about her play style, and, lastly, reset her initial play style information to that of a different hero, effectively resetting the preference model to an initial utility function set.

Information was collected through a questionnaire, game logs and an open discussion at the end of the testing session.

4.2 Results

Results collected were both quantitative and qualitative in nature. Quantitative result analysis shows that scenario A was significantly superior when compared to scenario B: generally, in scenario A the game experience was more enjoyable, recommendations were more appropriate, more recommended weapons were bought, queries were observed to make more sense, and it was noticeable how queries had a more positive effect on subsequent recommendations.

It is, however, important to note that, while scenario A was shown to be superior, the game experience in scenario B had an average rating of 3.41, in a scale of 1 to 5, which is above the medium value of 3 and relatively close the average of 4.12 for scenario A. This can be explained by the fact that both scenarios had exactly the same vendor interface; even when, for example, the recommendations proved to not be the best, the player still typically feels that she is being supported, which improves the game experience.

Similarly, the quality of queries in scenario B was rated with an average of 3.24, as opposed to an average of 4.00 for scenario A, in a scale of 1 to 5. This can be attributed to the fact that queries in scenario B followed the same construction template as those of scenario A; only the values asked were randomly generated.

Interestingly, the number of times the players asked for recommendations and the number of times they asked to be queried were approximately the same in both scenarios. This indicates a willingness to interact with such a vendor and use the respective services only made possible by a recommender system.

Qualitative results confirmed that, in general, players expressed their interest in interacting with an intelligent vendor, such as the one shown in the CRPG, and felt that the idea of a well-constructed recommender system might be a very welcome and useful addition to the world of CRPGs, not only for recommendation purposes, but also for game experience personalization.

5 Conclusions

The act of bringing preference handling concepts and techniques, in particular preference elicitation, to the world of CRPGs is still a novel concept, yet to be fully explored, and has shown here to be particularly challenging. Game interaction and respective interface creation require more care than many store applications which try to prioritize being effective and efficient for when a customer is interested in buying something, rather than building a more natural connection to the player, stimulating her to express her preferences and feel compelled to buy something, not because she strictly needs it, but because she learned to want it. Thus, implementing a recommender system within a game's virtual world is arguably much more of an art than implementing it on a classic web store's artificial interface.

This work intends to give an overview of the current state of the art techniques applied in the area of preference handling, particularly in a preference elicitation context, directs its attention to the world of CRPGs and its current vendor mechanics and interfaces. The primary goal of this work is to gain a better understanding of how one can build a recommender system in the context of a CRPG to help players make more accurate decisions when interacting with vendors and, thus, achieve a better game experience.

The results obtained during the evaluation of the developed recommender system to be integrated in a game are very encouraging, with every participant manifesting their enthusiasm towards such an approach. Result analysis confirms their interest and willingness to use such a system in real games.

Two scenarios, A and B, were tested. Scenario A used the recommender system as it was described in the section 3, and scenario B used a modified version that randomly generated queries and recommended the worst items according to the *minimax regret* decision criterion. In any of the two scenarios, all players made sure they used the recommender system and asked the vendor for recommendations, and, in consequence, the majority felt that a good recommender system positively impacts the game experience. Overall, the recommender system in scenario A was considered considerably better and more useful than its scenario B counterpart.

Many players also felt interested in having the vendor knowing them better and offering them better recommendations, and thus specifically requested the vendor to query them.

It is also worthy of notice how the recommender system in scenario B was still seen as a real recommender system, as many players felt that the system was still trying to help them, even though ultimately not performing as well as the recommender system in scenario A, and thus proved to be a valuable object of comparison for this study.

Finally, even though several improvements and further study can still be achieved, one can say that the developed recommender system met the goals of this work and is, without a doubt, a great starting point for gaining a reasonable understanding about introducing recommendation support systems to the world of games.

5.1 Future Work

There are several steps that naturally follow the work introduced here, be them improvements to existing functionality or the introduction of new features to make the recommender system more effective and simple to understand, from a player perspective. The suggested points where future work should focus on are as follows:

– Allow for explanatory information within the system to be directly visible in vendor interfaces. A player's trust in a vendor is typically directly related to how much she understands the vendor; if the recommender system can compute and associate a natural-looking reason for each recommendation made, the player will feel more compelled to believing in the system and following its advice.
– Add support for storing and managing multiple preference models, in the context of a multiplayer game. In an age where online games of every genre are very common, it is only a natural evolution to consider multiplayer options. Additionally, it would be interesting to study how a player's preference model could influence the recommendations for a different player.
– Further study the elicitation parameters, to better understand which values they should have in specific contexts, and, additionally, how should they change throughout elicitation cycles and throughout the whole game experience.
– Add support for different types of items, with different attributes. For example, potions, which will most likely have an entirely different attribute set than weapons, could also be covered by the recommender system along with weapons. The current recommender system only handles items with the same attribute set; although, admittedly, it would be possible to run X separate instances of the developed recommender system, to accommodate for X item types with different attribute sets.
– Allow the recommender system to customize attribute dependencies for each player, through understanding her. The current system assumes a fixed GAI utility structure for every player; utility personalization can then be considered a natural, although non-trivial, next step for this work. This problem has already started to be approached in Brafman and Engel (2009) [6].
– Support more powerful preference revision techniques (see subsection 2.3.1). The current recommender system's preference revision is mostly limited to resetting the preferences to initial values.
– Consider formulating the recommender system logic as a constraint-based optimization problem, as in Boutilier et al. (2006) [4], making the system more flexible to the introduction of other types of queries, such as comparison queries, and other query strategies, such as the ones presented in Braziunas and Boutilier (2010) [9].
– Use of more appropriate interfaces should be considered. Although not directly related to the recommender system itself, the use of interfaces that allow for ordering and categorizing items displayed would be particularly helpful to avoid confusion and stimulate trust in players (see subsection 2.3.1 on *explanation interfaces*).
– Improve how queries are shown to the player. While not directly related to the recommender system, it is important that queries asked feel natural, not too technical, and in context with the game's setting.
– Consider introducing calibration across factors, to make sure the utility values used for all factors are meaningful and consistent with each other. See Braziunas and Boutilier (2005) [7] and Gonzales and Perny (2004) [13] for further details. The current system simply uses utility values from 0 to 100, for all factors.
– Explore further personalization of game experience through preferences. The developed recommender system supports the use of preferences to customize the game experience, for example through story branching, but evaluation didn't prioritize this idea, focusing only on assessing the recommender system, the core of this work.

References

1. Boutilier, Patrascu, Poupart, and Schuurmans. Constraint-based optimization with the minimax decision criterion. In *ICCP: International Conference on Constraint Programming (CP), LNCS*, 2003.
2. Craig Boutilier. A pomdp formulation of preference elicitation problems. In *AAAI/IAAI*, pages 239–246, Edmonton, 2002.
3. Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 929–934. Professional Book Center, 2005.
4. Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using minimax decision criterion. *Artificial Intelligence*, 170(8–9):686–713, 2006.
5. Ronen Brafman and Carmel Domshlak. Preference handling - an introductory tutorial. *AI Magazine*, 30(1), 2009.
6. Ronen I. Brafman and Yagil Engel. Directional decomposition of multiattribute utility functions. In Francesca Rossi and Alexis Tsoukiàs, editors, *ADT*, volume 5783 of *Lecture Notes in Computer Science*, pages 192–202. Springer, 2009.
7. Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. In *UAI*, pages 42–49. AUAI Press, 2005.
8. Darius Braziunas and Craig Boutilier. Minimax regret based elicitation of generalized additive utilities. In Ronald Parr and Linda C. van der Gaag, editors, *UAI*, pages 25–32. AUAI Press, 2007.
9. Darius Braziunas and Craig Boutilier. Assessing regret-based preference elicitation with the UTPREF recommendation system. In David C. Parkes, Chrysanthos Dellarocas, and Moshe Tennenholtz, editors, *ACM Conference on Electronic Commerce*, pages 219–228. ACM, 2010.
10. Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 363–369, Menlo Park, CA, July 30– 3 2000. AAAI Press.
11. L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, EPFL Technical Report IC/2004, 2004.
12. B. Faltings, P. Pu, M. Torrens, and P. Viappiani. Designing Example-Critiquing Interaction. In *Proceedings of the International Conference on Intelligent User Interface(IUI-2004)*, pages 22–29. ACM Press, 2004. Funchal, Madeira, Portugal.
13. Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *KR2004: Principles of Knowledge Representation and Reasoning*, pages 224–233. AAAI Press, Menlo Park, California, 2004.
14. Pearl Pu and Li Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29(4), 2008.
15. Pearl Pu, Paolo Viappiani, and Boi Faltings. Increasing user decision accuracy using suggestions. In *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems*, volume 1 of *Social computing 1*, pages 121–130, 2006.