POLITECNICO DI MILANO
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Exploring Recommender Systems for decision-making in e-Tourism

Relatore: Prof. Franca Garzotto
Correlatore: Prof. Paolo Cremonesi

Tesi di Laurea di:
Antonio Donatacci, matricola 748840

Anno Accademico 2011-2012

Ai miei cari

"Soltanto l'ardente pazienza porterà al raggiungimento
di una splendida felicità."

Pablo Neruda

Summary

Many factors that influence users' decision making processes in Recommender Systems (RSs) have been investigated by a relatively vast research of empirical and theoretical nature, mostly in the field of e-commerce.
In this thesis, we discuss some aspects of the user experience with RSs that may affect the decision making process and outcome, and have been marginally addressed by prior research. These include the nature of users' goals and the dynamic characteristics of the resources space (e.g., availability during the search process).
We argue that these subjective and objective factors of the user experience with a RS call for a rethinking of the decision making process as it is normally assumed in traditional RSs, and raise a number or research challenges.
These concepts are exemplified in the application domain of on-line services and specifically hotel booking, a field where we are carrying on a number of activities in cooperation with a large stakeholder (Venere.com - a company of Expedia Inc.).
Still, most of the arguments discussed in this work can be extended to other domains, and have general implications for RS design and evaluation.

Ringraziamenti

Prima di tutto vorrei ringraziare mia mamma, mio papà e mia sorella per avermi sempre supportato e sopportato in questo lungo e a volte difficile percorso di Laurea.
Un grazie speciale lo dedico anche alla mia fidanzata, Iza. Grazie al suo sostegno e alla sua solarità mi ha sempre aiutato nei momenti peggiori, facendomi ritrovare l'allegria.
Grazie alla Professoressa Franca Garzotto e al Professore Paolo Cremonesi per la proposta di questa tesi, per avermi sempre fatto sentire a mio agio e per la loro costante disponibilità.
Vorrei anche ringraziare Roberto Turrin per il sostegno offertomi in una fase delicata dello sviluppo di questa tesi, nonchè tutto il gruppo Moviri per la calorosa accoglienza e simpatia con cui mi hanno accolto.
Grazie anche a Venere.com, e in particolare a Vito Perrone e Fabrizio Salzano, per l'aiuto e la fiducia riposta nel nostro gruppo di lavoro e per averci fornito strumenti fondamentali per la realizzazione di questo lavoro. Senza il loro contributo nulla di tutto ciò sarebbe stato possibile.
Infine vorrei ringraziare tutti gli amici e i compagni delle mie studiate, a cominciare da Giuseppe F., il quale, anche a distanza, si fa sentire (eccome!), per continuare con Alessio (il Frank), Matteo (il Coach), il Dolphin, Alice, Silvia, Marco e Luca. Grazie a tutti per la compagnia.

Antonio Donatacci

Introduzione

I sistemi di raccomandazione (RSs) supportano l'utenza durante la ricerca di vasti insiemi di contenuti digitali, identificando con più efficacia gli elementi, quali prodotti o servizi, che possono essere considerati più attrattivi e utili. In quanto tali, i RSs si contraddistinguono come strumenti che forniscono assistenza alle persone nel corso del loro processo decisionale, ovvero quando prendono una decisione tra un elevato numero di alternative [40].
Una considerevole parte delle ricerche in questo ambito si è dedicata a capire come i RSs influenzano i processi decisionali e gli esiti di tali processi. Una revisione della letteratura su questo tema, incentrata sull' e-commerce, è riportata in [40]. Tali autori sostengono che, nel considerare i RSs come strumenti di supporto, il design e la valutazione debbano tener conto di aspetti che vanno al di là degli algoritmi di raccomandazione utilizzati, poichè questi aspetti influenzano in maniera significativa i processi decisionali e il risultato di questi ultimi. Questi aspetti sono legati a fattori soggettivi individuali ed anche a caratteristiche di design della user experience con i RSs.
Sebbene esistano una serie di argomentazioni teoriche e studi empirici a supporto dell'influenza positiva che tali sistemi hanno nei confronti della qualità del processo decisionale, la ricerca in questo settore è tuttavia non conclusiva, sottolineando la necessità di ulteriori studi. Questa tesi è volta a fornire nuovi contributi in quest'area di ricerca.
I principali studi sui RSs, in qualità di strumenti decisionali di supporto, si sono focalizzati sul dominio dell'e-commerce, in cui gli utenti acquistano prodotti on-line. Il nostro lavoro invece esplora i processi decisionali nel vasto dominio applicativo dei servizi online, in particolare nella prenotazione degli alberghi. Abbiamo svolto una serie di attività in stretta collaborazione con uno stakeholder di primo piano in questo settore, Venere.com (www.venere.com). Quest'ultimo fa parte del gruppo di Expedia.inc, gruppo leader nel mercato della prenotazione alberghiera online, che può contare su più di 120.000 hotel, Bed&Breakfast e strutture vacanziere in 30.000 de-

stinazioni in tutto il mondo.
In questo dominio investighiamo alcuni aspetti soggettivi della user experience con i RSs (il tipo di obiettivo degli utenti), alcuni attributi oggettivi dei RSs (cioè relativi ad aspetti di design) e la natura delle risorse (ad esempio la disponibilità di queste ultime nel tempo, in particolare durante il processo di ricerca) che potrebbero influenzare in modo significativo il processo decisionale supportato dai RSs. Inoltre, molte delle nostre considerazioni possono ritenersi valide in altri domini ed avere quindi implicazioni in aspetti teorici e pratici nel design dei RSs e nella loro valutazione in generale.

Principali contributi della tesi

Il lavoro di tesi fornisce in questo contesto quattro nuovi contributi che sono i seguenti.
Il primo è rappresentato dall'esplorazione dei sistemi di raccomandazione come strumenti di supporto nel vasto dominio applicativo dei servizi online, quale ad esempio la prenotazione di alberghi online. A tal fine abbiamo lavorato cooperando con uno stakeholder di grande importanza in questo dominio, nello specifico Venere.com, affiliato del gruppo Expedia.inc.
Il secondo contributo consiste nell'aver considerato una serie di nuovi aspetti della user experience in relazione ai RSs che sono stati trascurati nei precedenti studi e che riteniamo possano avere una grande influenza nei processi decisionali e nelle scelte finali dell'utenza; tra questi abbiamo esaminato le caratteristiche dei compiti assegnati all'utente svolti col supporto del sistema (ad esempio, la prenotazione di un hotel per motivi di vacanza o viaggio di lavoro) e le caratteristiche dinamiche degli oggetti (ad esempio, la loro disponibilità o meno durante il processo di ricerca). Nel primo caso, abbiamo arricchito la rappresentazione degli oggetti considerando come la differente tipologia di obiettivo dell'utenza possa avere un impatto diverso sui processi decisionali sotto la condizione di limitate risorse cognitive (bounded rationality). Nel secondo caso, abbiamo incrementato la complessità degli oggetti considerati, introducendo il cambiamento del loro status durante l'interazione dell'utente col sistema.
Il terzo contributo è rappresentato dalla definizione di un modello concettuale che ha come obiettivo quello di supportare il design e lo studio empirico da parte dei ricercatori di un vasto spettro di applicazioni dei RSs. Tale modello definisce un framework più completo rispetto a quelli esistenti e migliora la nostra comprensione dei RSs in relazione ai processi decisionali.
Infine il quarto contributo è l'implementazione di un software framework per la valutazione che faciliti l'esecuzione di studi controllati sull'utenza in

questo ambito. Tale framework ha un'architettura modulare che può essere facilmente personalizzata rispetto a diverse sorgenti di dati e tipologie di algoritmi di raccomandazione e abilita la ricerca nello studio della manipolazione e controllo di diverse variabili, con lo scopo di accertare in maniera sistematica gli effetti dell'utilizzo dei RSs sui processi decisionali.

Organizzazione

La tesi è organizzata nel modo seguente.
Il capitolo 2 descrive lo stato dell'arte e i concetti chiave che sono ritenuti necessari per la comprensione di tutti gli aspetti della tesi. Il capitolo è suddiviso in quattro sezioni, ognuna dedicata ad uno specifico argomento. La prima si riferisce ai Sistemi di Raccomandazione (RSs), in cui è presente una panoramica delle caratteristiche di tali sistemi e una loro tassonomia. La seconda sezione presenta gli studi sul comportamento dell'utenza e sulla valutazione dei RSs. In dettaglio sono discussi i principali fattori di persuasione dei sistemi, fornendo un modello del processo decisionale. In aggiunta viene descritto un framework per la valutazione dei RSs. La terza sezione illustra il punto di partenza della nostra ricerca, ovvero il modello concettuale di Xiao e Benbasat, che consente di studiare i principali effetti dell'uso dei RSs sui processi decisionali e sulla valutazione dei RSs da parte degli utenti. Infine nella quarta sezione è descritto il contesto dell'e-tourism e vengono forniti i più significativi esempi di applicazione dei RSs in questo dominio.
Il capitolo 3 definisce il modello concettuale, presentando il nostro approccio teorico al modello precedente di Xiao e Benbasat. Inizialmente vengono discusse le principali problematiche dei servizi di e-booking, da cui le derivanti research questions. Successivamente, al fine di studiare tali research questions, viene descritto il modello concettuale esteso e vengono fornite le principali metriche di valutazione dei suoi costrutti.
Il capitolo 4 presenta il framework PoliVenus che abbiamo implementato. Il capitolo inizia con una breve descrizione della sua architettura e della tecnologia impiegata per il suo sviluppo. Inoltre si mostrano in dettaglio tutte le componenti e i moduli del software e le loro funzionalità, con particolare enfasi sul modulo abilitante i test e le sue possibilità di configurazione.
Il capitolo 5 illustra lo studio del design degli esperimenti empirici, con le spiegazioni in merito ai vari scenari proposti in questo lavoro per valutare i diversi fattori dei RSs e la procedura operativa pianificata, descrivendo i gli strumenti adottati, i partecipanti, il luogo di svolgimento dei test e la strategia di ricompensa scelta.
Infine il capitolo 6 conclude il lavoro riassumendo quali aspetti possono esse-

re considerati in prospettiva futura. Si suddividono tali aspetti in due aree principali, ossia il lavoro da svolgere per arricchire il framework software e possibili studi da compiere sul modello concettuale.

Indice

Elenco delle figure

Elenco delle tabelle

Chapter 1

Introduction

Recommender Systems (RSs) help users search large amounts of digital contents and identify more effectively the items - products or services - that are likely to be more attractive or useful. As such, RSs can be characterized as tools that help people making decisions, i.e., make a choice across a vast set of alternatives [40].
A vast amount of research has addressed the problem of how RSs influence users' decision making processes and outcomes. A systematic review of the literature about this topic, focused on e-commerce, is reported in [46]. These authors pinpoint that when we regard RSs as decision support tools, the design and evaluation of these systems should take into account other aspects beyond the algorithms that influence users' decision-making processes and outcomes. These aspects are related to individuals' subjective factors as well as the design characteristics of the user experience with the RS. While several theoretical arguments and empirical studies exist that support the positive effects of RS use on decision making quality, research in this field is still inconclusive, highlighting the need for further research. This thesis provides some novel contribution to this research area.
Most prior work on RSs for decision support focused on e-commerce domains where users buy on-line products. Our work has instead explored decision making processes in the wide application domain of on-line services, specifically, hotel booking. We are carrying on a number of activities in close cooperation with a key stakeholder in this field, Venere.com (www.Venere.com). This is a company of the Expedia Inc. group which is leader in online hotel reservations market featuring more than 120,000 hotels, Bed and Breakfasts and vacation rentals in 30,000 destinations worldwide. In this domain, we investigate some subjective aspects of the user experience with RSs (the type

of users' goals), some objective, i.e., design related, attributes of RSs, and the nature of the resources space (e.g., the availability of items along the time in general, and specifically during the search process) that may affect the decision making processes supported by RS.
Still, most of our considerations can be extended to other domains, and have implications for research and practice in RS design and evaluation in general.

1.1 Thesis contribution

This thesis provides four novel contributions.
First, we explore RSs as decision support tools in the wide application domain of on-line services, such as hotel booking. We have worked in close cooperation with key stakeholders in this domain, specifically Venere.com, a company of Expedia Inc.
Second, we take into account a number of new aspects of the user experience with RSs which have been neglected by previous studies and may significantly influence users' decision-making processes and outcomes; these include the characteristics of the tasks performed with the system (e.g., booking a hotel for vacation or for a business trip) and the dynamic characteristics of items (e.g., availability during the search process). In the first case, we enrich previous studies considering how different typologies of goals can impact the user decision making under bounded rationality constraints. In the second case we increase the complexity of items by introducing the change of their status during the interaction between user and system.
Third, we define a conceptual model that helps researchers design and contextualize empirical studies in a wide spectrum of RS application domains. Our model provides a more comprehensive framework than the existing ones, and improves our understanding of user decision making processes in RSs.
Fourth, we provide a software framework for evaluation that facilitates the execution of controlled user studies in this field. The framework is based on a modular architecture that can be easily customized to different datasets and types of recommender algorithms, and enables researchers to manipulate and control different variables in order to systematically assess the effects of RS use on users' decision making processes.

1.2 Thesis organization

The thesis is organized as follows.
Chapter 2 describes the state of art and the key concepts considered necessary to understand all the aspects of the thesis. The chapter is divided in four parts, each one for a specific argument. The first is related to *Recommender Systems* (RSs), where we provide an overview of the main characteristics of these systems and a taxonomy. The second part presents the studies on user behavior and RS evaluation. Most specifically we argue the RSs' persuasive factors, providing a model for the user decision making activity. In addition it is introduced a framework for the evaluation of RSs. The third part illustrates the starting point of our research: the Xiao and Benbasat conceptual model, a framework to study the effects of RSs on decision making and user's evaluation of RS. Finally in the fourth part we describe the e-tourism context, providing examples of the main works concerning RSs.
Chapter 3 defines our extended conceptual model, with the explanation of the theoretical approach to the existing Xiao and Benbasat model. At first we argue about the main issues that succeed to e-booking services, which lead to the research questions. Afterwards, in order to study these research questions, we provide the extensions of the existing conceptual model and a description of the evaluation metrics of its constructs.
Chapter 4 presents the PoliVenus framework we developed. We start the chapter with an overview of its architecture and the enabling technology. Furthermore we show in detail the main modules and functionalities of the framework, with a particular emphasis on the testing application and its configurations.
Chapter 5 illustrates the design of the empirical study, with the explanation of the scenarios proposed in this work to evaluate RSs factors and the planned execution, describing the instruments, the participants, the location, and the reward strategy.
Finally, chapter 6 summarizes the main areas of interest for future works, with an attention on theoretical works on the conceptual model and on the completion of the framework.

Chapter 2

State of Art

In this chapter it is exposed the state of art concerning the Recommended Systems (RSs) on e-commerce platforms and their distinctive features.
It is also investigated the potential influences of the relevance, transparency, effort required, and persuasion that those systems exert on consumers decision processes as well as their influence on user's evaluation of RSs.
In order to argue on these subjects, the most important and widespread frameworks and conceptual models are analyzed. We illustrate the behavioral model introduced by Fogg to evaluate the influence of technology on users, the decision making model adopted, and the ResQue framework of Chen and Pu, in order to analyze which aspects should be taken into account concerning RSs evaluation.
Finally it is discussed the role of RSs as Decision Support Systems (DSS); in this perspective RSs differ from the traditional assistance of a DSS as they are not the decision makers but provide a support, facing a class of problems called *preferential choice problems*.
Furthermore will provide relevant examples of RSs in the tourism domain, with a particular care about the application of those technologies (e.g., recommendation of travel destinations).

2.1 Recommender systems and technologies

The increasing size and complexity of product assortments offered by e-commerce platforms requires appropriate technologies which alleviate the retrieval of products by online customers.
Different recommendation technologies have been developed to help customers to easily find the best matching product. Those technologies have

been successfully applied in different domains such as financial services, electronic goods, or movies. Some examples of applications can be found in [30]. Although several Recommender Systems (RSs) exist in literature and their characteristics concerning tools, technology, methods, and algorithms are sometimes quite different, it is possible to classify them into the following taxonomy.
As first dimension we can identify whether a RS is personalized or non-personalized [36].

- personalized recommendations
- non-personalized recommendations

In the first case we assume that recommendations are made with the support of a user profile, built by the RS itself for a specific user. In detail such systems differ each other by the domain of interest, the knowledge and data they refer to, the algorithm, and finally the way recommendations are presented in the layout layer.
In the latter a user model is not taken into account, recommending items which are the same for every user. An example of these systems is when they suggest the best movies of the week.
Considering the first approach (personalized RSs) the most addressed by researches and studies in this sector, we can further divide this branch into another classification which depends on the family of the recommender algorithm.
The most widespread technologies of personalized RSs are summarized below [9], [1]:

1. Collaborative filtering recommender system(CF)
2. Content-based filtering recommender system(CBF)
3. Hybrid recommender systems
4. Knowledge-based recommender system (KB)
5. Case-based recommender system (CBR)

2.1.1 Collaborative filtering recommender system

The first type of recommender system is the collaborative - or social - filtering one, which exploits user ratings of products in order to identify appealing products that user may like as well. Hence they ignore content and exploit the preference of other users with similar tastes.

The main input structure of those algorithms can be reduced to a *user item matrix*, where $r_{u,i}$ is the value of the rating expressed by the user u to the item i. Non present values are set to zero. An example of this matrix is showed in table 2.1 Two main approaches belong to this family:

User	Item	Value
1000234name 1000234surname	hotel-id-2	2.2476
1000258name 1000258surname	hotel-id-2	9.4133
1000307name 1000307surname	hotel-id-2	8.9967
1000331name 1000331surname	hotel-id-2	7.9433
1000337name 1000337surname	hotel-id-2	9.5333
1000339name 1000339surname	hotel-id-2	8
1000366name 1000366surname	hotel-id-2	9.0033
1000369name 1000369surname	hotel-id-2	8.2951
1000426name 1000426surname	hotel-id-2	7.4751
1000430name 1000430surname	hotel-id-2	9.1084
1000433name 1000433surname	hotel-id-2	10
1000442name 1000442surname	hotel-id-2	9.645
1000455name 1000455surname	hotel-id-2	7.4117

Table 2.1: User item matrix example

neighborhood models : in this approach the prediction is computed through the similarity relationship between the items or the users.

latent factors models : this approach (informally Singular Value Decomposition models - SVD) tries to define ratings thanks to factors or preferences of items or users inferred by customers' feedback. To characterize users or items, these family of algorithms analyze obvious dimensions (such as category of a movie etc ...).

User-based and item-based collaborative filtering are two basic variants of the first approach.
As shown in table 2.2, both variants are predicting to which extend the active user would like currently unrated items. User-based approaches to collaborative filtering try to identify the nearest neighbors of the active user (users having similar tastes), and calculate a prediction of the active user's rating for a specific item. This rating can be defined, for example, as the weighted average of the nearest neighbors' ratings. In the simplified example of table 2.2, User1 is found to be the nearest neighbor ($k = 1$) of User3 (the active user) and his/her rating for the 4th product ('Conspiracy Theory')

will be taken as prediction for the rating of User3 ($rate = 2$).
In contrast, item-based collaborative filtering is searching for items which received similar ratings from other users and were also (positively) rated by the active user. In the given example from [1], 'Pretty Women' has been rated by all users. This is the most similar item to 'Conspiracy Theory' and it is assumed that User3 will have the same preference for 'Conspiracy Theory' ($rate = 1$).
In this approach the main advantages are obtained with the item-based CF because generally it is more scalable [1] and it is easier to explain the reason of the recommendation to the consumer.
In the neighborhood models of collaborative filtering, the most popular algorithms are:

- *Correlation neighborhood (CorNgbr)*: Correlation Neighborhood predicts the rating $r_{u,i}$ for user u on item i as the weighted average of the ratings of similar items properly unbiased[2].

- *Non-Normalized Cosine Neighborhood (NNCosNgbr)* : Simplified version of the CorNgbr which ranks the items basing on their appeal to the users, without forcing ratings into a specific range of value.

In the latent factors models, the main algorithms studied are:

- *Asymmetric SVD (AsySVD)* : the main principle of this algorithm is to reduce the correlation between users and items to some factors; for a given set of factors, items are estimated basing on the extent they possess those factors; on the other side users are estimated basing on the extent they are interested in some items with those factors.

- *PureSVD* : proposed recently, it is based on conventional AsySVD with the difference that unknown ratings are treated as zeros.

2.1.2 Content-based filtering recommender system

Content Based Filtering, namely CBF, provides recommendations for preferred products category, i.e., they suggest items which have been previously rated positively by the user in the past [21].
Let us assume, the active user has rented the 'Pretty Women' movie; using descriptions of genre, starring and price, CBF recommends, for example,

[1] The number of items is in general lower than the number of users

[2] This can be obtained in the simplest form by subtracting to the single rating the average rating of the user (i.e., eliminate the bias defined as the tendency of certain users to rate higher than others)

	Content based			Collaborative		
Product	Genre	Starring	Price	User1	User2	User3
Pretty woman	Romance	Gere, Roberts	13	1	3	1
Runaway bride	Romance	Gere, Roberts	10	4	2	3
Under suspicion	Thriller	Hackman, Freeman	13	3	2	3
Conspiracy theory	Thriller	Gibson, Roberts	10	2	3	?

Table 2.2: Content-based and Collaborative Filtering Recommendation (k=1, 1=very good; 4=bad)

'Runaway Bride'. If no product categorization is available, and the items are represented only as free text descriptions (e.g., Netnews, books, emails etc.), alternative CBF solutions based on, for example, information retrieval techniques are available. They extract a set of keywords from textual product descriptions, compute the users preferences expressed in terms of keywords which are contained in products bought by the user, and build the list of recommendations by searching for products that match the user's preferences. In both cases, items are defined as bag of words (BOW), where each word is weighted basing on its importance in representing the item content.
A typical weighting schema is the TF-IDF, where the more a word occurs in an item, the more importance it has but, on the other side, the more this word is shared between items, the less important it is for each one [37].
Once items are represented in terms of BOW, users can be defined as the ratings they previously gave to items. Subsequently recommendations are obtained by calculating vectors similarity.
An example of BOW related to this work, obtained with a stemming elaboration, whose value is obtained with TF-IDF scheme is showed in table 2.3.
To conclude the description of these two main families of RSs, we will provide an example of reasoning sentences which can describe enough good the background work of RSs. In the case of collaborative filtering algorithms, the reasoning behind the RS is "Customers who bought this item also bought..." In the case of content-based algorithms the reasoning is "Look for related items by keyword" and "Look for similar items by category".

2.1.3 Knowledge-based recommender system

Both of the previous approaches have their own strength and weakness point. The strength point consists of the possibility, in a collaborative RS, to use the 'collective preferences of the crowd' [9] in obtaining appealing alternative set of products, while in a content-based RS, to walk the user down a discrimination tree of product features. The weakness point is identified

Stem	TF-IDF
stazion	0.018121222838544938
time	0.024002234459047214
apert	0.02319293850885766
riunion	0.02362791312035167
est	0.026131836777936738
trov	0.019089883308093192
soddisf	0.02637578632902712
lussos	0.22502189449400153
dirett	0.03873548428054095
ristorant	0.05678919449038975
pied	0.021515886962092848
offert	0.02047892393602137
autobus	0.024769030329365633

Table 2.3: Stem TF-IDF matrix example

with the two well-known problems called the first 'cold-start', the second 'stability vs. plasticity'. In fact in a CF approach, as the number of ratings increases, the probability that a user in the system model will match a new user (i.e., the system will be likely to be useful) will also increase. Therefore CF RSs need to initialize their dataset with a large amount of data.
The same argument can be discussed with a CBF approach, where the user must have rated a good number of items to make comprehensible to the system his/her preferences. The aspect to have numerous ratings to initialize the data source is crucial to make these types of RS operate good and this issue is well addressed in the cold-start problem.
The second problem shows up when a user model is built up in the system and it becomes difficult to change one's preferences [8]. For example a steak-eater who becomes a vegetarian will continue to get steakhouse recommendations from a content-based or collaborative recommender for some time, until newer ratings have the chance to tip the scales.
From these considerations follows the idea that a RS should not depend on a base of user ratings and additionally its judgments should be independent from user tastes.
The two main ideas under a knowledge-based RS are

- limit CF and CBF drawbacks
- provide knowledge, information, and interaction during the support process

The first aim is achieved by an explicit representation of items, decision processes, and context which leads these kind of systems to manage, in an e-commerce environment, complex product and context restrictions (such as services or mobile environment). The second aim derives directly from the explicit knowledge representation which leads to:

- calculate solutions that fulfill certain quality requirements
- explain solutions to a customer
- support customers during the decision

Hence knowledge-based recommender systems do not attempt to build long-term generalizations about their users, but rather base their advice on an evaluation of the match between a user's need and the set of options available. Knowledge-based RSs 'reason' on the relationship between user needs and preferences and the way items meet these needs and requirements. This knowledge called 'functional knowledge' [8] is used by these systems to build a user profile made of functional rules which support inference. An example of these systems is Google RS, in which the recommender examines the relation (in terms of functional knowledge) between links and provides advices based on these recognized matching patterns. In this case the Google RS examines the correlation between pages to infer the popularity and authoritative degree.
The main crucial point of these systems is the acquisition of knowledge about item and user features in order to make inference. The knowledge involved is of three main categories:

Catalog knowledge: Knowledge about the objects being recommended and their features. For example, for a restaurant the recommender should know that 'Thai' cuisine is a kind of 'Asian' cuisine.

Functional knowledge: The system must be able to map between the user's needs and the object that might satisfy those needs. For example, the recommender knows that a need for a romantic dinner spot could be met by a restaurant that is 'quiet with an ocean view'.

User knowledge: To provide good recommendations, the system must have some knowledge about the user. This might take the form of general demographic information or specific information about the need for which a recommendation is sought.

In order to gain independence from items and user ratings these systems generally rely on ontologies as their main 'knowledge container'. An ontology

is in fact defined as a conceptualization of a domain into a human understandable, but machine-readable format, consisting of entities, attributes, relationships, and axioms [28]. Ontologies can provide a rich conceptualization of a domain, representing the main concepts and relationships.These relationships can be isolated information or an activity. These RSs use a semantic utility function to rank the items users may be interested in, defining this utility value on the item classification.
Adomavicius and Tuzhilin [2] formalized a recommendation problem by assuming an utility function *rec* that measures the usefulness of an item $i \in I$, where I is the set of items, to a user $u \in U$, i.e. $rec : U \times I \longmapsto R$, where R is a totally ordered set of numbers within a certain range. The knowledge-based approach then applies a sequence of filters to evaluate items properties depending on the user model. Each filter consists of a condition *(If ...)*, a consequent *(then ...)*, and a priority value. If the subset of results is empty or too small[3], i.e. none of the items in the product repository satisfies all applicable restrictions, then the lower priority personalization filters are stepwise relaxed until at least one item is part of the result.
A simple and spreadwise technique to obtain an utility function is the scheme Multi-Attribute Utility Theory (MAUT), where each attribute is weighted in correspondence of user preference. For instance the estimation of the user $u \in U$ in item i is computed as $rec_{ut}(u,i) = \sum_{p \in P} score_{u,p} \cdot score_{i,p}$, where P is the set of abstract domain properties, $score_{u,p}$ the estimated interest of u in a property $p \in P$ and $score_{i,p}$ the utility score an item i achieves with respect to p.
$rec_{ut}(u,i)$ is then computed as the weighted sum of an item's property score with the estimated user interest in that property.
However, pure knowledge-based recommenders do not compute an utility score, but decide if an item is part of their result set or not. Therefore the rec function is defined as follows:

$$rec_{kb}(u,i) = \left\{ \begin{array}{ll} 1 & \text{if } kb \vdash i \\ 0 & \text{else} \end{array} \right\} \tag{2.1}$$

The main drawback of these systems, the so called interests-acquisition problem, consists of the need for knowledge acquisition [28]. Another aspect to consider is that knowledge-based RSs can make recommendation as wide-ranging as its knowledge base allows.

[3] This condition can vary depending on the policy adopted

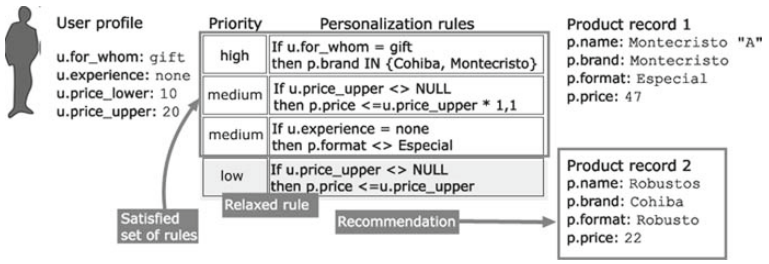

Figure 2.1: An example of knowledge-based recommendation

2.1.4 Hybrid recommender systems

The hybrid recommender system approach aims to avoid some of the drawbacks encountered with CF and CBF techniques, combining two or more recommendations strategies to gain more performance. Typically a collaborative algorithm is combined with another technique to avoid the cold-start problem. In the table 2.4 are exposed the main ways of combination methods

Hybridization method	Description
Weighted	Combination of several techniques into a single recommendation.
Switched	Switching between recommendation techniques on different situations
Mixed	Different recommenders results are presented at the same time
Feature combination	Features from different data sources together into a single algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Table 2.4: Hybridization methods

Weighted combination

In a weighted hybrid recommender, the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system. For example, the simplest combined hybrid would be a linear combination of recommendation scores. The combination can be obtained in two ways: the system can give collaborative and content-based recommenders equal weight, but gradually adjusts the weighting as predictions about user ratings are confirmed or disconfirmed. Or alternatively the system treats the output of each recommender (collaborative and content-based) as a set of votes, which are then combined in a consensus scheme.
The benefit of a weighted hybrid is that all of the system's capabilities con-

vey to a single direction into the recommendations while the main drawback is constituted by the assumption that the relative value of the different techniques is more or less uniform across the space of possible items [4].

Switching combination

A switching hybrid uses some criteria to switch between different recommendation techniques.
The first criterion that addresses the switching process consists of a threshold which is used to establish a level of system's confidence of recommendations. If a first algorithm (generally content-based) fails to recommend items over the confidence threshold level, the second algorithm (collaborative one) is used. In this situation the novelty is the possibility to recommend items cross genres (when collaborative algorithm takes over), which may be perceived useful from the user. Another way to adopt this strategy is to use an history of algorithms predictions and user preferences in order to make a comparison between the two and to choose the algorithm that reflects better user's tastes.
The main problems are still the cold-start[5] and an increasing complexity due to a switching criteria parameter that must be inserted in the processing of this hybrid approach.

Mixed combination

Where it is practical to make large number of recommendations simultaneously, it may be possible to use a 'mixed' hybrid, where recommendations from more than one technique are presented together. Recommendations from the two techniques are combined together in the final suggested program. The content-based component can be relied on to recommend new items on the basis of their descriptions even if they have not been rated by anyone.
It does not get around the 'new user' start-up problem, since both the content and collaborative methods need some data about user preferences to get off the ground.

[4] Collaborative algorithm is weaker for items with low number of raters

[5] This hybridization is conditioned by using only one recommendation technique while the other comes to play when the former fails

Feature combination

Another way to achieve the content/collaborative merger is to treat collaborative information as simply additional feature data associated with each example and use content-based techniques over this augmented data set. The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item. Conversely, it lets the system have information about the inherent similarity of items that are otherwise opaque to a collaborative system [8].

Cascade combination

Unlike the previous hybridization methods, the cascade hybrid involves a staged process. In this technique, one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set. Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well-differentiated by the first or that are sufficiently poorly-rated that they will never be recommended. Because the cascade's second step focuses only on those items for which additional discrimination is needed, it is more efficient than, for example, a weighted hybrid that applies all of its techniques to all items. In addition, the cascade is by its nature tolerant of noise in the operation of a low-priority technique, since ratings given by the high-priority recommender can only be refined, not overturned [8].

Feature augmentation combination

One technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique. While both the cascade and augmentation techniques sequence two recommenders, with the first recommender having an influence over the second, they are fundamentally quite different. In an augmentation hybrid, the features used by the second recommender include the output of the first one. In a cascaded hybrid, the second recommender does not use any output from the first recommender in producing its rankings, but the results of the two recommenders are combined in a prioritized manner.

Meta-level combination

The last way that two recommendation techniques can be combined is by using the model generated by one as the input for another. This differs from

feature augmentation: in an augmentation hybrid, we use a learned model to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input.
The benefit of the meta-level method, especially for the content/collaborative hybrid is that the learned model is a compressed representation of a user's interest, and a collaborative mechanism that follows can operate on this information-dense representation more easily than on raw rating data.

2.1.5 Case-based recommender systems

The CBR is a knowledge-based system that exploits a 'search and reuse' approach. The search is performed on the catalogue of items (to be suggested), and the reuse of retrieved items could be implemented in different ways, from a simple display of the retrieved items to a more complex adaptation of the items to fit to the peculiar preferences of the user [22].
In detail CBR is a multidisciplinary subject that reuses the knowledge and experience acquired in the past, by wrapping them into a *case*. In literature such systems are well described as a computational paradigm for problem solving applied in specific situation (specific knowledge), in contrast to classical approaches based on general knowledge about problem domain, representable with a knowledge language composed of rules, frames, semantic networks and first order logic.
The fundamental issue encountered in a CBR approach is the definition of three crucial aspects:

i the content of the case

ii the structure to describe such content

iii the case memory organization

The first aspect is generally strictly domain dependent and requires analysis of what nuances should be taken into account. The structure of a case can be implemented in three main ways: as a linear vector (eventually a set) of features previously identified; as a text, free or semistructured; finally as a labeled graph or as a series of object (in an object oriented paradigm).
In the first approach, CBR systems are strongly connected with machine learning problems and pattern recognition as their case language representation is oriented to problem solving resolution through a classification or approximation tasks.
In the text-based CBR systems, the text is considered as the source of knowledge and therefore it is elaborated with techniques of information retrieval

such that text is lead in a machine readable format. The techniques applied consist of text summarization, keywords extraction[6], and tag generation with metadata content.
The last approach consists of a combination of previous techniques where a case is defined as a node in a graph. Such node can be defined in an object oriented paradigm where it is possible to highlight the hierarchy of the case structure (e.g., the task, the subtask etc ...).
An example of a generic case in a CBR model[7] decomposed in its four components: $CB \subseteq X \times U \times S \times E$,
where X is the product/content model, U is the user model, S is the session model, and E is the evaluation model. This means that a general case $c = (x, u, s, e) \in CB$ in a generic CBR-S consists of four (optional) subelements x, u, s, e which are instances of the spaces X, U, S, E respectively [22].
The structure to describe the content of a case, independently from the content itself, is divided in three components: the problem description, the solution, and the outcome. The first refers to the part that is matched when a new problem arises. This must include all the information needed to first guess that a case can be fruitfully reused for solving a similar problem. Considering problem solving as function approximation, the problem description becomes the domain of the function, where the co-domain is given by the solution and outcome.
The solution models the chunk of information that is searched for, e.g. the diagnosis of a malfunction or the plan to reach a destination.
Finally, the outcome provides an evaluation of the applicability or goodness of the solution to the given problem.
The last point refers to the problem-solving CBR cycle which can be summarized in the four steps below [16]:

1. **Retrieve:** given a problem description, retrieve a set of cases stored in the case base, whose problems are evaluated as similar. A similarity metric is used to compare the problem component of the new case being built with the problem description of the cases in the base. Indexes, case base partitions, case clusters or other similar tools can be used to speed up this stage.

2. **Reuse:** the retrieved cases are reused to build the solution. This stage could be very simple, e.g. only extract the 'solution' component from

[6] We already discussed of some of these techniques in the content-based algorithms section

[7] This representation is independent from its implementation

one retrieved case, or much more complex, e.g. to integrate all the solutions extracted from the retrieved cases to build a new candidate solution.

3. **Revise:** the solution is then adapted to fit the specific constraint of the current situation. For instance, a reused therapy for a patient suffering for similar disease must be adapted to the current patient (e.g. considering differences in the weight or age of the two patients).

4. **Review:** the constructed solution is evaluated applying it (or simulating the application) to the current problem. If a failure is detected, backtracking to a previous stage might be necessary. The 'reuse', 'revise' and 'review' stages are also called case adaptation.

5. **Retain:** the new case is possibly added to the case base. Not all the cases built following this process must be added to the case base. There could be poorly evaluated cases or cases too similar to previous situations, and therefore not bringing new knowledge.

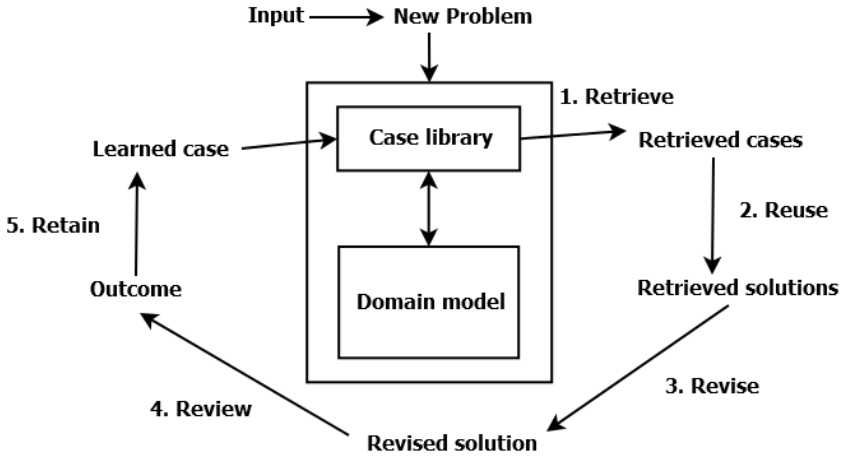

Figure 2.2: CBR process

The main drawback encountered in the CBR approach is the possible performance limitations due to storage of cases. It is a crucial aspect of this methodology to adopt a correct policy of cases' memorization, which can also provide an efficient discarding technique of unnecessary cases [22].

2.2 Consumer behavior and evaluation of RSs

The increasing interest in user-centric approach and the need to study and evaluate the impact and effects that RSs produce on consumers lead to augmented consideration in how the persuasion can affect users' evaluation and satisfaction [31] and their final behavior [14].
Becoming a necessary component that proactively suggests items of interest to users, RSs are no longer considered a fanciful website add-on, making the evaluation of the quality of user experience and users' subjective attitudes toward the acceptance of recommender technology an important issue.
In this perspective, all the major theoretical frameworks and models are considered in order to study two important phenomena of RSs:

- the influence on the decision making process
- the user's evaluation of RS

In the next sections are examined the main set of theoretical principles from different well-known theories concerning on the two aforementioned subjects. These theories[8] are finally realized in the Xiao and Benbasat model, i.e. the conceptual model which we refer to in our work, which aims to study RSs and their impact under a multi-dimensional approach.

2.2.1 Recommender systems persuasive factors

The most widespread and famous instrument to analyze the persuasive effect of a computing product (including RSs) is the Fogg Functional Triad. In Fogg work the influence that a system is able to produce is declined in three main dimensions called roles:

- *System as a tool*
- *System as a media*[9]
- *System as a social actor*

This wide framework characterizes all type of computing products, therefore we will investigate only the main aspects which can be referred to RSs. In particular these include the tool and the social actor role.

[8] Part of them are still to be investigated while another part are the main cues of this work

[9] This aspect is related to the possibility that a system is used to create a simulation of real experiences to influence behavior and therefore will not be considered in this study

The tool role

In the first dimension, the 'tool role', the author pinpoints that one of the main characteristic of a computing product is constituted by its ability to reduce the computational weight of the user, making the whole task performed easier and effortless. Under this approach, a RS can be seen as persuasive technology tool which interact in three different ways with the user:

Reduction: technologies which use this strategy make target behavior easier by reducing a complex activity to a few simple steps [13].

Tunneling: technologies which lead users to the goal through a sequence of predefined steps

Tailoring/suggestion: technologies which compute products that provide information relevant to individuals

Psychological and economic theories suggest that humans seek to minimize costs and maximize gains. The theory behind reduction technologies is that making a behavior easier to perform increases the benefit/cost ratio of the behavior. According with Fogg, increasing the perceived benefit/cost ratio increases user's motivation to perform the task and to adopt the support technology (namely RS).
An example of this concept is well explained by Amazon "one click" function, where with one click of the mouse a consumer can purchase automatically items, with billing on credit card, packaging up, and shipping off included.
With a tunneling technology, a system can lead user into a predetermined sequence of steps to reach a target behavior, in our case a decision. The main aspects which are considered in this interaction are aesthetics and content type, where it is particularly important for the user to be guided in a choice or to explore new alternatives which he/she may not consider otherwise.
The last functionality with which a system can influence user behavior is part of the definition of RSs themselves. The basis of this strategy is that systems can provide better functionality if they reduce the space of items for consumers, in order to provide them only relevant choices. This can lead to an increasing sense of benefit and a major adoption of the system, besides a perceived better result in reaching goals.

The social actor role

In the social role, it is discussed the influence that a system can achieve by giving a variety of social cues that elicit social responses from human users [13].

Tintarev [41] highlights the importance explanations assume with respect to transparency, as well the other roles explanations have that are relevant to persuasion, e.g., trust, effectiveness (helping users make good decisions), efficiency (helping users make faster decisions), and satisfaction. Let's consider, for example, a collaborative RS which "explains" to the user that a specific recommendation is gathered because other users with similar profile, that liked his/her preferred hotels in the past, also liked the one recommended. It is possible that the user may be more susceptible about the suggestions with respect to a system which doesn't provide any explanation. This happens because user's mind tends to associate the system to a familiar (more human) figure from which it is possible (in reliable and trusted way) to retrieve advices [13].

2.2.2 Credibility

Credibility can be defined as the *"believability"* that a user may have towards a system [13]. Some recent studies define the credibility of a system as the union of two factors:

- Perceived expertise
- Perceived trustworthiness

Users evaluate these two elements and then combine them to develop an overall assessment of credibility.

Expertise

The first dimension of credibility is the expertise, defined as the knowledge, skill, and experience of the source.
Many cues lead to the perception of expertise. Among them we can find 'labels', representing signals that systems can send to users to indicate their level of expert or professional degree. Another cue is the appearance that a system can show through interaction or environment in which the system acts; the last cue is the 'accomplishment information' through which the system indicates its operating degree and thus usefulness.

Trustworthiness

The trustworthiness is the second key factor of credibility. It captures the perceived goodness or morality of the source.
The first guideline of trustworthiness is the perception that a source is fair and unbiased; a second guideline is built on the contrast between own-interest

act, other-interest act: sources that argue against their own interest are perceived as being credible. Finally perceived similarity between users and the system leads to perceived trustworthiness.
As a conclusion, given that both trustworthiness and expertise lead to credibility perception, the most credible sources are those that have high levels of trustworthiness and expertise.
Another consideration must be done on the inter-relationships between these two factors: if one dimension is strong while the other is unknown, the system still may be perceived as credible, due to the "halo effect" which suggests that if one virtue is evident, the other may be assumed; on the contrary if it is perceived a dimension as weak, the whole credibility will suffer even though the other dimension is strong or not.
Credibility can furthermore be declined into four types which can come into play in a system:

- *Presumed credibility*
- *Surface credibility*
- *Reputed credibility*
- *Earned credibility*

Presumed credibility can be defined as the extent to which a user believes the system because of general assumption. This type of credibility is referred to the role that computing systems play in relation to the user. In the perspective of support systems, their presumed credibility can be assumed with a positive value because of the intrinsic nature of their function which leads to think about them as expert systems.

Surface credibility is related to the very first impression of the user over the system. This kind of judgment is the shortest process of credibility formation, but also the most frequent and enduring.
In the theoretical and empirical studies it is argued that this kind of credibility is related mainly to design aspects: presentation, interaction and navigation are crucial to the first impact of the user and constitute the starting point to build this kind of credibility.

Reputed credibility can be defined as the extent to which a person believes a source because of what third party reports. This type of credibility has a strong relationship with the provider of RS, because in this situation

RSs are embedded either in the websites of online vendors or in third party websites and therefore such providers influence users' credibility about RSs.

Earned credibility is the strongest form of credibility. It derives from users' interactions with the system over an extended period of time. This type of credibility is the most solid form, leading to an attitude that may not be easily changed. It also plays an important role since it can be formed during a repeated use of the system or in general pretrial use of it.

Type of credibility	Basis for believability
Presumed	General assumption in the mind of the perceiver.
Surface	Simple inspection or initial experience.
Reputed	Third-party endorsements or referral.
Earned	First hand experience that extends over time.

Table 2.5: Types of credibility

2.2.3 Consumer's behavior and decision making

Consumer behavior is defined as the "acquisition, consumption, and disposition of products, services, time and ideas by decision making units" [46]. Major domains of research in consumer behavior include information processing, attitude formation, decision making, and factors (both intrinsic and extrinsic) affecting these processes.
Our attention focuses on consumer decision making related to the acquisition or buying of products, and separates the outcomes of decision making from the processes of decision making.
According with [46], the application of RSs to assist in consumers' shopping task influences their decision-making processes and outcomes. When supported by RSs in decision making, consumers will enjoy improved decision quality and reduced decision effort [46].
Agreeing with these considerations, the effect of RSs on consumer decision making can be analyzed on two complementary aspects:

- the impact of RS use on decision processes
- the impact of RS on decision outcomes

Impact of RS use on decision processes

In complex decision-making environment, users often can't evaluate in-depth all available alternatives before making their choices [46]. This theory, called

bounded rationality, is based on the constraints of time and computational effort under which a user is subjected.
Besides, a user often has to seek to attain a satisfactory, although not necessarily an optimal, level of achievement, by applying their rationality only after having greatly simplified the set of choices available.
Under this assumption, the decision process can be seen as a dual stage process; in the first stage, the user tries to simplify the number of items; in the second the user can execute a deeper comparison of the alternatives between which he/she will finally make a decision. Of course the information processing vary stage by stage.
In detail the first stage is called *screening stage* while the second is called *in-depth comparison stage*.
As mentioned above, the information processing, as well as the cognitive effort and the set of alternatives, may change during the progressing of the decision process: it is possible to identify four sets of alternatives during the whole process. The alternative sets are:

- *Complete solution space:* the whole set of products available in the database(s)
- *Search set:* the subset of products that is searched
- *In-depth set:* the subset of alternatives for which detailed information is acquired
- *Consideration set:* the subset of alternatives seriously considered
- *Final choice:* the alternative chosen

Since RS affects both stages in terms of facilitating screening and comparison activities, their influence on decision making process can be measured with respect to *effort* exerted during the stages.
According with [46] the effort is defined as "the amount of effort exerted by the consumer in processing information, evaluating alternatives, and arriving at a product choice". It can be divided into *decision time* and *extent of decision choises*.
The former is defined as the time consumers spend searching for product information and making purchase decisions. Since RS are processing information in the screening and comparative phase, according with users' expressed preferences, consumers can dedicate their time in a deeper comparison between less alternatives, factor which lead to decrease complessive time [46].

The latter is defined as the number of product alternatives that have been searched, for which detailed information is acquired, and have seriously been considered for purchase by consumers. Thus, a good indicator for the extent of product search is the size of the alternative sets and, in particular, the size of the search set, the in-depth search set, and the consideration set.
Since RSs present lists of recommendations ordered by predicted attractiveness to consumers, compared to consumers who shop without RSs, those who use RSs are expected to search through and acquire detailed information on fewer alternatives (i.e., only those close to the top of the ordered list), resulting in a smaller search set, in-depth search set, and consideration set. Thus, the use of RSs is expected to reduce the extent of consumers' product search, by reducing the total size of alternative sets as well as the size of the search set, in-depth search set, and consideration set.
An additional indicator of the consumer decision effort is the *amount of user inputs*, defined as the number of interaction required from the user to express or to explicit his/her preferences.
Xiao and Benbasat [46] et al. pinpoint that relevant results show how decision time and extent of decision choices decrease under usage of a RS, producing a useful support perception to users.

Impact of RS use on decision outcomes

A second aspect of the consumer decision making to be analyzed is the RS impact on the decision outcomes. This effect refers to the objective or subjective quality of a consumer's purchase decision [46].
First of all, it is possible to establish if a chosen item is a *dominated (suboptimal)* or *non-dominated (optimal) choice*.
The second evaluation refers to *preference matching score* of the selected alternatives, which measures the degree to which the final choice of the consumer matches the preferences she has expressed [33].
The third evaluation is referred to the quality of consideration set, averaged across individual product quality.
The last dimension is the *product switching*, that is, after making a purchase decision, when given an opportunity to do so, if a customer wants to change her choice and trade her initial selection for another [46].
To complete this analysis, it should be considered that for RSs, the trade-off between the maximization of accuracy (i.e., decision quality) and the minimization of effort disappears: the searching, comparing, reordering and listing activity demanded to the RSs let the user free to spend his decision time on deepening analysis of alternatives, resulting in a perceived better

choice.

2.2.4 ResQue: an RSs evaluation framework

The increasing interest in user-centric approach and the need to study and evaluate the impact and effects that RSs produce on consumers lead to augmented consideration on evaluating the quality of user experience and users' subjective attitudes toward the acceptance of recommender technology.
An important framework which aims to identify essential determinants that motivate users to adopt this technology is the Chen-Pu framework called *ResQue* [34].This framework uses two widespread usability theories, the first called TAM (Technology Acceptance Model), the second called SUMI (Software Usability Measurement Inventory).
In the TAM theory it is explained the correlation between some RSs qualities and the intention from the user to adopt this technology; the main constructs of this model are perceived ease of use, perceived usefulness and users' intention to use the system.
The second theory, namely SUMI, is a psychometric evaluation model to measure the quality of software from the end-user's point of view. The model consists of 5 constructs (efficiency, affect, helpfulness, control, learnability) which can be used to assess the quality of use of a software product. By adapting these two theories, ResQue framework is composed by four es-

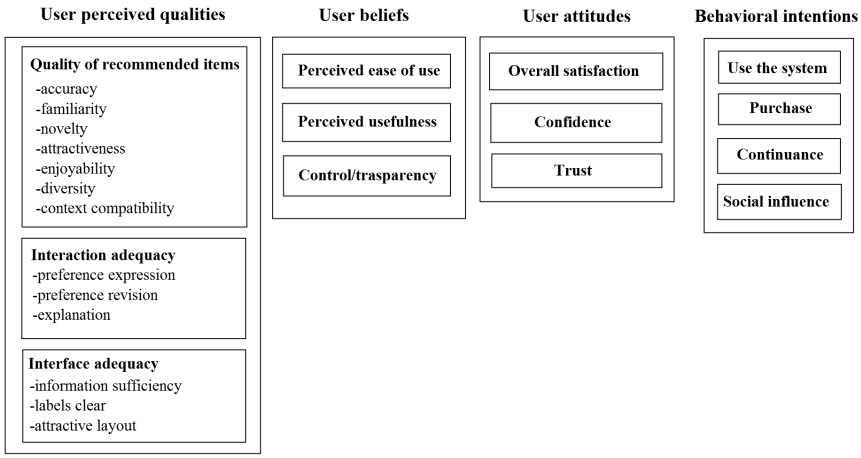

Figure 2.3: ResQue framework

sential constructs: *1) User perceived quality of the system, 2) user beliefs, 3) user attitudes, 4) behavioral intentions.*

User perceived quality of the system

This construct can be divided into four dimensions.
The first is the *quality of recommended items* which refers to the functional and informational aspect of a recommender and how the perceived qualities of these aspects influence users' beliefs on the ease of use, usefulness and control/transparency of a system. Into this functional block there are information quality and genuine usefulness of the suggested items. In detail, it is possible to improve such dimension by grouping recommended items in a significant area of the interface, moreover improve the quality of recommendation. Such quality is related to the following factors:

Perceived accuracy: the degree to which users feel the recommendations match their interests and preferences.

Familiarity: describes whether or not users have previous knowledge of, or experience with, the items recommended to them.

Novelty: the extent to which users receive new and interesting recommendations

Enjoyability: refers to whether users have enjoyed experiencing the items suggested to them.

Diversity: measures the diversity level of items in the recommendation list.

Context compatibility: evaluates whether or not the recommendations consider general or personal context requirements.

The second dimension is *the interaction adequacy*, which defines the ability of a RS to propose alternatives into three variables: the ability in presentation of the results, the possibility to revise those alternatives (feedback from users), and finally the possibility to receive an explanation from the system about the recommendations.
The last dimension is the *interface adequacy* which investigates how to optimize the recommender page layout to achieve the maximum visibility of the recommendation, i.e. whether to use image, text, or a combination of the two.

Beliefs

User beliefs are the results in terms of RSs qualities in terms of ease of use, usefulness and control.
The first measures users' ability to quickly and correctly accomplish tasks

with ease and without frustration.
Besides the dimension of the ease of use is further divided into the perceived initial effort, i.e. the perceived effort users contribute to the system before they get the first set of recommendations, and the easy to learn effort which refers to the minimal amount of learning by design that RSs require at the very first interaction.
The usefulness of a recommender is the extent to which a user finds that using a recommender system would improve his/her performance, compared with their previous experiences without the help of a recommender.
The user control measures whether users felt in control in their interaction with the recommender. The concept of user control includes the system's ability to allow users to revise their preferences, to customize received recommendations, and to request a new set of recommendations.
Related to this concept is the *transparency* of the system which is measured as the extent to which a system allows users to understand its inner logic, i.e. why a particular item is recommended to them.

User attitudes

Attitude is a user's overall feeling towards a recommender, which is most likely derived from his/her experience as he/she interacts with a recommender. An attitude is generally believed to be more long-lasting than a belief. Users' attitudes towards a recommender are highly influential on their subsequent behavioral intentions.
Main factors which influence user attitudes are the *overall satisfaction*, that determines what users think and feel while using a recommender system; the *confidence inspiring*[10], referring to the recommender's ability to inspire confidence in users, or its ability to convince users of the information or products recommended to them; finally *trust*, that indicates whether or not users find the whole system trustworthy.

Behavioral intentions

According with [35], behavioral intentions towards a system is related to whether or not the system is able to influence users' decision to use it and purchase some of the recommended results.
This crucial factor can be measured in some ways:

- *user loyalty*, i.e. the system's ability to convince users to reuse the system

[10] In Fogg the term persuasion is used with the same meaning, see paragraph above

- *user acceptance of the system*, i.e. the purchase event
- *user retention*, i.e. the user self conviction to reuse the system

Generally these subjective factors are measured through a survey.

2.3 Recommenders as decision support systems

The study of RSs falls within the domain of information systems.
As information technology artifact [5], RSs are characterized as a type of customer decision support system (DSS) based on the three essential elements of DSS: (1) DSSs are information systems, (2) DSSs are used in making decisions, and (3) DSSs are used to support, not to replace, people. Like other DSSs, when employed, RSs receive user inputs and process this information to retrieve items in line with such preferred features and produce the output solution.
The unique feature which distinguishes RSs from traditional DSSs is the *RSs support function:* the first principle with which DSSs were developed was to execute computational complex problems in order to assist high-level planning activities such as logistics, marketing, and financial planning. In this routine, the output of the system was the solution itself users could only acknowledge.
In the RSs support perspective, users are assisted in their activity and the system is treated as belonging to the domain of *preferential choice problems* [42]. Thus, choice models, which support the integration of decision criteria across alternatives, are the primary decision support technologies employed by RSs.
Additionally, RSs are designed to understand the individual needs of particular users (customers) that they serve. Users' beliefs about the degree to which the RSs understand them and are personalized for them are key factors in RS adoption. Moreover, there is an agency relationship between the RSs and their users; users (principals) cannot be certain whether RSs are working solely for them or, alternatively, for the merchants or manufacturers that have made them available for use. Therefore, users may be concerned about the integrity and benevolence of the RSs, beliefs that have been studied in association with trust in IT adoption models, as aforementioned in the section above [46][13].
The design of RSs is composed by three main components: the input, namely where user preferences are elicited, explicitly or implicitly; the process, where recommendations are generated; finally the output, where recommendations are presented to the user.

According with Xiao and Benbasat [46], most of the previous studies on RSs focus only to the process, loosing sight of the input and output strategies. Besides many studies consider only algorithms aspects, without considering adequately other factors which influence users about RSs (as explained in the previous section) [31].
From results of many studies on this subject, Xiao and Benbasat describe a conceptual model to explain how different factors of RSs, such as the characteristics of RS input, process, output, source credibility, and product-related and user-related factors, determine the effectiveness of RSs [46].

2.3.1 Xiao and Benbasat conceptual model

The aforementioned reasons, in addition to the need for discovering the influence of RSs on decision making process and how their characteristics can influence users' evaluation, are investigated by Xiao and Benbasat with the following research questions:

1. *How do RS use, RS characteristics, and other factors influence consumer decision-making processes and outcomes?*
    - 1.1 *How does RS use influence consumer decision-making processes and outcomes?*
    - 1.2 *How do the characteristics of RSs influence consumer decision-making processes and outcomes?*
    - 1.3 *How do other factors (i.e., factors related to user, product, and user-RS interaction) moderate the effects of RS use and RS characteristics on consumer decision-making processes and outcomes?*
2. *How do RS use, RS characteristics, and other factors influence users' evaluations of RSs?*
    - 2.1 *How does RS use influence users' evaluations of RSs?*
    - 2.2 *How do characteristics of RSs influence users' evaluations of RSs?*
    - 2.3 *How do other factors (i.e., factors related to user, product, and user-RS interaction) moderate the effects of RS use and RS characteristics on users' evaluations of RSs?*
    - 2.4 *How does provider credibility influence users' evaluations of RSs?*

In order to answer these research questions, which unify the study on behavioral and decision process subjects in high level concepts, Xiao and Benbasat developed a widespread framework to analyze RSs in a decision support system point of view.

Furthermore, in this perspective, they summarize the characteristics and features of RSs and their relationships among the outcomes of RS use.
Their conceptual model is composed by a set of constructs based on previous conceptual and empirical research in information systems in general, and decision support systems, RSs in particular. These key constructs are: *outcomes of RS use*, (consisting of consumer decision making and users' evaluations of RSs), *product*, *user*, *user-RS interaction*, *RS characteristics*, *provider credibility*, and *RS use*. The model is further decomposed into a collection of propositions which map the relationships among these constructs and constitute the statements of research questions.
Such propositions derive from five theoretical perspectives:

- theories of human information processing
- the theory of interpersonal similarity
- the theories of trust formation
- the technology acceptance model (TAM)
- the theories of satisfaction

The final purpose of the model is to answer the research questions expressed above and to enable the set up of empirical studies which can validate (or eventually invalidate) research questions' hypothesis. Hence research questions enabled by the model are in strict correlation with the relationships among the different constructs.
In the following paragraphs there is a description of the conceptual framework and its constructs and relationships. These latter investigate the impact the constructs have on the consumer decision making process and the evaluation of RS.
In order to simplify the presentation of the conceptual model, we will provide at first definitions and descriptions of constructs relating to the consumer decision making process, which answers to the first research questions' group; afterwards we will give the same definitions relating to user evaluation of RS which provide answers to the second research questions' group.

Relationship constructs - consumer decision making process

RS use According with Xiao and Benbasat [46], as predicted by the theoretical perspective of human information processing, when supported by RSs in decision making, consumers will enjoy improved decision quality and reduced decision effort.

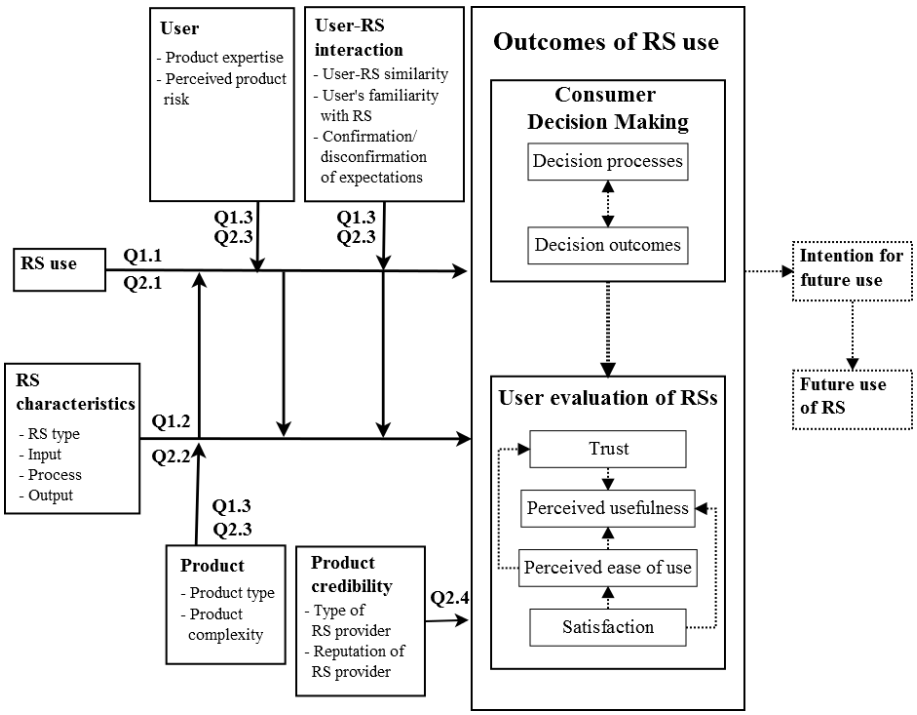

Figure 2.4: Xiao and Benbasat conceptual model

In agreement with theories of *bounded rationalities* discussed above, the model indicates a relationship between RS use and the consumer decision making. This relationship can be further divided into the decision process construct and the outcomes of RS use.
In the former it is inferred that RS reduces extent of products to be examined by the consumer, therefore also the number of products for which detailed information is needed; RS reduces the time spent by the user to complete shopping task (i.e., reduces the waist of time in preliminary activities such searching, comparing, and reordering).
By the definition, decision effort is composed by the three components:

- Extent of product search
- Decision time
- Amount of user input[11]

Therefore it can be inferred that consumers experience a reduction in decision effort.
The outcomes of RS use increases user's decision quality because consumers can free up some of the processing capacity in evaluating alternatives, which will allow them to make better quality decisions. Moreover, RSs enable consumers to easily locate and focus on alternatives matching their preferences, which may also result in increased decision quality.

RS characteristics According with Xiao and Benbasat [46], all RSs are not created equal. As such, the effects of RS use on consumer decision making are determined, at least in part, by RS characteristics (i.e., RS type and characteristics associated with the input, process, and output design).
The model proposed by Xiao and Benbasat enables the comparison of different RSs which use different process strategies (i.e, different algorithms). As we explained above, the taxonomy of such systems is composed by a multitude of algorithms' families; the most known are collaborative, content, hybrid, knowledge-based, case-based algorithms.

RS Input Characteristics Another analysis dimensions proposed by Xiao and Benbasat [46] are preference elicitation methods and included product attributes.
The authors suggest that the way information about preference and consumer's interests are captured can influence the decision making process.

[11] No studies are provided by the author on this dimension

Following theories of bounded rationality [39], it is possible to infer that due to limited cognitive resources, consumers tend to form their ideas on products and their preferential features on spot, for example when they face a choice.
In the context of RS use, consumer preferences can either be gathered implicitly (by building profiles from customer's purchase history or navigational pattern) or elicited explicitly (by asking customers to provide product ratings/rankings or answer preference elicitation questions). The means by which preferences are elicited may affect what consumers do with the RSs' recommendations. In fact since users preference elicitation methods can lead to sub-optimal solution choices, a direct relationship between the RS input characteristics and consumer decision process occurs.
Another aspect highlighted by Xiao and Benbasat [46] concerns the role of RSs in the usage of input elicitation. This role is by definition the simplification and computational reduction of decision tasks, and therefore it is mainly based on the consideration of aspects for which the user expresses an interest or a preference. This role can be simplified in the concept that RSs are highly 'selective' [17].
A way RSs can influence consumer decision process is to include the attributes of the products and their value in the presentation of results. This is an important aspect under the assumption that users perceive RSs are expert on the domain and believe that such systems evaluate products features according with the preference expressed. Hence they are likely to consider the product attributes included in the RSs' preference-elicitation interface to be more important than those not included. Consequently, products that are superior on the included attributes will be evaluated more favorably and will be more likely to be chosen by consumers than products that are superior on the excluded attributes.

RS output characteristics The RSs characteristics discussed in this construct refer to the methodology of presentation of recommendations. In detail it is discussed how the design of the recommendation and its content are fundamental aspects in the model which influence the decision making process.
Besides it is also considered the inclusion of the predicted ratings and the utility scores calculated by RSs in the layout's presentation.
According with Xiao and Benbasat [46], given that the primary function of RSs is to assist and advise consumers in selecting appropriate products that best fit their needs, it is expected that the recommendations presented by the RSs will influence consumers' product choices.

Beyond this, under limited cognitive resources, consumers tend to be influenced by the context surrounding the recommendation; that's why utility scores and predictions about user rating add an expertise sensation that can positively influence the consumer final choice.
As exposed by the authors [46], recommendation format can exert an important role in the decision making process. This design variable can be analyzed under two dimensions:

- the order of recommendations
- the number of recommendations

The first dimension presents two possible patterns: a sorted recommendation list and a random recommendation list.
In the first case the list of recommendation is completely ordered following the RSs computation criteria. Under this condition, the user may find all interesting recommendations but the choice may become more difficult because all the proposals differ very little each other.
In the second condition, consumers may find easier to choose a proposal because of the increased gap between each recommendations, but on the other hand the space of recommendation can be waisted in not useful proposals and the credibility of the whole system can decrease.
The second aspect is the number of recommendations. Providing too many recommendations, consumers effort in decision process may increase due to the large number of comparisons. On the other hand providing a small set of proposals may result in user's dissatisfaction.

Product-related factors According with the authors [46], the factors which influence the most the relationship between product and consumer decision making are the type of product and the complexity of its evaluation.
A product can be defined as *search product* or *experience product*; while the former is characterized by attributes (e.g., color, size, price, and components) that can be assessed based on the values attached to their attributes, without necessitating the need to experience them directly, the latter is characterized by attributes (e.g., taste, softness and fit) that need to be experienced prior to purchase [46].
Because the evaluation of experience products requires more information search activities than search products, users may feel uncertain about the matching between the selection and their expectations.
An important study, which defines the level of sub-contract of user decision

making process to RSs, claims that considering experience products, consumers may accept more a help in cognitive process as a consequence of increasing need for information [20]. As a consequence of this theory, users may rely more on other-based decision-making processes than own-based, with the result in an increased adoption of RSs support.
The second factor, namely the complexity, is defined in multiple ways. As the authors suggest [46], the complexity can be defined as *number of product attributes*, *variability of each product attribute*, and *interdependence of product attributes*, in addition to the *inter-correlation of attributes*.
The first dimension refers to the number of attributes possessed by a product; the second refers to the number of values each attribute can have; the third is referred to the relation between each attribute (e.g., when an attribute is changing, another may suffer this variation); the last dimension indicates whether at an attribute variation, corresponds another variation in inverse way (e.g., if one attribute increase its value, the other one decrease etc ...)
As showed by Xiao and Benbasat [46] in their work, the high complexity of products reflects in a more heavy process of search and evaluation of alternatives. Therefore consumers exploiting RSs support can improve the final decision quality and reduce their effort.

User-RS interaction As discussed by Fogg et al. [13] and previously analyzed in the above paragraph, RSs must demonstrate similarity to their users, that is, they must internalize users' product-related preferences and incorporate such preferences into their product screening and sorting process. Xiao and Benbasat argue that searching and presenting recommendations following consumers preferences and tastes are perceived as more familiar and their "personality" more similar to the user.
Hence this kind of interaction leads to improve the predictability of the RSs' behavior and to focus users' attention on more attractive product alternatives, thus resulting in improved decision quality and reduced decision effort in online shopping tasks.

Relationship constructs - users' evaluation of RS

RS use According with Ajzen et al.[3], an individual's descriptive beliefs about an object can be formed through direct experiences with such object. Therefore the authors pinpoint that previous or repeated experiences with the RS can serve as a basis for the formation (or update) of user evaluations (i.e., usefulness, ease of use, trustworthiness, and satisfaction) of the Infor-

mation System at a subsequent stage.
In the context of RS-assisted online shopping, the use of RSs is expected to influence consumers' evaluations of the RSs.

RS type The influence of RS type on the evaluation of the RS itself concerns the tradeoff between trust and confidence of the user towards the system, and the effort it is due to express needs and preferences.
Xiao and Benbasat pinpoint that while hybrid or knowledge-based approaches may receive a better feedback in terms of recommendations and therefore as level of trust, confidence, perceived ease of use from the user, on the other side there may be an increased effort to highlight preferences and needs. Such systems in fact need more user interaction, as a consequence of the need to know detailed aspects of the problem domain [8], [22]. The result of such interactive aspects may cause an increased perception of effort to reach the goal by the user. Hence a decrease of user satisfaction [46].
Therefore this construct gives the opportunity to compare between different types of RSs and their interaction in order to evaluate their impact in terms of user's satisfaction and perceived usefulness and ease of use.

RS input characteristics This dimension is evaluated by the means by which users' preferences are elicited, the ease for users to generate new or additional recommendations, and the amount of control users have when interacting with the RSs' preference elicitation interface.
Because of users reluctance to extend their cognitive efforts, the model enables the evaluation of the tradeoff between elicitation techniques explicit versus implicit. In the first case the user must interact with the system to let the latter collect data on the user. In the second case the system will implicitly examine the user behavior in order to infer needs and goals.
Another aspect to take in consideration is the ease of generating new recommendations.
Under the assumption that user may change his/her mind, the more the system is able to provide new recommendations the more the user will feel satisfied in its usage. If the system will make the user repeat the whole elicitation process to generate new recommendations, the user will perceive the system as *restrictive*.
In order to obtain a good result in terms of flexibility's perception, RSs should support a multitude of decision models that the decision maker might choose to employ. In exchange, users will feel a more active role, improving their perception of *control over interaction* [46].

RS process characteristics This construct highlights how the relationship between the RS process of generating recommendations and the user's evaluation of RS itself can be referred to two parameters:

- *the information about search process*
- *the response time*

In the context of RSs support, the computation load is shifted from the consumer side to the system side, as a consequence of the assistance role performed by the system.
While users are reluctant to increase their cognitive efforts, they generally welcome other efforts [19]. Hence a positive influence on the evaluation of RS can derive from showing the search process status to highlight the load and the effort made by the RS. This effort can influence users about the goodness and expertise of the system and therefore influence their evaluation.
The second parameter (i.e., the system response time) is defined as the time between the user's input and the computer's response. It has been widely recognized as one of the strongest stress factors during human-computer interaction.
Assessments of the effects of response time have been conducted for personal computer use in a variety of contexts. Long response time increases stress levels, self-reports of annoyance, frustration, and impatience of personal computer users. According with Xiao and Benbasat [46], response time influences users' perception of system quality and thus their satisfaction with information systems.

RS output characteristics The RS output characteristics which influence the user evaluation of RS are the familiarity of the recommendations, the amount of information on recommended products, and the explanations on how the recommendations are generated.
The first characteristic, i.e. familiarity, is related to knowledge-based trust theories [26] which are based on the principle that users develop trust on systems over time as they accumulate knowledge. Therefore as the predictability of the recommendations grows, the user will consider the system more familiar and this improves user's trust more than those systems that provide unfamiliar or novel recommendations.
However it is part of a RS to produce "unfamiliar" alternatives, as this factor falls into the novelty of recommendations. The main issue is to present familiar items together with less familiar one, trying to prevent rejection from the user. In this way recommendations of familiar products can serve

as a context within which unfamiliar recommendations are evaluated, hence improving the attractiveness of the unfamiliar recommendations and the evaluation of the RSs.
The second aspect involving the influence on RSs evaluation is the content and the layout with which recommendations are provided.
To support users with detailed informations, content (e.g., product descriptions in text or multimedia format, expert reviews, or other customers' evaluations), can signal to the users that RSs care about them, act in their interests, and behave in an honest and unbiased fashion, thereby contributing to users' assessments of the RSs' benevolence and integrity [46].
Another aspect to consider is that providing accurate information can educate the user on the domain, thus contributing to the users' perception of the RSs' usefulness. Detailed information also promotes users' perception of RSs' information quality, thereby enhancing their satisfaction with the RSs.
The last dimension of this construct analyzed is the output format of recommendations: given the reluctance of users to extend their cognitive efforts, everything which is perceived as not clear or negatively evaluated, in terms of navigation or layout of the presentation of recommendations, is perceived as inconvenient to use.
Besides, RS benefits can be underestimated if not supported by efficient design strategies.

Product-Related Factors The main product factor concerning the user evaluation of RS is the product typology. As mentioned above, product can be defined as a search product and an experience product. While search product refers to a type in which qualities are well defined and understandable on their own, the experience one needs more assessment to be understood.
In this context RSs can be perceived as more useful in the case of search products, because these can be reduced to images, text end property easily manageable in a search activity. In the case of experience product, users are not able to manage and understand all the information related to these kind of products and they are more favorable to other-based decision process.
As a consequence they will rely more on RSs recommendation, developing a stronger sense of trust.

User-related factors Whith this dimension, Xiao and Benbasat explore the level of expertise of users and argue how such parameter can influence their evaluation of a RS.
The main principle which lead to introduce this dimension in their model is

that users with different level of domain expertise have different degree in searching and comparing many products. In detail, users with higher expertise are more likely to compare better different features of more alternatives, while not expert ones are searching for more general comparative measurements [33].
In this context, the characteristics of RS are fundamental for user evaluation; for example expert domain systems (such as knowledge or content based RS) can be perceived more restrictive by expert users while they can be appreciated from less expert users because they can identify simply their needs and communicate with RSs.
In the same point of view, RSs of such type can obtain a negative response from experts because these latter are not able to compare individual qualities as they want. On the other side, detailed information or responses from the system can be perceived less useful from not experts who may desire more needs-based answers.

Factors related to user-RS interaction In this construct, it is identified the similarity between the RS and the user in terms of predictable behavior of the system as the main factor of influence towards RSs.
The similarity between the system and the user is a concept well formalized in the *unit grouping* theory [27]: "a cognitive categorization processes individuals use to develop trusting beliefs in new relationships." Individuals who are grouped together tend to form more positive trusting beliefs about each other, because they tend to share common goals and values. Therefore grouping together the user and the RS, i.e. the user perceives the similarity of the system with respect to the comprehension of the generated recommendations and the possibility that the user expects what RS will provide, improves the perceived usefulness of the system itself.
This aspect is considered also in the enhancement of user's trust towards the system.
Another correlate dimension is the *confirmation/disconfirmation paradigm*, which establishes a relationship between user satisfaction, and the level of confirmation/disconfirmation of the expectations.
Confirmation occurs when perceived performance meets the expectation. Positive (negative) disconfirmation occurs when perceived performance exceeds (falls below) the expectation. Satisfaction is achieved when expectations are confirmed.
As a conclusion of their work, Xiao and Benbasat draw some guidelines to identify further relationships between constructs of user's evaluation of RS and between the decision making process and the user's evaluation of RS.

These guidelines pinpoint the correlation between how the quality of the decision process and of the outcome can influence the entire perception of the system. Moreover it is identified that a perceived satisfaction can lead to an increase in the perceived ease of use and consequently in the perceived usefulness and trust (these both are also correlated variables).
Because of lack of experiments and empirical proofs, no proposition or statements are argued about these considerations.

2.4 Recommended systems in the tourism domain

The context of this research is the e-tourism, the digitalization of all the processes and value chains in the tourism, travel, hospitality and catering industries that enable organizations to maximize their efficiency and effectiveness [6].
In this area, the greatest efforts produced by stakeholders such as Expedia.com, Venere.com, Booking.com, attempt to improve the understanding of how sensory elements of web sites can make an impact on the experience, attitude and behavior of consumers.
Furthermore it can be assumed that the interaction between potential travellers and tourism providers' or destinations' websites play an important role in shaping the traveller's expectations, attitude and behavior toward the tourism products, providers and destinations.

Beyond these factors of interest, it has been proved how the relatively high frequency of destination-information search across different personality traits implies that destination information is a universal type of information sought in travel information searches [12].
In this perspective RSs, with their ability to fit those searches with a personalized user profiling, can enhance performances of online travel agencies (OTAs), with increasing results in overall satisfaction of the whole e-commerce process and, as a consequence, in the economic performance [43].
Beyond these aspects, despite the fact that consumption and decision-making processes related to tourism both are considerably driven by hedonic and emotional aspects [10], the significant role of sensory elements has been broadly neglected in tourism marketing. Therefore the need to conciliate performance strategies, i.e. the correct application of RS technology, with design strategies is fundamental to address the issues of the e-tourism context.
Our research is carried on with the collaboration of Venere.com, a company of the Expedia Inc., group which is leader in online hotel reservations market featuring more than 120,000 hotels, Bed and Breakfasts and vacation rentals

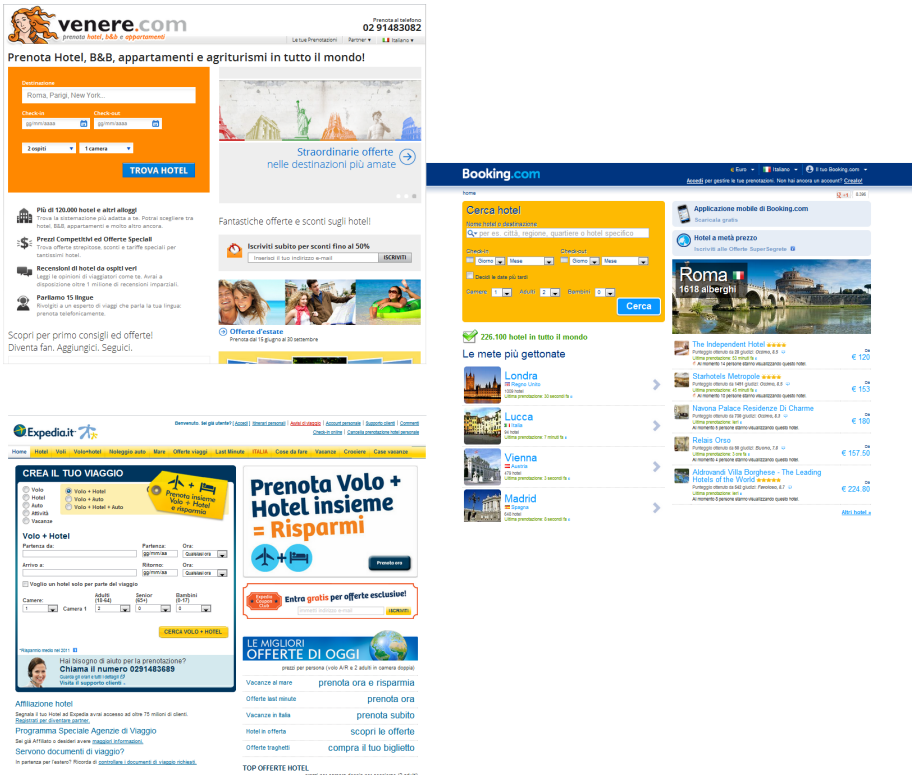

Figure 2.5: Venere.com, Booking.com and Expedia.com: the three main OTA

in 30,000 destinations worldwide.
The main thematics faced by Venere.com, as well as other OTAs, expand in three different areas:

- *offer more personalized booking items to consumers*

- *address the problem of bounded resources*

- *facilitate the integration in their interface of new strategies of proposals*

The first issue is clear since all OTAs try to maximize their gain and therefore, as previous studies and experiments confirm, the possibility to introduce a personalization on the proposals to consumers may lead in an enhancement of economic performances. This is especially true in a domain where the enormous number of items is in contrast with the fact that consumers' final decision is based on a limited set of choices.
Another consideration which should be done at this point is that it is common for an OTA to interact with unregistered users: people are reluctant to register or provide information about themselves so the web applications must face the problem of offering the best alternatives with no information about the consumer.
The second area refers to the well known problem of *bounded rationality*, i.e. the unavailability of the desired product under specific constraints.
In this issue our partner identifies that the *user give-up*[12] parameter during the search of a reservation is strongly influenced by the fact that a chosen structure, i.e. hotel, bed&breakfast, residence etc ..., can be unavailable for a certain period.
The integration of sales strategies and interface components has always been a crucial aspect for the e-commerce, in particular for OTA companies. In these context there must be found a balance between components' factors such as text, images, multimedia, and signals from the provider[13] with how and in which area to introduce the proposals for consumers. This can be addressed into the field of design strategy to verify how the layout and the presentation influence users and their purchase intention.
An example is well represented in the decision to integrate different kind of proposals in a dedicated area in a website or rather integrate them between other elements of the page.

[12] It is a parameter which indicates the degree of booking abandonment

[13] Signals are interpreted as OTAs communicational strategies as for example discount labels which indicate a promotion, or a scarcity message which urges consumers to book the last room remained

In these contexts, a designer's goal should go beyond the support to findability - to enable users easily locate what they are precisely looking for - and to the tasks involved in the decision making process as it is conventionally intended. RS design should also support tasks that are essentially explorative in nature [25], [19], and are oriented towards constructing preferences in the specific domain and decision environment.
A challenge is to provide a seamlessly integrated set of interactive design strategies that leverages existing patterns of exploratory interaction, such as faceted navigation and search [25], with existing RS design strategies. It is worth noticing that serendipity can be an important goal also in this exploratory phase and not only when providing recommendations.
Promoting crucial contents the existence of which users did not even suspect, so that users can stumble and get interested in them (even if they were not looking for that kind of information), can be as effective (or perhaps more effective) in this phase than in following phases of the decision process.
In this context some research studies have proposed different web applications which exploited RSs to offer more than a pure booking functionalities. We will introduce now the most considerable works on RSs in the e-tourism domain and their main characteristics which are taken into account for our research.

2.4.1 User model typology

A first approach to analyze the existing RSs in the e-tourism is to consider how they differ in personalizing recommendations.
For a system to be able to provide personalized recommendations it should make inferences about the users' preferences. Such information as well as information about the users' previous experiences is stored in a user model. Indeed recommender systems offer guidance based on users' profiles or visiting background. Therefore, every recommender system builds and maintains a collection of user models [18].
The main dimensions that characterize user models are:

- One model of a single, canonical user vs. a collection of models of individual users
- Models specified explicitly either by the system designer or by the users themselves vs. models inferred by the system on the basis of users' behavior.
- Models of fairly long-term user characteristics such as areas of interest or expertise vs. models of relatively short-term user characteristics

such as the problem that the user is currently trying to solve.

Single user model vs. Stereotypes

Taking into account the first categorization of user models, a recommender system may maintain an individual user model or some user models that represent classes of users.
When commercializing complex customizable products online, there may be various classes of users of the configurator that differ in properties such as skills, needs and knowledge levels. These classes are called stereotypes.
Some examples of these models can be found in the Trip@dvice platform and the Traveller RS.
The first, i.e. Trip@dvice, is a flexible software tool that can be integrated in existing tourism portals to support the user in his/her trip definition tasks. The tool allows the visitor to put together a tailored travel package, choosing a hotel, places to visit, things to do or activities to practice. The user profile is expressed as the combination of all these typologies of user models.
Trip@dvice exploits collaborative features during the conversational phase to gather information about the general characteristics of the desired travel and the stereotype class user belongs to; at the same time it provides, during next steps of interaction, the possibility to select the content features of the items, building therefore the content portion of the user model.
Finally, as it is a CBR, it exploits the case stored in its memory to compare preferences of the actual user preference to build the short-term (i.e. contextual) model.
Another example of this mixed strategy for user modeling is the Intelligent Travel Recommender (ITR), where a similar procedure is engaged with the user in order to build a multi faceted user model that exploits all the features of collaborative, content and stereotype classes properties.
Traveller RS is a system which offers the possibility to pack multiple activities into the "Holiday package". A Holiday package is a composition of tours and its characteristics are the name, the modality of the trip (e.g. honeymoon), the departure date, the price per person, and the duration (in number of nights and/or number of days). The characteristics of a tour are: category, place or destination, price, means of transport, transport company (e.g. name of the airline), accommodation (e.g. name of the hotel), type of accommodation (hostel, 5 star hotel, 4 star hotel), type of room (single, double; standard, suite), duration, and type of service.
Traveller RS encloses the actions of multiple agents specialized into the processing of content, collaborative, and demographic information. The content

information exploited to build the user model is the knowledge about the user's traveling preferences obtained from his/her purchases and complaints. The collaborative user profile is built from the ratings given by other users to the tours the user has taken.
The demographic profile simply contains demographic information about the user.
Another system which we take into account into our analysis is MastroCARonte RS [23], which is a multiagent system which provides a car driver with personalized tourist information.
The system identifies the user and tailors information according with his/her model and to the contextual conditions. As it is possible to understand, the user profile is build on contextual information, which is necessary to locate the car and the moment of recommendation, and on long term user characteristics expressed as content constraints.
In fact MastroCARonte offers recommendations about hotels, restaurants, and places of interest and geo-politic features. Its dataset is composed by content features of the aforementioned touristic areas. Of course, thanks to GPS and other geolocation technology, the system builds the contextual portion of the model, in association to the content part which is created by storing user preferences about suggested items during the sessions.

Implicit vs. explicit elicitation techniques

The second dimension identified involves the way of information acquisition for the user model.
Information about the user may be acquired explicitly or may be inferred implicitly from the user's previous interactions or both.
Enabling consumers to develop their online profile and to include personal data that indicate their reference can support tourism organizations to provide better service [6].
The main problem with explicit user models is that users may have to answer too many questions. Furthermore, users may not be able to describe themselves and their preferences accurately. In previous works, implicit user modeling has been considered as more reliable and non-intrusive than explicit user modeling. However, one main problem of this approach is that the hypotheses generated by the system for each user may not be accurate. Furthermore, there may not be sufficient time for the system to observe the user for producing accurate hypotheses about him/her.
After the overall description of advantages and disadvantages of both tech-

niques, we provide some examples of their application into RSs.
As a first example we consider DieToRecs platform [16]. This RS provides both techniques of elicitation: while explicit elicitation is obtained by exploiting the conversational nature of the system during all the user session, implicit elicitation is gathered during the selection of the items as a preference signal. In the first case, i.e. explicit elicitation, the system asks users to provide preferential features to trace the recommendation problem (for instance, the nationality of the user or the travel purpose). This information is managed to build the collaborative portion of the user model.
In the second case, the implicit elicitation technique is exploited when the user selects a package suggested and the system saves the content preferences expressed with this choice.
Also MastroCARonte is an expression of the joint use of these two techniques. The explicit elicitation is exploited by a feedback agent that translates actions of the user respect to the device into a feedback (turning off the voice of the PDA or switching off the monitor etc...).
The implicit elicitation is obtained by some inference on user behavior and by geo-location instruments: if the user ignores the suggestions of the system about an interesting location when he/she is near, then this can be interpreted as negative feedback.
An example of a RS which is using only an implicit elicitation technique is the Personal Travel Assistant (PTA) of Coyle at al., which is used for reserving and selling flights. It is one of the first mobile recommender systems CBR based. This RS enables flight reservation with a desktop application and also mobile devices. The system also offers different packages beyond the flight booking such as car rental, and post flight services. These packages are obtained through proxy towards travel agencies and then reordered following a priority list of user preferences and needs.
The user model of the PTA is based on the progressive gathering of information about the characteristics selected for the flight. The content selected through the PDA or mobile phone is used to retrieve informations from web services; the system then uses some similarity metrics to infer about what flights fit the best with the user profile basing on the features of such flights and related extra packages.
This setting in the elicitation technique is in part due to the fact that on mobile devices there is a little possibility for interaction. Therefore the system is limited to capture only implicit signals.
Others RS that exploits exclusively the implicit elicitation technique are the Entree and Recommender.com [7], systems which are based on the FindMe platform. These systems provides a series of implicit signals which are con-

sidered the ratings of the user respect to an item.
The two ways provided by this platform to gather the implicit signals are on one side the direct choose of an item into a catalog and on the other hand a selection of constraints through some filters which depend on the characteristics of the domain.
For examples for the Entree system, which suggests restaurant close to user, the main features to be filtered are the city, the area, the type of cuisine, the price, the style, and the occasion.

Short vs. long term user profile

Finally, the last dimension involves short-term vs. long-term user models but all recommender systems maintain long-term user models as the previous interactions of the particular user or other users with similar characteristics are essential for content-based, collaborative or demographic filtering. Almost all tourism recommendation systems use long-term user models, as this is considered more effective.
The contextual information used in those systems and the complexity of the item and the type of recommendation, make the context a key aspect which must be taken into account. This is the case of DieToRecs system and MastroCARonte.
In the first, the bundle of services makes it infeasible to provide recommendations basing only on long term preferences, as the bundle is composed by different heterogeneous components (e.g. a flight, an accomodation, and a car rental) and the consideration of the context is essential for the correct functioning of the application.
Beyond if there are both long and short term profiles available, the precedence for recommendations goes to the short term which is more exploited (e.g. items are ranked more on top if related to contextual profile).
In MastroCARonte the situation is even more critical in the sense that it is impossible to exclude the contextual profile: all the recommendations are based on short term information such as if the user is approaching a town, it is late in the evening, he/she has been traveling for some hours and he/she is not headed home, the query management agent may initiate a query for asking about restaurants or hotels. Alternatively, if the user is moving to the center of a town, the system may initiate a query to check whether she is close to a monument that may be of interest for her.
These two types of recommendations are based on the contextual information like “it is late in the evening and the user didn’t go home yet” or “the

user is close to a town" etc...

2.4.2 Recommendation techniques

Another characteristics related to existing RSs is the variety of approaches that have been used to perform recommendations in this domain. These include content-based, collaborative, demographic, knowledge-based or hybrid approaches.
The two approaches that are more popular are content-based filtering and collaborative filtering. Content-based filtering refers to recommending items or services based on analysis of the user's previous actions or purchases, while in collaborative or demographic filtering, the items are recommended based on the recommendations of other users.

Content-Based recommender

Content-based filtering suggests to a user, items or services which are similar to those s/he bought or searched in the past, by matching the characteristics of the item or service with the characteristics of the user that are maintained in his/her user model.
For example, MastroCARonte and ITR [15] use a content-based approach, in which the user expresses needs, benefits, and constraints using the offered language (attributes). The system then matches the user preferences with items in a catalog of destinations (described in the same language).

Collaborative recommender

Collaborative filtering, also known as social filtering, focuses on the behavior of users towards items or services, such as purchasing habits or preferences, rather than on the nature of items or services the system offers.
In systems that use collaborative filtering approaches, recommendations are made by matching a user to other users that have similar interests and preferences. An example of a system which uses this recommendation strategy is Traveller [38], where each user is suggested with items or services that other users, similar to the one interacting with the system, have bought before.
For this purpose, every time a new user is logged on to the recommendation system, the user has to be connected to those that appear to be closer to his/her interests.

Hybrid approaches

In view of the disadvantages of the content-based and collaborative filtering, many researchers have proposed hybrid approaches, where content-based and collaborative filtering methods are combined in order to exploit their advantages and reduce their deficiencies.
For example, the restaurant recommender proposed by Burke makes recommendations by finding restaurants in a new city that he/she will travel similar to restaurants the user knows and likes in his/her own town based on the knowledge of cuisines to infer similarity between restaurants.

2.4.3 Recommendation moment and format

Two important dimensions to study existing RSs are the moment and the format of recommendations. These two important factors indicate when to introduce the results of RSs during the user-RS interaction and what to present to users and the number of results.
Two clear examples of recommendation moment and format are showed in DieToRecs and Recommender.com systems.
DieToRecs provides an interaction with the user in which there is always the support of the RS, as the recommendations are always provided in some areas of the interface. The only moment when items are not presented to the user happens at the very beginning of the interaction, where the system provides a form with a set of specific questions oriented to understand the main collaborative aspects of the actual user. In this screen the user gives the system information related to the location of the travel, the type of traveller (e.g. single, with family etc...), the duration of the travel, the budget, and the period (e.g. Christmas of summer holiday). Beyond these interactive aspects, the system enables two different ways to support the user: a "single item recommendation" and a "seek for inspiration" mode.

In the first mode of functioning, the system redirects the user to a catalog of travel packages. This functionality is designed for a user who has some rather precise needs, who wants to search the available options driven by these requirements. The user can modify the constraints of some filters to limit or to increase the number of considered items. These filters modify the type of sport activity (e.g. skiing, bike, trekking etc...), the social and cultural features of the travel (e.g. museum, shopping etc...), and other parameters, if present, like leisure and relax. Beyond this, the system let the user specify again the basic parameters of the travel like the price, the city etc...
In this case the recommendations are obtained on the content features pro-

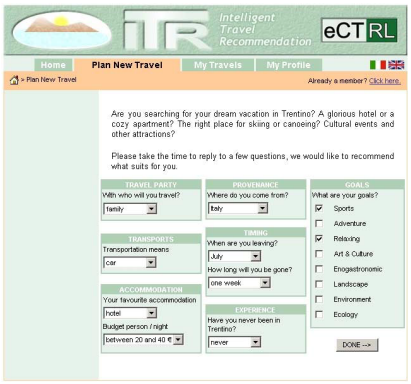

Figure 2.6: An example of ITR conversational interaction (based on the same RS of DieToRecs)

vided and on the items selected. The travel packages are presented as a list with a thumbnail and a textual description. The second way of functioning,

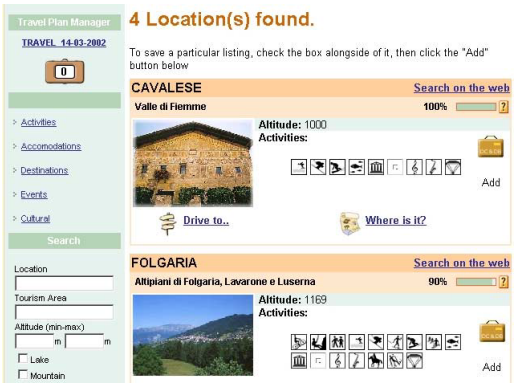

Figure 2.7: ITR recommendations

i.e. seek for inspiration, is designed for users who would rather browse the options and get inspired by the alternatives before taking some decision and focus on some particular products.
DieToRecs provides a limited set of six predefined scenarios of travel. These holiday packages express in a more detailed way the features of the travel than the single item recommendation. It is provided a wider thumbnail and some features of the travel (e.g. the destination, the price etc...). This way the user can provide a rate ("i like" link) to suggest the system that some properties interest him/her and receive as a consequence another set of six recommendations.

To maintain flexibility the systems let the user free to skip between one recommendation style and the other at anytime in the interaction. After clicking on the link of an item, the user can go down another level in the navigation to the detail page of each package.
Another style of recommendation is provided by the Recommender.com system (as well as Entree and other RSs based on FindMe platform). This system implements two different retrieval modes: the "similarity finding" and the "tweak" mode.
In the similarity case, the user has selected a given item from the catalog (called the source) and requested other items similar to it. To perform this retrieval, a large set of candidate entities is initially retrieved from the database. This set is sorted based on similarity to the source and the top few candidates returned to the user.
Tweak application is essentially the same except that the candidate set is filtered prior to sorting to leave only those candidates that satisfy the tweak. For example, if a user responds to item X with the tweak "Nicer", the system determines the "niceness" value of X and rejects all candidates except those whose value is greater.
The recommendation integration strategy of this RS is similar to the seeking for inspiration of DieToRecs: the screen presents a small set of recommendations but with more descriptive content. In fact in Recommender.com there is the title of the movie recommended, the genre, the director, the trailer with the possibility to deepen the information about the movie by clicking on some links.

Figure 2.8: Entree recommendations

2.4.4 Evaluation approaches

Empirical evaluations are needed to determine which users are helped or hindered by user adapted interaction in user modeling systems. The key to good empirical evaluation is the proper design and execution of the experiments so that the particular factors to be tested can be easily separated from other confounding factors.
However, empirical evaluations are not so common in the user modeling literature. Many researchers have also stated that it is important when evaluating adaptive systems to assess whether the system works better with the user modeling component as opposed to a system deprived of this component.
While an evaluation framework for RSs has been introduced in the previous section, we introduce now three examples for which and evaluation of RS action is provided.
The first evaluation example is the Coyle and Cunningham RS, which use simulating interactions between the PTA [11] and the user exploiting the system's history of user interactions in order to evaluate their approach.
The system stores all the interaction with the user during the interactions in a first phase. Subsequently, to evaluate some changes or variation in the system, exploits another session with other uses and compare such interactions sequences with the stored ones. Finally an evaluation is inferred basing on the comparison of different sessions interactions.
Another example of evaluation of RS from the perspective of the user is adopted for the ITR system [29], a multi-agent planning system. More specifically, the evaluation is based on two variants of the system, namely ITR+ and ITR- : in the ITR+ are provided all the features of the RS developed while ITR- is a baseline version, without the interactive query management and the ranking functions. Thus ITR- is not able to suggest how to change a query, and it does not provide any smart sorting of the selected products: the result order mirrors the order of the items in the product catalog. In both the system variants each result page displays three items.
The evaluation is accomplished after an empirical study where two different groups of people (students and employees of Trento University) receive a common task: plan a vacation in Trentino, selecting a set of travel products and putting them in a virtual repository (travel bag).
The two groups are not aware about the difference between the two systems (i.e. they consider the two systems the same) and, after a period of pretrial (about 10 min) they accomplish their task and complete a final evaluation survey.

An important example of RS evaluation is performed on DieToRecs prototype [4]. The evaluation procedure considers two basic forms of data that will be collected: objective and subjective measures.
"Objective" data include the tracking of data using the system log functionality. Data about the user input, the information displayed, the interaction with the recommendation functions, system errors, performance and time measures are collected. The metrics are showed in the following table.
As for the ITR, the people involved with the RS evaluation experiments per-

User Input and Search
the number of queries
Interaction with the Rec. Functions and System Errors
number of suggested relaxation operations
number of system errors
Information displayed
number of results for each query
Performance and Time Measures
Total task time
Number of items added to the travel plan

Table 2.6: Objective measurements

formed at first a pretrial phase, where they are supposed to take confidence with the system. Afterwards two different variant of RS are tested. The last activity performed by the user is a paper-and-pencil survey to evaluate the RS in under a more subjective point of view. The questionnaire is prepared with the aid of different surveys used in the past and several theories, the most important PSSUQ - The Post Study System Usability Questionnaire - and SUMI aforementioned.
The final survey focuses on different areas such as design and layout, functionality, ease of use, learnability, satisfaction, outcome and future use, and system reliability. An example of the survey is provided below: The last example of evaluation of RSs is the Travel Planner, a recommender system integrated in the Austria tourism portal. This system belongs to the conversational recommender system family: the needs and preferences are discovered explicitly during a continuous interaction with the user which can be through filters, questions or more interactive forms (pictures, text etc . . .)
The novelty of this system is the strategy with which it learns preferences and relax constraints: the *reinforcement learning* is obtained through a *Markov Decision Process* (MDP).
This process is defined by a set of system states with significant variables;

Design / Layout
I liked using the interface of the system.
The organization of information on the systems screen was clear.
The interface of this system was pleasant.
Functionality
This system has all the functions and capabilities that I expect it to have.
The information retrieved by the system was effective in helping me.
The list of item recommendations supported travel planning a lot.
I would have missed the function "complete travel recommendations" a lot.
Ease of Use
It was simple to use this system.
It was easy to find the information I needed.
The information provided with this system was clear.
Overall, this system was easy to use.
Learnability
It was easy to learn to use the system.
There is too much information to read before I can use the system.
The information provided for the system was easy to understand.
Satisfaction
I felt comfortable using this system.
I enjoyed my travel planning session with this system.
Overall, I am satisfied with this system.
Outcome / Future Use
I was able to complete the tasks quickly using this system.
I was able to efficiently complete the tasks using this system.
I could not complete the tasks in the preset time frame.
I believe I could become productive quickly using this system.
The system was able to convince me about the goodness of the recommendations.
From my current experience with using the system, I think I'd use it regularly.
Errors / System Reliability
Whenever I made a mistake using the system, I could recover easily and quickly.
The system gave error messages that clearly told me how to fix problems.

Table 2.7: Subjective measurements

set of system actions; a reward for the reinforcement learning; a transition function which means the probability to move from one state to the other. The system is implemented in two functioning modes and in two variants: the two mode are the training phase and the testing phase. The two variants are the variant train and the variant test. In the training phase are evaluated different systems parameters to be experimented as systems reaction in the training phase. Afterwards the best achieved policies are utilized into the test phase.

Chapter 3

Proposed methodology

In this chapter are described the main issues and the proposed solutions of our work, in the context of the e-tourism domain.
It is provided at first a portrait of the encountered complexities, as well as the main research questions of our work.
Three extensions of the Xiao and Benbasat conceptual model are introduced, in order to give a formal method of analysis for the argued thematics and for a general approach with RSs. Finally we discuss the main metrics to perform an evaluation of every logic construct of the model.

3.1 Challenges for RSs

Let us consider the following scenarios, in which the user is engaged with an online hotel reservation system.

1. *You have to come to Milan and work with your business partners from August 6 to August 10, 2012. You want to reserve a room in a hotel in Milan for that week.*

2. *You will spend a holiday in Milan from September 19 to September 25, 2012, and want to reserve a room.*

3. *You have to attend a business meeting in Milan from September 19 to September 20, 2012, and you need to reserve a room in a hotel in Milan on that dates, for one night.*

4. *You are planning a holiday in Central Italy in mid September 2012, and will visit Rome for few days. You need a hotel in that period.*

How do the above scenarios differ?
In all of them, the user is doing a similar operational task: buying a service, specifically, reserving hotel rooms. Still, there are some significant differences that may influence the decision making processes, and are induced by *i) the different nature of the user's goal; ii) the dynamic nature of the services offered by the system the user is interacting with.*
In scenarios 1, 2 and 3, user's goals are sharp, users' preferences are well defined and have clear-cut criteria for their optimal satisfaction. In scenario 4, the user have less strict preferences - his/her dates are "flexible", and we may not exclude that he/she is flexible also with respect to other criteria, or may not know all her preferences beforehand.
Preferences are likely to be shaped and changed throughout a session in the specific decision environment. Using the terminology of goal oriented requirements engineering [44], scenario 4 depicts a situation that is characterized by soft goals, i.e., open-ended needs that are progressively elaborated during the interaction with an environment and the decision process, and may be somehow supported by one or more combinations of solutions and decisions.
Further differences in the above scenarios are related to the intrinsic nature of resources, in particular, to the dynamic, time dependent characteristic of the items in terms of their availability.
In scenario 1, the user is making a decision in the context of a very vast set of stable alternatives: in the second week of August, hotel availability in Milan is huge, as most people and companies or institutions are on holiday. No matter when and how you reserve a hotel, it is very likely that you will find one that matches you preferences. In contrast, in scenarios 2, 3, and 4, the user is taking decision in the context of limited or very limited resources, or of resources that become limited, or even fully unavailable, as the decision process proceeds.
In scenario 2, the user is looking for hotels in a period - from September 19 to September 25, 2012 - when Milan will host one of the most important international events in the fashion world, the Milan Fashion Week, attracting thousands of people from all over the world. Most hotels are booked one year in advance for that event. Hence, we can reasonably expect that, when searching a room for the whole week, no hotel is available.
Scenario 3 considers reservations in the same period of time, but here the user's requirement is less demanding - he/she is searching a room only for the first day of Milan Fashion week. There might be rooms available on that single date. Still, it may happen that other people are simultaneously trying to make a similar reservation, so that when the user takes her decision, the chosen hotel is not available any more.

In scenario 4, the user hasn't decided yet when he/she exactly will go to Rome, and his/her dates are flexible. It is likely that he/she has not specified the reservation period at the beginning of the process, and finds many alternatives matching his/her preferences on hotel characteristics. Still, the preferred time frame for reservation - mid September - is high season in Rome, and finding a hotel in that period time may be difficult. When he/she makes a specific choice, decides the dates and attempts to make a reservation, the selected hotel may result to be fully booked.
In all contexts depicted in the above scenarios, the user is facing a problem falling in the class of so called "preferential choice problems" [48], i.e., he/she needs to take decisions across an initially vast set of potential alternatives.
In this context, decision making processes are typically modeled as "bounded rationality" phenomena [39].
Bounded rationality - which provides a key theoretical underpinning for RSs - is the notion that, in complex decision-making environments, individuals are often unable to evaluate all available alternatives in great depth prior to making their choices, due to the cognitive limitations of their minds, and the finite amount of time they have to make a decision; hence they seek to attain a satisfactory, although not necessarily an optimal, level of achievement, by applying their rationality only after having greatly simplified the set of choices available.
Several authors suggest that the cognitive effort can be reduced with a multiple-stage decision-making process, in which the depth of information processing varies by stage [46].
Initially, individuals screen the complete solution space (e.g., the set of all hotels featured by the on-line reservation service provider) to identify the (large) set of potential alternatives, or search set (e.g., the set of hotels that could be of some interest); then they search through this set, and identify a subset of promising candidates (the consideration set). Subsequently, they acquire detailed information on selected alternatives to be seriously considered (in-depth comparison set), evaluate and compare them in more detail, and finally commit to a specific choice. Although some of the above actions can be iterated, this process is intrinsically linear and it is likely to end with the user making a specific choice and hopefully buying a service.
The same process may not apply exactly in the same terms in the situations described in scenarios 2, 3 and 4 (3.1). In scenario 2, the search set is likely to be empty (no hotel is available for the specified period). In scenarios 3 and 4, the search set, the consideration set and the in-depth comparison set are not empty, initially. Still, their size decreases as the decision process proceeds (e.g., because other users buy some items, or because the user refines

her decision criteria, e.g., fixing the dates). Hence, when the user reaches the final step and makes a decision, her choice will likely result unfeasible.
In all these cases, after experiencing the unavailability of resources, i.e., of rooms in the desired hotel(s), the user may either give up (e.g., he/she leaves the current on-line reservation service and tries a different one) or iterate the process, providing extra input to modify their preferences, exploring the search set, consideration set and in-depth set again, and attempting to make a different decision.
The examples discussed in the previous section highlight that the decision process in RSs is influenced by the characteristics of both users' goals and the resources meeting users' needs and preferences.
How the nature of the goal (sharp or soft) and the dynamic of resources play in the decision making process has been marginally explored in current RS research, and opens a number of research challenges.
A first challenge is to understand the degree at which some key theoretical

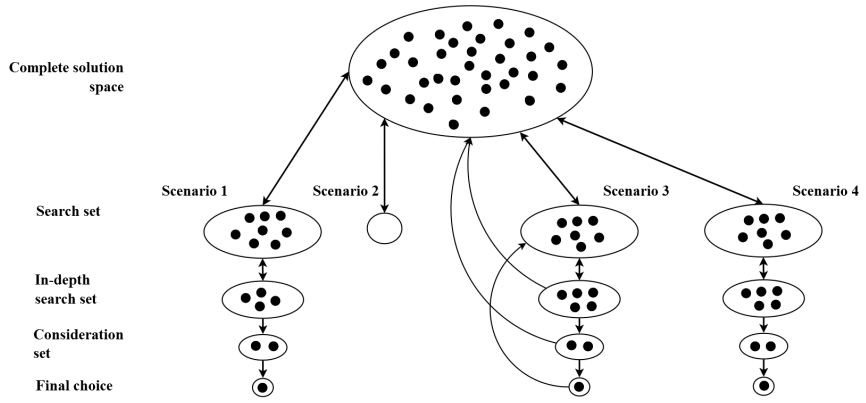

Figure 3.1: The dynamic user decision making

assumptions underlying most of the existing RSs, such as "bounded rationality", are valid in the context of users' soft goals, and how the structure of RS supported decision making processes can be defined in these situations.
On the one side, it remains true also that a decision-maker lacks the ability and cognitive resources to arrive at the optimal solution in a vast set of alternatives, and at some point of time he/she needs to apply his/her rationality after having greatly simplified the choices available.
On the other side, the decision-maker might not be modeled as a "satisfier" - one seeking a satisfactory solution rather than the optimal one, minimizing the cognitive effort - along the entire decision making process.

At the beginning of the process, the user may indeed be looking for an optimal solution, because his/her needs and preferences are initially poorly defined, and he/she does not know yet what the characteristics of such optimal solution are. Hence the initial step of the decision process is more a kind of "sense making" activity than a focused "search": the user is attempting to understand the complexity of the domain and the characteristics of the items in relationship to the specific field of interest, in order to decide what he/she needs and wants.
In this context, the decision making process seems to include a preliminary phase, taking place before the progressive elaboration of alternatives, in which the user forges her own preferences, and transforms a soft goal into a sharp goal that characterizes an actual preferential choice problem. In this preliminary sense making phase, optimizing cognitive resources and reducing effort might not be an issue, as suggested by some studies [25].
From a different perspective, also the bounded resources condition challenges existing results concerning the decision making process in typical RSs.
The process depicted by [42] and discussed in the previous section applies well in the context of unbounded resources, exemplified by scenario 1 and characterized by a very vast set of alternatives that remains large when screened and filtered according with user's preference criteria.
In this situation there are theoretical arguments as well as a large number of empirical studies - mostly in the e-commerce domain [33], [35] - that claim that typical RSs can provide effective support to users in all stages of the decision-making process. They facilitate both the initial screening of available alternatives and the in-depth comparison of item alternatives within the consideration set, reducing the total size of information processed by the users in the search set, consideration set and in-depth search set [42].
Hence we can posit that, under the unbounded resources condition, typical RSs reduce users' decision effort and users' decision time, hence improving the quality of the decision process.
In all cases depicted in scenarios 2, 3 and 4, the decision process is influenced by the "bounded" characteristics of the resources meeting users' needs and preferences, which may affect the validity of the above proposition and the effectiveness of traditional RSs for decision making purposes.
It is well known that, in any context, the RS attempt of reducing the user decision effort risks to create the so called filter bubble effect.
This term, first coined by Eli Pariser in [32] describes a phenomenon in which RSs tend to show only information which agrees with users' past viewpoints, effectively isolating the user in a bubble that tends to exclude items that may be helpful for the users' goals, i.e., novel and serendipitous items.

We cannot exclude that potentially negative effects of the bubble phenomenon get amplified in the context of bounded resources: the bubble can result so narrow that, as pinpointed by the discussion in the previous section, the intersection between the bubble and the set of available items is empty. If this is the case, the decision process must be iterated, possibly several times. This situation is likely to increase users' decision effort and users' decision time, and therefore decrease the quality of the decision process. This in turn have potentially negative effects on the users' perception of on her trust in, usefulness of, and satisfaction with the RS. Even worse, the user may give up before completing the decision process, leaving the current on-line reservation service and trying a different one, with obvious implications for the service provider, in terms of customers' trust and actual business outcomes. In order to overcome these problems, users must be exposed to novel and serendipitous recommendations [31]. This is a paradigmatic shift for the role of RSs in the decision process: from a tool that helps users in narrowing the search set and consideration set in the case of unbounded resources, to a tool that expands the in-depth set in the case of bounded resources.
Defining the design strategies of RSs that take into account the possibility of bounded resources are a challenging issue. Some requirements that need to be taking into account are the following:

- Support to decision making processes that are strongly iterative, maximizing the usability of doing and re-doing previous steps, particularly in the re-definition of preferences as the user becomes aware of the lack of available items matching her requirements.

- Need to maintain users' trust [34] and keep the user engaged with the decision process, in spite of the initial failures that potentially can occur because of the lack of resources. In this respect, specific explanation strategies [47] and appropriate conversational interfaces [45] should be defined, which not only improve transparency and explain how recommendations are generated, but also make the user aware of the shortage of resources

- Ability to act both as filter that limits the set of valuable alternatives and as multiplier that helps the user expand his/her horizons by recommending serendipitous alternatives.

Finally, the concepts of user's goals (sharp or soft) and bounded resources both have implications on evaluation models, methodologies and empirical studies regarding RSs as decision support tools.
Another issue is the absence of results which can address how the perceived

usefulness and satisfaction of a RS influence the behavior of the user.
While Xiao and Benbasat pinpoint that there can be a relationship between the decision making process and the evaluation (i.e. at a reduction of effort and an increase in decision quality correspond a better user's evaluation of RSs), no assumption is stated about how a positive or negative influence of user's evaluation of a RS can affect his/her behavior.

3.2 Research questions

From this set of challenges, we defined three main research questions which are referred to three areas concerning the interaction between RSs and users and have not been completely investigated yet:

i **How do RSs influence users' decision making process when users have different tasks?**

ii **Do the limited availability of on-line items and the corresponding time-pressure on the user influence users' evaluation of RSs?**

iii **Do users' evaluation of RSs influence the decision making process?**

The first research question tries to address the set of problems connected with the relationship between the task and the influence of RSs on the behavior of the user.
Xiao and Benbasat pinpoint that the complexity of the product can influence the degree of which a recommender can be adopted: a user who must search a complex and experience product needs more cognitive efforts to search, to analyze and to compare such product with another and therefore the RS support is more accepted [1]. To complete this analysis we also propose a different motivation in RS use (and therefore an impact on the decision making process) depending on users' tasks.
We believe that the characteristics and the use of a RS are not only in strict correlation with the recommended items (as already demonstrated in other researches) but also the task plays an important role in the process which leads the user to a decision.
An example of this can be done considering the scenario 1 and 4. It is possible to believe that in these two different scenarios users may perceive the

[1] The assumption under this statement is that the hotel is considered an experience product

support of a RS in very different way, and this is true also for its perceived usefulness and its overall influence on the user and on the outcome in general.
In the business scenario a user may perceive more useful a recommendation list with features very similar with the hotel's features he/she prefers. This can be inferred because the task has a big stress on constraints.
In the second situation instead, the user may find interesting also recommendations which doesn't reflect exactly the characteristics of the actual preference; this is possible because the user may be interested in novelty proposals or in finding even a better choice.
With these two easy examples we motivate the need for study the dependence between user decision, RS characteristics and adoption and the task or goal which a user is subjected to.
The second research question pinpoints an aspect which should be taken into account when treating the hotel reservation or a complex item in general: the dynamic feature of items involved in the decision process.
The main studies till now consider the items as static elements during the user-recommender interaction.
In our perspective we want to investigate the decision making process with respect to the dynamic properties of the items: instead of considering the linear *input-process-output* routine of decision making, we consider variable the items in the stages of the decision process. In other terms we investigate how RSs and users interact when the alternatives vary depending on dynamic factors. Some examples of these factors can be the price of a room over the period of booking and the availability of a room over the user session, i.e. after some minutes the room maybe is booked and therefore not available.
The last research question is introduced as there is not evidence which can proof that users' evaluation of a RS can influence also the decision making process.
Xiao and Benbasat in their conceptual model make the hypothesis, not empirically proved, that there should exist a connection between the decision process, in terms of effort, time and quality of the decision and the evaluation of the RS.
While this can appear as an evident relationship, a more accurate study may highlight how the direction of the relationship between user behavior respect to the evaluation of the system should be bidirectional.
As suggested by Pu and Chen [35], the evaluation of a RS can influence its support and adoption during the decision process, affecting also the final evaluation of the entire choice process (its quality, effort and time spent in decision).

Existing conceptual models for evaluation [46] do not provide explicit constructs for users' goals. Previous studies on decision making [32] pinpoint how the nature of users' tasks are important factor affecting individual's behavior and performance. Still, a task as defined in previous studies - "the set of functions that a working person, unit, organization is expected to fulfill or accomplish" [32] - has mainly a functional flavor.
Our study emphasizes the need for extending this functional perspective and raising the level of abstraction of the task concept, to address "goals", i.e., broader users' needs.
In addition, the discussion presented in the previous sections suggests extensions of existing frameworks for RS evaluation with explicit constructs that address the temporal and dynamic characteristics of RS resources.
All these extensions can lead to a more powerful conceptual model that can help contextualize a wider spectrum of empirical studies in a wide range of RS application domains and situations of use.

3.3 Design of the extended conceptual model

As suggested by its authors, the complexity of the Xiao and Benbasat conceptual model makes it infeasible to validate the model as a whole, hence small groups of features should be tested at a time. In addition, among the directions for future research, Xiao and Benbasat pinpoint the need for further developing their conceptual model.
We have followed both suggestions. We have extended the model along three directions:

i we have inserted a new proposition (dotted green line in the figure 3.2) for the user evaluation of the decision making process;

ii we have added a new feature to the item construct, the item availability;

iii we have inserted a new construct (the user task);

In the following we describe three possible extension of the model presented by Xiao and Benbasat.

3.3.1 Extension 1: user evaluation and the decision making process

In the Xiao and Benbasat model, the users' evaluation of RSs is addressed in general terms, i.e., is regarded as the users' perception of success of the

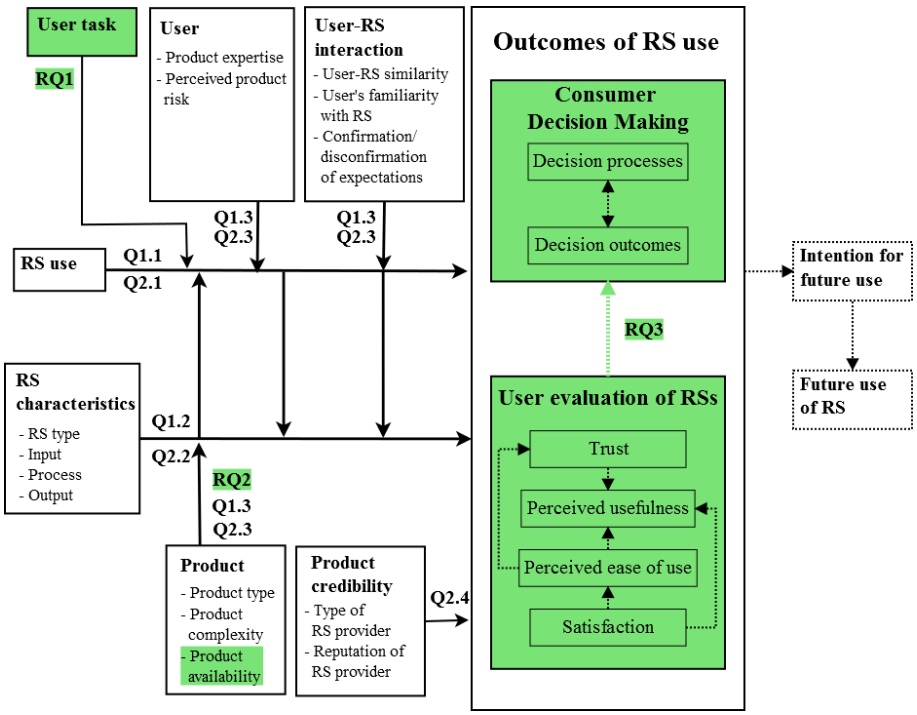

Figure 3.2: Extended conceptual model

RS, in line with acceptance theories (TAM). Thus they are only investigated in relationship to RS characteristics.
In contrast, we are interested in examining users' evaluation attributes as moderating factors of perceived quality of the outcome of RS use, i.e., perceived decision quality. In other words, we want to investigate the following proposition:
P: users' evaluation of RS influences perceived quality of the decision process.
Our hypothesis is that the perceived quality of the decision process is moderated by the perceived quality of decisions outcome (satisfaction of the outcome) and the perceived effort of the decision process (perceived decision time).

3.3.2 Extension 2: Task modeling

Among the suggestions for extending their model, Xiao and Benbasat pinpoint, among other aspects, that an additional construct worth further investigation is task, identified by Eierman et al. [24] as an important factor affecting user behavior and performance.
Task is defined "as the set of functions that a working person, unit, organization is expected to fulfill or accomplish. It is the job that is to be done using the decision support system".
The construct is not included in the original conceptual model since the "task" is fixed in Xiao and Benbasat's paper as the shopping task, i.e., purchasing a product with/without an RS individually. However, this construct becomes relevant when investigating RSs for users' tasks of different nature, complexity, or structures.
Tasks can differ for:

- the "action": buying a product, reading a news, booking a hotel, etc.;
- the "object": a book, a hotel room, a camera, etc.;
- the "goal": booking a hotel room for summer, family holidays vs. business travel, buying a camera for hobby vs. professional activity.

The first novelty of our approach is to extend the model with the "task" construct. The second novelty is to investigate the influence of RS use on decision making in two different tasks.
These tasks are characterized:

Business (strict goal): book a hotel room for specific location and date and in a specific price and category range (e.g., book a hotel in Rome,

single room, check in 14 of July, check out 16 of July, room must cost no more than 150€ per night and hotel category must be 3 or 4 stars).

Holiday (soft goal): book a hotel room for a generic location, with flexible dates (e.g., book a double room for two nights in a hotel on the Lake district in Italy, preferably in the central week of August).

Hence our study explores the differences in decision making (i) between users who are assisted by RSs and those who are not, as well as (ii) between users having different types of tasks. These considerations are represented in the following proposition we want to investigate:
P: Different tasks influence in different ways the user decision making process when using the RS.

3.3.3 Extension 3: bounded resources

An aspect that, to our knowledge, has been seldom considered in previous studies is that of RSs used in scenarios were items availability is limited and decrease with time. This is typically the case when booking a hotel room or a seat on a flight.
Specifically, we aim at studying and comparing the decision making process of RSs users under two conditions: the traditional conditions that characterize the *"preferential choice problem"*, where users need to take decisions across a vast set of alternatives (i.e., they need to identify the preferred idem within a large offer of products and services), and under the "bounded resources" conditions, when the offer of products and services among which a user can choice is very large, but the items that effectively match her initial preferences is very small (or even empty) so that he/she needs to reshape his/her preferences.
At the iterative nature of the process corresponds a variable nature of the sets identified in the stages of the decision process: some items in the search set may become unavailable at some moments of the process. Therefore the user must reconsider and maybe relax or revise some features between the preference in order to make a choice.
This situation leads to further extensions of the Xiao and Benbasat model (which assumes the existence of no constraints on product availability):

- the introduction of new a new variable on item availability (related to item characteristics)

- the introduction of a new proposition:

P: under bounded resources conditions, RS use influences users' perceived quality of the decision process and decision outcomes effort at a higher degree than under unlimited conditions.

3.3.4 Our approach to the extended model

The extended model presented above enables the study of all constructs already existent in the Xiao and Benbasat previous work. In addition we add three new postulates which we want to verify and which summarize our extensions:

PN1 Users' evaluation of RS influences perceived quality of the decision process

PN2 Different tasks influence in different ways the user decision making process when using the RS

PN3 Under bounded resources conditions, RS use influences users' perceived quality of the decision process and decision outcomes effort at a higher degree then under unlimited conditions

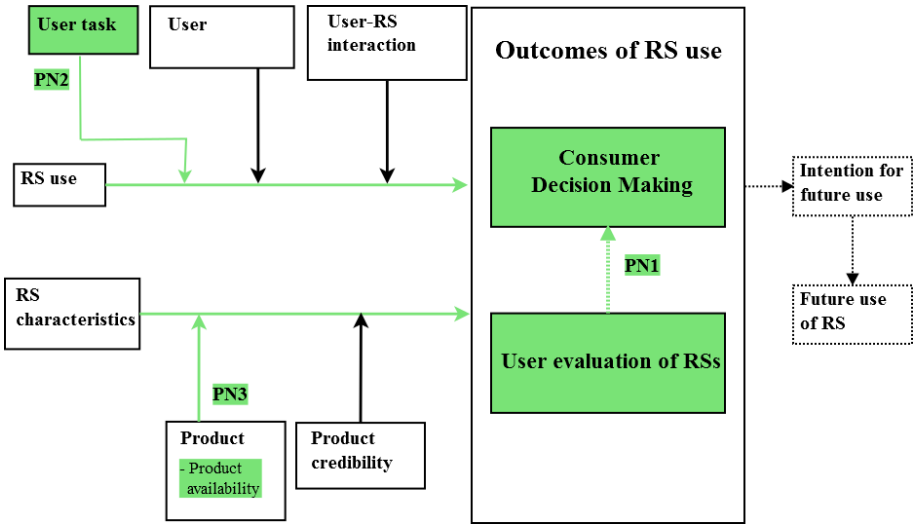

Figure 3.3: The correlation between constructs and new postulates

As our conceptual model is supported by a software framework, which let us study the dimensions of the model and set up empirical tests, at the moment we have the possibility to make experiments on a part of the whole model, while the remaining part will be considered in future works. In the two

tables below are showed all the possible dimensions enabled at the moment by the framework, while others are marked as future work. In the first table are presented all propositions concerning the decision making while in the second there are all postulates related to user's evaluation of RS.

Propositions	Construct	Status
P1, P2	RS use	enabled
P3	RS type	enabled
P4	RS input characteristics	future work
P5	RS input characteristics	variable not considered
P6, P7	RS output characteristics	future work
P8, P9, P10	Product	variable not considered
P11,P12	User related factors	enabled
P13	User-RS Interaction	enabled

Table 3.1: Propositions enabled by the framework concerning the decision making

Propositions	Construct	Status
P14	RS type	enabled
P15, P16, P17	RS input characteristics	future work
P18, P19	RS process	variable not considered
P20, P21	RS output characteristics	future work
P22	Product	variable not considered
P23, P24	User related factors	enabled
P25, P26, P27	User-RS Interaction	future work
P28	Provider credibility	future work

Table 3.2: Propositions enabled by the framework concerning the user's evaluation of RS

3.4 Metrics of evaluation

As described by the two authors, these constructs can be defined by a set of properties which can be measured by objective and subjective metrics, used to evaluate every single block of the model.
These metrics refer to three areas of analysis: *consumer decision making*, *users' evaluation of RSs*, *user-RS interaction*, and *user related factors*.

3.4.1 Consumer decision making

Consumer decision making construct is divided into decision processes and decision outcomes. These two sub-areas are studied with two metrics: the *decision effort metrics* and the *decision quality metrics.*

Decision effort metrics

Consumer decision effort refers to the amount of effort exerted by the consumer in processing information, evaluating alternatives, and arriving at an item choice.
In our work we have used three decision effort metrics: *decision time*, *the extent of item search*, and *the amount of user inputs.*
Decision time refers to the time consumers spend searching for item information and making purchase decisions. Since RSs assume the tedious and processing intensive job of screening and sorting items based on consumers' expressed preferences, consumers can reduce their information search and focus on alternatives that best match their preferences, resulting in decreased decision time.
The extent of item search refers to the number of alternatives that have been searched, for which detailed information is acquired, and have seriously been considered for purchase by consumers.
Considering the decision making process model introduced by Xiao and Benbasat[2], a good indicator for the extent of item search is the size of the search, consideration and in-depth sets.
Since RSs present lists of recommendations ordered by predicted attractiveness to consumers, compared to consumers who shop without RSs, those who use RSs are expected to search through and acquire detailed information on fewer alternatives (i.e., only those close to the top of the ordered list), resulting in a smaller search set, consideration set, and in-depth search set. Thus, the use of RSs is expected to reduce the extent of consumers' item search by reducing the total size of alternative sets as well as the size of the search set, in-depth search set, and consideration set.
However, in the desire of reducing the user decision effort, there is the risk to create a filter bubble effect.
This term, first coined by Eli Pariser in [32] describes a phenomenon in which RSs tend to show only information which agrees with the user's past viewpoint, effectively isolating the user in a bubble that tends to exclude novel and serendipitous information.

[2]It is described in the chapter 2

The amount of user inputs is the quantity of preference information provided by the user prior to receiving recommendations. Such metric can be retrieved by the interaction between the user and the system: for example by using filtering or menu ordering, the user can express information about his/her preference. In this point of view, a user who makes little interactions can perceive a better usefulness of RS and experience a better decision making process because of the minor effort and time spent to express features of interest.

Decision quality metrics

Decision quality refers to the objective or subjective quality of a consumer's purchase decision. It is measured in various ways [33], [40]:

(a) whether an item chosen by a consumer is an optimal or sub optimal decision alternative;

(b) as a calculated preference matching score of the selected alternatives, which measures the degree to which the final choice of the consumer matches the preferences expressed by the consumer;

(c) by product switching: after making a purchase decision, when given an opportunity to do so, if a customer wants to change the initial selection for another;

(d) the consumer's confidence in the purchase decisions.

Even in the case of RS use decreasing the number of users who purchase optimal items (because of the reduced search sets), the perceived decision quality may improve because of the reduced decision effort.
In this study we analyze the impact of RSs on decision quality measured by all the four dimensions aforementioned.

3.4.2 Users' evaluation of RSs metrics

As described by Xiao and Benbasat, users' evaluation of RSs in relation to users', RS's and item's characteristics can be divided in four main constructs: *trust*, *perceived usefulness*, *perceived ease of use* and *user's satisfaction.* In order to evaluate how RSs influence users' evaluation we must analyze all these variables to obtain an estimation of the overall user's evaluation of RS. As all of these metrics are subjective, we try to investigate this part of the conceptual model with a survey. The guidelines we refer to in order to state the questions are showed in the Pu Chen framework, namely ResQue,

in which are summarized previous works on RSs' evaluation [35].
Perceived ease of use and usefulness are defined as part of user beliefs of the system qualities [35].
In our work perceived ease of use, also known as efficiency and perceived cognitive effort, measures users' ability to quickly and correctly accomplish tasks with ease and without frustration. It also describes the extent to which RSs help users to quickly find their preferential items and therefore to accomplish their assigned task.
While task completion time can be measured objectively[3], it is harder to distinguish between the actual task completion time and the measured time. In fact this can depend on various factors; an example is when users explore the website to discover information unrelated to the assigned task. Such metric is evaluated through a set of questions designed to discover the *ease of decision making*. Such questions aims to discover the following guidelines:

- using the recommender to find what I like is easy;
- I was able to take advantage of the recommender very quickly;
- I quickly became productive with the recommender;
- finding an item to buy with the help of the recommender is easy;
- finding an item to buy, even with the help of the recommender, consumes too much time.

Another dimension of users' evaluation of RSs analysis in Xiao and Benbasat model is the perceived usefulness, which is the extent to which a user finds that using a recommender system would improve his/her performance, compared with experiences without the help of a recommender.
This dimension help us to understand users' opinion as to whether or not this system is useful to them.
Users must manage a an overwhelming flood of information and make high-quality decisions under limited time and knowledge constraints. Therefore the two main aspects of the perceived usefulness considered in the evaluation are the decision support and the decision quality.
Decision support measures the extent to which users feel assisted by the recommended system; the decision quality can be assessed by confidence criterion, which is the level of a user's certainty in believing that he/she has made a correct choice with the assistance of a recommender.
Here we present the main guideline to discover RS perceived usefulness suggested by Pu et al. [35]:

[3]Time is a decision effort metric

- the recommended items effectively helped me find the ideal product.
- the recommended items influence my selection of products.
- I feel supported to find what I like with the help of the recommender.
- I feel supported in selecting the items to buy with the help of the recommender.

The last metric to define user evaluation of RSs is the satisfaction. This attribute refers to users general attitude towards recommenders and is connected with their experience with RSs as they interact with such systems. Evaluating overall satisfaction determines what users think and feel while using a recommender system. It gives users an opportunity to express their preferences and opinions about a system in a direct way.
The satisfaction can be furthermore divided into confidence, which refers to the recommender's ability to inspire confidence in users, or its ability to convince users of the information or products recommended to them, and trust, which indicates whether or not users find the whole system trustworthy.
Such variables can be investigated following these guidelines:

- Overall, I am satisfied with the recommender.
- I am convinced of the products recommended to me.
- I am confident I will like the items recommended to me.
- The recommender made me more confident about my selection/decision.
- The recommended items made me confused about my choice (reverse scale).
- The recommender can be trusted.

3.4.3 User-RS interaction factors metrics

The User-RS interaction construct study the influence of factors related to the interaction of RS and the user during the experience and is focused on three main variables:

- user-RS similarity
- user's familiarity with RSs
- confirmation/disconfirmation of expectations

The first metric measures the similarity between user and RS in terms of past opinion agreement, goals, decision strategies, and attribute weighting. The second, measures the user's familiarity with the workings of RSs through repeated use. The last metric, measures the consistency/inconsistency between the user's pretrial expectations about the RS and the actual performance of the RS.

3.4.4 User's related factors metrics

User's related factors indicates two main characteristics which influence the decision making processes referred to users: the degree of user expertise about the items and the perceived risks about the items.

- product expertise
- perceived product risk

The first metric, product expertise factor, measures the user's knowledge about the intended item. The second, i.e. perceived product risks, measures the user's perception of uncertainty and potentially adverse consequences of buying an item. A consideration must be stated concerning our study. While in most of the researches, the use of a RS is in opposition to a normal system, in our testing environment we enable the coexistence of the RS support functionality with the normal use of the system. Hence we have the possibility to test the system with no RS support and the system in which the user may request the help of the recommender, enhancing the complexity of the analysis.
In fact, in our user's evaluation of RS as well as in the influence of RSs on decision making process, we have to consider also the integration between the RS support component and the normal use of the web application.
Therefore in the analysis of our research we trace the behavior generated by the recommender and the behavior which is not, as well as for the survey, where we don't ask direct questions about the recommender and we afterwards infer the validity of the postulates.

Chapter 4

The framework PoliVenus

In this chapter is introduced the architecture of the developed framework, namely PoliVenus. This framework, implemented with the JEE technology, has different modules which provide various functionalities: dataset building and data retrieval, content management and processing, and a testing application.
In the following sections we provide an overall description of the system and of the main technology applied (JEE). Finally it is discussed the structure of every module implemented in the system.

4.1 Overall description

According with the conceptual model described in the previous sections, we have developed a modular framework based on JEE technology that enables user experiments on hotel RSs. Figure 4.1 shows the system architecture. The application - namely PoliVenus - is implemented according with the MVC (model-view-controller) pattern and deployed on the open source application server (AS) JBoss. The application makes use of the Java search library Apache SOLR which manages the information related to hotels. User activity (e.g., user clicks) is tracked client-side by means of JavaScript and Ajax.
The application is connected to a MySQL database which stores both the hotel dataset (hotel name, location, room availability, user reviews/comments, ...) and the test settings (e.g., user tasks) and tracking (e.g., user browsing activity).
The management of the datasource is performed by two specific modules that retrieve all the information about hotels and manage its content to cre-

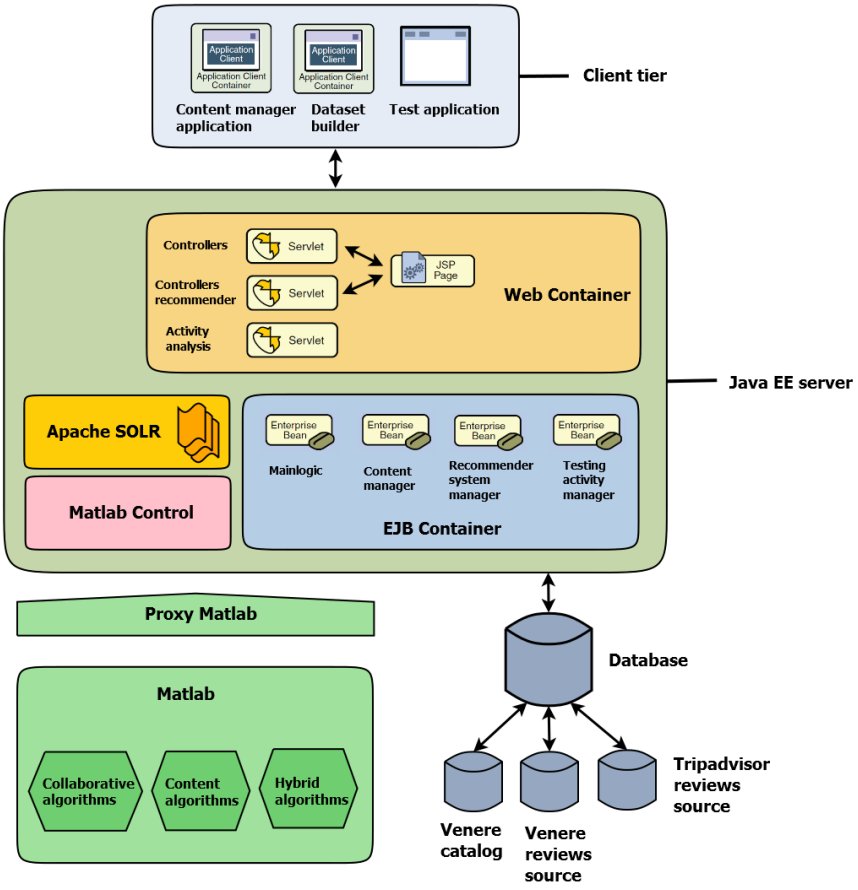

Figure 4.1: The PoliVenus framework

ate an intermediate layer of data which can be accessed by the numerical computing environment Matlab through MatlabControl, a Java API that acts as a proxy between Java and Matlab.
A multitude of recommendation algorithms have been implemented in Matlab and are available within the framework, spanning from collaborative and content-based filtering to hybrid solutions. Such algorithms provide different personalized lists of items: (i) recommendations of hotels based on the user activity, (ii) hotels similar to the one currently inspected by the user, and (iii) alternatives to the hotel selected by the user but not available in the specified dates.

4.2 JEE: the structure of the framework

The core of the system's functionality is the Java Platform, Enterprise Edition, which is designed to support inherently complex applications which can potentially access data from a variety of sources and distributing applications to a variety of clients.
The objectives of this technology are several. The first objective is to define an architectural model to build enterprise applications that are distributed, component-based, and transaction-oriented. Next aim is to provide a powerful set of APIs while reducing development time, reducing application complexity, and improving application performance. The main benefits of these APIs are:

- reduce development time
- reduce application complexity
- improve performance
- allow the application to access to various data sources
- offer the application functionality to various kinds of clients

Another important advantage in using this technology is the possibility to abstract from low level problems such as:

- transaction managament
- state management
- multi-threading
- connection pool management

JEE follows the phylosophy of "Write Once, Run Anywhere", which let its application to run on different machines and support interoperability with different systems.
The four elements on which is focused this technology are:

- an architectural model
- a logic model
- a JEE component definition
- a container

Architectural model The JEE technology is structured with a *three tier architectural model*, as explained in figure, which partitions the work needed to implement a multitier service into two parts: the business and presentation logic to be implemented by the developer, and the standard system services provided by the Java EE platform. The business logic, which is encapsulated in the middle tier, offers the access to all implemented services and is connected with external applications.

Logic model The logic model on which is based the JEE platform is the *MVC paradigm*, namely model-view-controller, which separates the tasks that each component must accomplish:

- *the model*, that provides methods to access the data for applications
- *the view*, that visualizes the data of the model and provides an interaction with users of the application
- *the controller*, which receives input from users (generally through the view) and modifies the other two components, i.e. the model and the view, to execute the command

JEE component has three characteristics:

- encapsulates the reusable implementation of some functionality
- can be composed without modifications
- respects the constraints imposed by a component model

A component model represents a strategy to organize components to build an application and how these objects must communicate.
A JEE component is defined as a self-contained software unit which is available to:

- develop clients (for example client applications or servlet, jsp etc. . .)
- develop the application logic and the data interaction mechanism (for example the EJB, enterprise java bean)

The EJB component can be of three types: an *entity bean*, a *session bean*, and a *message driven bean*.
An entity bean is a component that offers an object-oriented view on data stored in a database and it can have a long life (as long as the one of data in the database); a session bean component serves a single client and has a relatively short life. It can also be transaction aware and access the persistent data of a database. The message driven bean is similar to a session bean but it is linked to subscribed events (e.g. messages) and it is not used in our work.
An entity bean is used to map relational data to objects. For example tables or relationships of an ER model of a database, can be represented as objects linked by relationships which define the same database. This mapping operation, enabled by entity beans, associates an entity of the database to a *POJO*, i.e. a plain old java object, which is the representation of the entity as a java class with some special annotations (typical of JEE platform).
Those java classes have the getters and setters methods to modify the properties of an object which is managed by the container through an Entity Manager object which attach or detach the entity to or from the database.
A session bean is a JEE object which contains the business logic. It is of two different types: a *stateful session bean (SFSB)*, which is a conversational service between a single client and the specific bean with no concurrency, and a *a stateless session bean (SLSB)*, which offers services associated to an operation where the bean does not keep track between two calls.
The main elements which constitute a session bean are the interface. An interface of a session bean provides methods to operate on the data. This methods can be called through the java naming directory interface[1] (JNDI) for local or remote calls or injection for local calls. The interfaces in fact can be local or remote. A local interface is used to make a communication between components of the same container, while the remote can be used from applications which are outside the scope of the container component.
The methods exposed by such interfaces are implemented into the session bean itself.

The container It is the EJB component which is responsible for the access to application resources from a high number of clients. It manages transac-

[1] It is an interface to publish and discover objects

tions, communications, concurrences etc... issues, raising the developer from taking into account such problems and develop at a higher abstraction level. The core of our framework is built on JEE technology and has in the PoliVenus EJB its logic core.
The whole framework is modular, in the sense that all of its components can vary their configuration or can even be included or excluded (turned on or off). The PoliVenus EJB can be divided into three main services[2]. We can group such services in the following functionalities:

- **dataset builder**

- **content manager**

- **test system**

The services are provided with different session beans, grouped as follows: a first group is responsible of the management of all static data of the hotels, for example rooms, reviews, policies, ratings related etc...; such group is accessed by the testing application and the dataset builder.
A second group of session beans manages the content of the data related to hotels; this group is responsible of data processing used for recommendation and it is accessed by the content management service.
A third group of session beans manages the access to the Matlab proxy and the reordering of recommendations results; these session beans are accessed by the testing application only.
Finally a fourth group of session beans organizes the testing variables to be stored (e.g. user activity or web application session variables) during the testing activity with users and it is also accessed by the testing application.
The database is mapped with a set of entity beans which represents the three partitions of our data.
The first partition is composed by the static data of hotels and all related entities.
The second partition is composed by the data which are the result of the processing of the content management.
The last partition is composed by data of the tests performed by the web application, with all tested users information. In the following sections we describe the three modules of the PoliVenus EJB and the correspondent applications.

[2]In the specific situation we do not mean web services but a set of functionalities applied into three applications

4.3 The dataset builder

The dataset builder application is a java-php standalone client application which is running on our server in a batch processing mode. Its main tasks are the population of the database with hotels and related properties and the extraction of reviews and ratings from different sources. This is obtained through a set of API, described in the next paragraphs, a php script, and a module of the EJB responsible for populating the dataset of "static" items (e.g. hotels, rooms etc. . .)
The dataset builder module provides a set of functionalities to manage the whole database. One part of these functionalities refers to the population of hotels and all entities such as room list, stay policies, hotel features and the rest of properties related to hotels from the catalog of Venere.com (our partner); another part refers to the management and the extraction of reviews and ratings from two different information sources: the Venere.com source and the tripadvisor website[3].
While the population of the hotels and properties are executed in a single run, starting from a catalog, the extraction of reviews and ratings is a more complex activity performed in a two phase processing: in the first phase ratings and reviews are extracted into an intermediate format in a batch execution mode; in the second all these intermediate data are handled and inserted into our database.

4.3.1 Hotels extraction

The part of hotel catalog management concerns with the population of the database with all hotels and related characteristics and sub entities. The catalog provided by our partner is in xml format and has the following elements per hotel:

id: the identifier of the hotel

name: the name of the hotel

status: if the hotel is available for booking

type: the type of structure, i.e. hotel or bed&breakfast. . .

roomsNumber: the number of rooms for the hotel

rating: the category of the hotel, e.g. 4 stars

[3] www.tripadvisor.it

venereRanking: the top viewed hotels ranking for Venere

userRating: the average rating of users about the hotel

cancellationPolicy: the cancellation policy which users who book the hotel are subjected to

amenityList: the list of features of the hotel, e.g. air conditioning etc ...

roomList: the list of rooms with all features and descriptions, e.g. the restaurant description, the room description, the price etc...

paymentDetails: is the payment constraints which people are subjected to, e.g. taxes included or available credit cards

doublePrices: the price of double rooms expressed in Euro

location: the geographic information of the hotel, including latitude and longitude coordinates

hotelDescription: all textual descriptions of the hotel such restaurant, overview etc...

hotelUrls: urls of some characteristics of the hotel such as the thumbnail etc...

hotelContacts: the contacts of the hotel: telephone and fax

imageList: the list of all images of the hotel

To parse this catalog XML, which is a file of 500Mb, we use the Streaming API for XML (StAX), a streaming Java-based, event-driven, pull-parsing API for reading and writing XML documents. Such API enables to create bidirectional a XML parser which is fast and has a light memory footprint. StAX was created to address limitations in the two most prevalent parsing APIs, SAX and DOM. The primary goal of the StAX API is to give parsing control by exposing a simple iterator based API. This allows us to ask for the next event (pull the event) and allows state to be stored in procedural fashion.
There are two programming models for working with XML infosets: streaming and the document object model (DOM).
The DOM model involves creating in-memory objects representing an entire document tree and the complete infoset state for an XML document. Once in memory, DOM trees can be navigated freely and parsed arbitrarily, and as such provide maximum flexibility.

However, the cost of this flexibility is a potentially large memory footprint and significant processor requirements, because the entire representation of the document must be held in memory as objects for the duration of the document processing.
This may not be an issue when working with small documents, but memory and processor requirements can escalate quickly with document size.
Streaming refers to a programming model in which XML infosets are transmitted and parsed serially at application runtime, often in real time, and often from dynamic sources whose contents are not precisely known beforehand.
Moreover, stream-based parsers can start generating output immediately, and infoset elements can be discarded and garbage collected immediately after they are used. While providing a smaller memory footprint, reduced processor requirements, and higher performance in certain situations, the primary trade-off with stream processing is that you can only see the infoset state at one location at a time in the document. You are essentially limited to the "cardboard tube" view of a document, the implication being that you need to know what processing you want to do before reading the XML document.
The other characteristic of StAX is that it is a streaming pull-parsing.
Streaming pull parsing refers to a programming model in which a client application calls methods on an XML parsing library when it needs to interact with an XML infoset; that is, the client only gets (pulls) XML data when it explicitly asks for it.
The main advantages of this API are summarized below:

- with pull parsing, the client controls the application thread, and can call methods on the parser when needed
- pull parsing libraries can be much smaller and the client code to interact with those libraries much simpler
- pull clients can read multiple documents at one time with a single thread
- StAX pull parser can filter XML documents such that elements unnecessary to the client can be ignored
- StAX parser limits the use of memory for the XML document

With this streaming pull parser we examine the XML from Venere catalog, which is a view of their database, "windowing" its content in order to examine one node each time. This because every node is rich of content and

with this parser it is possible to store a fragment of the XML to augment the performances.
After storing the nodes related to a hotel during multiple scan of the document, we complete our main object, which is an instance of the Hotel class, adding all its properties and saving the object into the database with the interface provided by the EJB.
This interface provides methods to persist the hotel object and all its sub-properties, i.e. all its sub-nodes of the XML represented also as classes, with relationships *one to one*, *one to many*, and *many to many*.
An example of these objects is provided in the source code listed below, while the interface which exposes methods to persist them into our database in in the appendix section.

Listing 4.1: Code of the Hotel class and the EJB remote interface to persist all objects

```
@SuppressWarnings(serial)
@Entity
@Table(name=hotel)
public class Hotel implements Serializable {

    private int id;
    private String name;
    private String type;
    private Integer rooms_number;
    private String rating;
    private Double user_rating_venere;
    private Double user_rating_tripadvisor;
    private String booking_methods;
    private String rooms_description;
    private String phone;
    private String fax;
    private Double venere_ranking;
    private CancellationPolicy cancellation_policy;
    private Prices prices;
    private HotelDescription hotel_description;
    private HotelURL hotel_url;
    private Venere_rating_properties_overall
        venere_rating_overall;
    private Tripadvisor_rating_properties_overall
        tripadvisor_rating_overall;
    private Location location;
    private PaymentDetails payment_details;
    private Set<Amenity> amenity_list;
    private Set<Code> codes;
    private Set<Image> image_list;
    private Set<Room> rooms;
    private Set<TripadvisorReview> tripadvisor_review;
    private Set<VenereReview> venere_review;
    private Set<ContentManagementFlat> content_management_flat;
    private Set<ContentManagementTFIDF>
        content_management_TFIDF;
    private Set<UserTestedBooking> user_tested_booking;
    . . .
    }
```

With this API and the module of the EJB PoliVenus, namely the main-logic module, which concerns the management of data about the hotel, we populate the partition of our database about the data of hotels and all its hierarchy: descriptions, photos, rooms, policies etc...
The client is a Java standalone application which runs once for several minutes and updates or inserts objects into the specific partition of the database.

The catalog is about 60,000 hotels between Italy and the rest of the world. For our work we store 6,000 hotels of the main cities of Italy, namely Rome, Milan, Bologna, Florence, Venice, Naples, Turin, Verona and all their provinces. We opt for a subset of hotels because we can provide for these hotels the reviews and ratings to apply the RS (especially the collaboratives algorithms).

4.3.2 Reviews and ratings extraction

To increase our dataset with social information, we extend the pure content of hotel properties with reviews and ratings from different sources.
The two main sources considered are the Venere dataset, concerning ratings and reviews of users who overnight, and the Tripadvisor website, which offers advices and collects data from users about their travels.
We keep this two sources distinct in our database; also the extraction procedure is performed with different techniques, depending on the available instruments. While the Venere dataset provides a set of web services to retrieve information about users, Tripadvisor does not expose its web services. Therefore in the first case we used a library specific for web services available for php language; in the second case we implement an HTML parser.
Another consideration to make about this service is that, unlike the hotels extraction and DB population, this is performed in two phases:

- *information extraction*

- *storage of data into db*

The first phase, the information extraction, is implemented based on the reference source. In the case of Venere, it is a php script, while for Tripadvisor it is realized with Java. Both of them however are working in a batch mode processing.
The second phase instead is implemented as a java client application which uses the interfaces of the EJB module of persistence to store data previously extracted into the database.
This process, as for the hotels extraction module aforementioned, is running standalone for some minutes periodically.
Both of the datasets of Venere and Tripadvisor are extracted into a common format, XML, from which they are inserted into the database during the second phase.
We argue now the extraction techniques adopted, first for Venere dataset and then for Tripadvisor website.

Information extraction from Venere.com

Venere.com provides us a set of web services to obtain data through their public interface XHI (XML Hotel Interface). The web services are implemented using SOAP messages. Their communication is secured in two ways: a communication on HTTPS (SSL) which grants the security of the communication socket or by an identification of the SOAP message. In this second case the message sent has a specific header in the envelope for authentication.
We implement the second option, with the authentication included into the header of the SOAP message. An example of response message is listed below.

Listing 4.2: Example of SOAP message response

```
<soap:Header>
  <TransactionData xmlns=http://www.venere.com/XHI>
    <TransactionID>TTID.XHI.f0wbwuox.1177417422279.1</
        TransactionID>
  </TransactionData>
</soap:Header>
<soap:Body>
  <XHI_FeedbackReadRS xmlns=http://www.venere.com/XHI
      msgTimeStamp=2007-04-24T14:23:42.599+02:00 success=true
      >
    <FeedbackItems size=3 defaultPropertyID=6894>
      <FeedbackItem itemID=1335 langID=it itemDate=2006-10-13
          T12:20:00 customerGlobalRating=4.45 customerTypeID=
          ct02 vacationTypeID=vt01 wouldComeBack=false
          customerName=Mario Rossi customerCountry=Italy
          customerCity=Rome/>
      <FeedbackItem itemID=1156 langID=en itemDate=2006-10-10
          T14:25:00 customerGlobalRating=6.122 customerTypeID
          =ct03 vacationTypeID=vt02 wouldComeBack=true
          customerName=John Doe customerCountry=United
          Kingdom customerCity=London/>
      <FeedbackItem itemID=1095 langID=en itemDate=2006-10-07
          T14:25:00 customerGlobalRating=7.43 customerTypeID=
          ct04 vacationTypeID=vt03 wouldComeBack=true
          customerName=E.G. customerCountry=France
          customerCity=Paris/>
    </FeedbackItems>
  </XHI_FeedbackReadRS>
</Body>
```

The request and the response messages are managed by a script written in php language. This script is running in batch mode and sends requests and

receives responses with the *nu-soap library*.
With such library it is possible to build a client for web services, including the authentication in the header of the soap message. This client executes a call to the web service through the WSDL descriptor file, which specifies:

- usable operations of the service
- the protocol of communication and the format of input and output messages, as well as the bindings of the services
- the *endpoint* of the service

The nu-soap client object exploits the WSDL descriptor file and establish the socket and the communication through the request message. More in detail it discovers the functionalities exposed by the web service and exploits such functions with the SOAP protocol. With the answers messages are provided information about users' feedback.
In the request we can specify the language of reviews we desire (for example french, italian reviews etc. . .), the range of dates of the reviews, if we want only ratings or, in addition, the text of reviews.
The structure of the answer is described in the following UML diagram: The data received from the response is then treated with a DOM document.

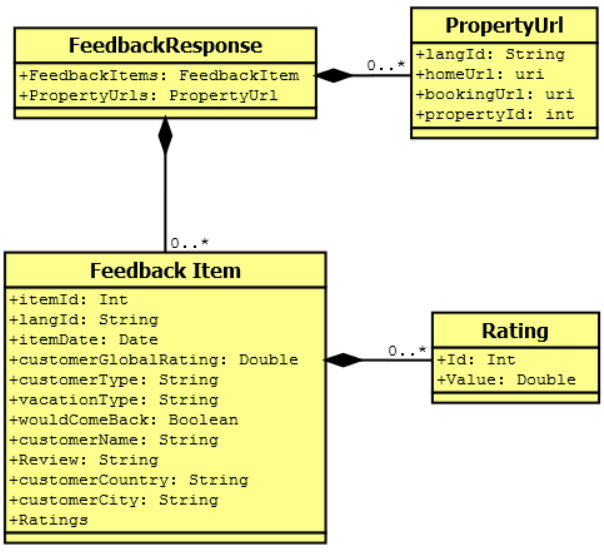

Figure 4.2: Web service response

This document enables us to manage the response as a tree which is stored

in memory, where every node is a tag with useful information.
The DOM document is then examined, read and saved into an XML file, containing the reviews properties of the referenced hotel. An example of a saved XML file containing the reviews of the response from the web service is listed below:

Listing 4.3: An XML file with the reviews and ratings for a hotel

```
<?xml version=1.0 encoding=UTF-8?>
<HOTEL_REVIEWS name=Montevecchio average_rating=7.6 id_hotel
    =135812>
  <REVIEW country=France username=9.9. travel_type=Piacere
      customer_type=Coppia giovane would_come_back=true date
      =2011-11-28T10:08:00 lang_id=fr id_review=996564>
    <RATING overall_rating=9.7667>
     <Pulizia_camere>10.0</Pulizia_camere>
     <Accoglienza>10.0</Accoglienza>
     <Silenziosita_camere>8.0</Silenziosita_camere>
     <Ampiezza_camere>9.0</Ampiezza_camere>
     <Spazi_comuni>9.0</Spazi_comuni>
     <Professionalita_addetti>10.0</Professionalita_addetti>
     <Qualita_servizio>10.0</Qualita_servizio>
     <Dintorni_hotel>9.0</Dintorni_hotel>
     <Prezzi_offerti>10.0</Prezzi_offerti>
     <Servizi_disponibili>10.0</Servizi_disponibili>
     <Fotografie>10.0</Fotografie>
     <Localizzazione_sulle_mappe>10.0</
         Localizzazione_sulle_mappe>
     <Tipologia_e_numero_camere>10.0</Tipologia_e_numero_camere
         >
     <Come_valuti_il_tuo_soggiorno>10.0</
         Come_valuti_il_tuo_soggiorno>
     <Rapporto_qualita_prezzo>10.0</Rapporto_qualita_prezzo>
    </RATING>
  </REVIEW>
  <REVIEW>. . .</REVIEW>
</HOTEL_REVIEWS>
```

As it is showed in the fragment above, it is possible to extract a set of information about the feedbacks concerning the hotels. In detail, the most peculiar characteristics are the country and the typology of the customer; moreover all the text of the reviews in addition to a set of ratings: an overall score plus a list of features concerning the rooms characteristics (e.g. the degree of noiseless or the quality of the service...). With this batch processing, at the moment we have in our database a set of 125,000 reviews and ratings of different hotel's properties concerning the Venere data source.

Information extraction from Tripadvisor.com

The second source of user information we consider in our work is the Tripadvisor website, as it provides users a lot of information concerning travels and hotels' evaluation of other users who already overnight there.
The website assists customers in gathering travel information, posting reviews and opinions of travel-related content and engaging in interactive travel forums. Tripadvisor was an early adopter of user-generated content. The website services are free to users, who provide most of the content, and the website is supported by an advertising business model[4].
Our idea to exploit this source of information is to extract and store all the free information about other users' experience in their travels. Due to the impossibility to use Tripadvisor API, the approach used to obtain this free information is to analyze the content of the website. More in detail we realized a *customized HTML parser*, with the help of some Java libraries, to analyze the structure of the page and extract from its node the content we need.
To realize this parser we evaluate different Java libraries which offers methods for HTML analysis. The library we finally choose is *JTidy*, which is a HTML syntax checker and pretty printer. It can be used as a tool for cleaning up malformed and faulty HTML. In addition, JTidy provides a DOM interface to the document that is being processed, which effectively enables the use of JTidy as a DOM parser for HTML.
The parsing procedure is composed by the following steps:

1. it is established a connection with the Tripadvisor website on the first review page of the selected hotel
2. the webpage retrieved by the URL is "sanitized" with the JTidy library
3. the page is stored in memory in a DOM document with the JTidy library
4. information is extracted with XPath queries on the nodes of the document
5. information is saved into an XML document

In the first phase, we redirect our search on the page of the Tripadvisor website referring to actual hotel[5]. It is then retrieved the URL of the first page

[4]The revenue is gathered by the advertise, i.e.the number of users conveyed by the advertise

[5]That is the hotel of which we are searching the reviews

of reviews and ratings of users:
http://www.tripadvisor.it/ShowUserReviews-g187791-d284236-r53341819
the URL is always composed by a first part which is always the same *http://www.tripadvisor.it/ShowUserReviews* while the second, *g187791-d284236-r53341819* which identifies the hotel, is obtained by specifying the name and the city of the searched hotel.
The first HTML page of the reviews of Tripadvisor is stored in a DOM object thanks to the extracted URL.
In the second phase, we use a special function of the JTidy library to improve the quality of the webpage: because HTML is often a not well structured DOM document (e.g. missing open or close tags, not unique identifiers etc...) due to some mistakes of web pages, we sanitize the retrieved page in order to store it in a well structured DOM format.
In the third phase we store the page in a DOM document; with this passage we pass from HTML to DOM format to analyze the nodes and the content of HTML with XPath expressions.
The fourth phase is characterized by the use of XPath, which is a query language for selecting nodes from an XML document. With some expression we convey our search on specific nodes of the document which contain the content we are searching for. An example of this content is expressed in the following code. An example of such XPath expressions:
//div[@id='REVIEWS']/div/@id
which indicates the parser to stop at the *div* element with id equal to *REVIEWS* and take the id property of the child of its child *div* (i.e. the XPath query will extract in this case the identifier of the review).
The last phase is the storage of the content in a well formed XML document, with the same structure as the document of Venere review. Below is the example for a review extracted from Tripadvisor: The phases are repeated cycling, where the next page of reviews is obtained by retrieving the URL in the actual page. When the last page of reviews for a hotel is reached, the cursor of the hotel scrolls for the next one.
The HTML parser implemented enables us to collect about 250,000 reviews and ratings of hotels to be used in our recommendations and OTA website reproduction.
A common mechanism that the two aforementioned batch clients share is the *resume research* feature: the batch process can be stopped by command line at every time. The software ends after completing the storage of the XML documents and saves informations about the research in a support file. When the batch process is started, it is asked if the user wants to start a new research or resume (if existent) an old one. If the search is resumed, in the

case of Venere, it is saved the last hotel analyzed of the list, so the restart considers the next hotel. In the case of Tripadvisor, it is saved the link of the last review page analyzed, so that the process can retrieve the link for the next page or next hotel (in the case it is the last page of reviews).
All the data extracted in XML files are organized in a hierarchical structure where at the top there is the city, defined as a folder; subsequently there is the set of XML files, each one for a hotel of the city, containing all the extracted reviews for that hotel. Here is a representation of this structure.

Once this intermediate layer is complete, we store the data into a portion

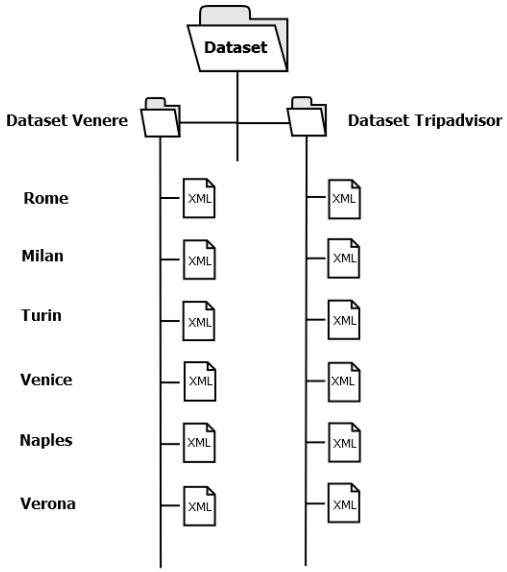

Figure 4.3: Structure of the extracted dataset

of database related to the information on hotels. As already mentioned the insertion into the database is a single shot operation which occurs thanks to the remote interface provided by the module of enterprise beans (i.e. the mainlogic module) which takes care of synchronizing the data into the XML structure with the database utilized in our tests.

4.4 Content manager

The content manager is a Java standalone application which is responsible of updating the information and the content of hotels stored into the database. This update is performed by the periodical requests of the client to an EJB

module that provides a set of methods to process the information of the hotels and to store it in the database.
This post processing of information is necessary since recommender content algorithms use a determined format of data to make the recommendations. Such content algorithms work on *matrix based* data representation: the main object they process is the *stem-item matrix*, which is composed by rows, formed of the stemming of content attributes of every hotel, and the column of the hotel's index itself.
The values expressed by the matrix correspond to a *weight* that such content stemmed has, as a representation of the feature of the hotel.
For example the word "silent" extracted from the description can express a particular feature of the hotel *Grand Hotel Flemming* in Rome. The word silent is the stem while the Grand Hotel Flemming is the item of the matrix considered.
The weight to be associated to the stem in relation to the item can be computed in different ways, depending on the origin of the content itself.
Therefore we define a content classification which divides the features and the description of the hotel: the *flat content*, composed by features represented as tags and the *free text* content. At this categorization corresponds a division in two tables of the content. In one table, *stem-item-flat matrix*, are inserted all features of the hotel representable as tags; the weight of these attributes is computed on a fixed scale of their reputed importance. In the other table, the stem-item-TFIDF matrix, the free text is pre-processed with a stemming algorithm and the weight is calculated with the TFIDF schema. An example of these two matrices is showed in the following tables.

Stem	Item	weight
booking methods.IB	13353	1
50-150	13337	5
Italia	12889	4
Lombardia	12771	4
Milano	12699	4
Palestro	11982	4
Vicino Milano	11806	4
tipology.C	11805	4

Table 4.1: Stem item flat matrix example

Stem	Item	weight
stazion	13353	0.018121222838544938
poi	13337	0.015036197476527931
time	12889	0.018926010849471597
riunion	12771	0.01910998189081384
swimming-pool	12699	0.01892215083171725
air-conditioning	11982	0.018988325096615973
famil	11806	0.01871866405755269

Table 4.2: Stem item tf-idf matrix example

4.4.1 Stem item flat matrix

Stem item flat matrix is the representation of features common to all hotels and a hotel. These features are:

- the category of the hotel (e.g. residence, hotel etc. . .)
- the rating of the hotel (e.g. the stars)
- the location, declined as country, region, city, city area
- the booking methods
- the price of the single room
- the amenities (e.g. cinema)
- the value add of the rooms (e.g. offered dinner)

For these features we attribute the following values The main purpose of

Feature	Weight
Category	4
Rating	5
Location	4
Booking methods	1
Price	5
Amenities	3
Value add	1

Table 4.3: Hotel features and related weight

this matrix is to directly compare the content of the hotels expressed as a set of features defined by the stems. From this matrix the algorithms

compare the similarities between items (e.g. hotels) in order to find the most similar one respect to the actual seen by the user. Therefore, because of the homogeneous structure of such features, it is possible to compare all items with this "tag representation".

4.4.2 Stem item TFIDF matrix

The assumption of the stem item flat matrix is that every hotel's content is defined by tags representing features uniformly comparable.
This assumption does not hold in the case of free text, because it is not possible to directly compare two texts at less than reducing both to elements which can be confronted. This is clearly the case of reviews extracted by the Venere and Tripadvisor sources, where the content must be processed in order to obtain a structure with a tag (or stem) describing a feature, an item (i.e. the hotel) and a weight (i.e. the value of the feature for the item).
Another aspect to take in account is that the text processing must consider the language of the reviews, since we extract both text in English and in Italian.
In order to overcome these problems we use a Java library, namely Lucene, and a search engine, SOLR, which we integrate in our Application Server AS JBoss. In the following paragraph we describe their functioning and features and how we use them to obtain the stem-item TFIDF matrix for recommender algorithms.

Apache Lucene and SOLR

The apache Lucene is a Java-based indexing and searching technology, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities.
Solr is a high performance search server, written in Java, and runs as a standalone full-text search server within a servlet container such as Tomcat. Solr uses the Lucene Java search library at its core for full-text indexing and search. It provides also functionalities for managing the content and address our issues.
Hence, in order to get solutions of the aforementioned problems, we must use the Lucene core API into SOLR search engine to address the content of our dataset, exploit a mechanism of language recognition and use a stemming algorithm in order to obtain, from the free text, a set of stems which characterize the content of the dataset. Finally, after obtaining for each hotel its stems deriving from reviews and descriptions, there must be computed the TFIDF schema to weight every stem.

We can therefore summarize the logic of the content manager module concerning the free text management with the following tasks:

- indexing the documents
- language detection
- language analysis
- compute the TFIDF schema

Indexing the documents To exploit the functionalities exposed by SOLR, we must at first address the documents into the search engine. This operation is an indexing which is performed with the lucene API.
The hierarchy of our information starts with the *INDEX*, which is used by SOLR, and therefore Lucene API, to search and perform other operations of data analysis. Such object is identified with the *IndexWriter* class of Lucene, which is used to create or use an existing index.
At the second level of the hierarchy there is the *DOCUMENT*, which represent the indexed object. It represents the information unit, in our case it is all the information we want to address and analyze concerning the hotel.
Every DOCUMENT has different *FIELD*, i.e. a couple key-value which indicates the information in our document. For our purpose, i.e. analyze free text with SOLR, the structure of the document must be the following: The

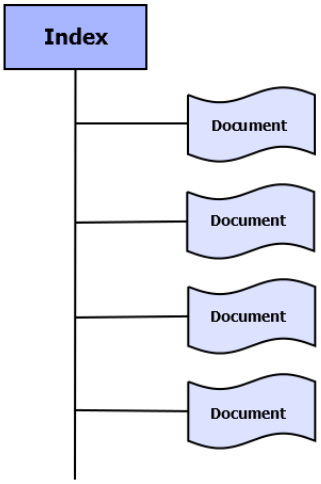

Figure 4.4: SOLR indexing

text fields we want to analyze to extract meaningful keywords are:

- rooms description
- hotel overview
- stay policy
- restaurant description
- service fees
- breakfast description
- how to reach
- Tripadvisor reviews
- Venere reviews

The other fields are inserted directly in the stem-item flat matrix with the relative weight, because they represent already homogeneous characteristics of the hotels which can be compared.

Language detection The second issue we want to address is the possibility to treat in different way different languages. This because the text of the reviews and descriptions in our database are in Italian and English languages. The techniques to extract the keys from the sentences are different because they are based on language rules. One of these techniques is the stemmer for Italian language and for English language, which needs to recognize the language to apply the right rules to obtain the stems.
A solution to this problem is provided by a feature of SOLR called *language detection*: it is possible in fact to dynamically recognize the language of the content of a field at indexing time.
This feature is provided by the *language detection processor*, which recognizes the text of a field and automatically generates a new field with the same name with an underscore followed by the identifier of the detected language. For example the *tripadvisor review* field can be easily transformed by the processor at indexing time into *tripadvisor review it* or *tripadvisor review en* field.

Listing 4.4: LangDetect library: the language detection processor

```
<updateRequestProcessorChain name=langid>
  <processor class=org.apache.solr.update.processor.
      LangDetectLanguageIdentifierUpdateProcessorFactory>
    <str name=langid.fl>hotel_overview,cancellation_policy,
       tripadvisor_review</str>
    <bool name=langid.map>true</bool>
    <bool name=langid.map.individual>true</bool>
    <str name=langid.fallback>it</str>
  </processor>
  <processor class=solr.LogUpdateProcessorFactory/>
  <processor class=solr.RunUpdateProcessorFactory/>
</updateRequestProcessorChain>
```

Hence it is necessary to specify two different text field typologies, one with policies of analysis for Italian and another one for English. This can be obtained defining two different analyzers.

Language analysis As described in the previous paragraph, the detection of text language leads to define two different strategies for text analysis to reduce the whole text to a set of stems. These strategies depend on the language.
The main technique in information retrieval to extract stems from the text is called *stemming*.
The stemming process is defined as the process for reducing inflected (or sometimes derived) words to their stem, base or root form-generally a written word form.
The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. For example the words “fishing”, “fished”, “fish”, and “fisher” to the root word, “fish”.
With this technique, all the sentences are reduced to a set of stems which refer to the hotel which contains the textual attributes. Hence the content algorithms can perform the recommendations from the comparison between the stems and their weights.
The most used stemming algorithm for English is the *Porter stemming*. A variant of this stemming for other languages is the *Snowball stemming* (Italian included). Therefore we adopt these two techniques in our work.
In order to make this analysis, SOLR offers the possibility to configure different parameters for a field type. As argued before, to define different analysis for Italian and English, we have to define two custom field types which are called “text content it” and “text content en” and for each of them specify a

policy of analysis:

Listing 4.5: The text analyzers

```
<fieldType name=text_content_it class=solr.TextField>
  <analyzer>
    <tokenizer class=solr.StandardTokenizerFactory/>
      <filter class=solr.ElisionFilterFactory ignoreCase=true
          articles=lang/contractions_it.txt/>
      <filter class=solr.LowerCaseFilterFactory/>
      <filter class=solr.StopFilterFactory ignoreCase=true
          words=lang/stopwords_it.txt format=snowball
          enablePositionIncrements=true/>
        <filter class=solr.ItalianLightStemFilterFactory/>
        <filter class=solr.SnowballPorterFilterFactory language
            =Italian/>
   </analyzer>
</fieldType>
  <!-- field type inglese -->
<fieldType name=text_content_en class=solr.TextField>
  <analyzer type=index>
    <tokenizer class=solr.StandardTokenizerFactory/>
       <!-- Case insensitive stop word removal.
          add enablePositionIncrements=true in both the index
              and query
          analyzers to leave a gap for more accurate phrase
              queries. -->
        <filter class=solr.StopFilterFactory ignoreCase=true
            words=lang/stopwords_en.txt
            enablePositionIncrements=true/>
        <filter class=solr.LowerCaseFilterFactory/>
        <filter class=solr.EnglishPossessiveFilterFactory/>
        <filter class=solr.KeywordMarkerFilterFactory protected
            =lang/protwords.txt/>
        <filter class=solr.PorterStemFilterFactory/>
  </analyzer>
  <analyzer type=query>
    <tokenizer class=solr.StandardTokenizerFactory/>
    <filter class=solr.SynonymFilterFactory synonyms=lang/
        synonyms.txt ignoreCase=true expand=true/>
    <filter class=solr.StopFilterFactory ignoreCase=true words=
        lang/stopwords_en.txt enablePositionIncrements=true/>
    <filter class=solr.LowerCaseFilterFactory/>
    <filter class=solr.EnglishPossessiveFilterFactory/>
    <filter class=solr.KeywordMarkerFilterFactory protected=
        lang/protwords.txt/>
    <filter class=solr.PorterStemFilterFactory/>
  </analyzer>
</fieldType>
```

from the code above, it is possible to understand that for the two languages are implemented two different analyzer: one Porter algorithm based and another Snowball based. The analyzers, beyond the policies defined by the two algorithms, make use of different text files called stopwords, contractions and protwords.
The first, stopword, is a file containing all the standard words which are not meaningful, such as "a", "the" etc... both for English and for Italian.
The contractions file is used for Italian language to indicate a set of words which have a contraction, i.e. a reduction with the apostrophe, for example "all"' etc...
Finally protwords file indicates a set of words which we do not want to stem, for example "swimming-pool"
With this feature, SOLR enables the indexing, with the document, of all the stems deriving from each text field analysis.

The TFIDF schema (term frequency-inverse document frequency) is a numerical statistic which reflects how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.
In our case we can use the property of each field stored in each document (i.e. the document in which is stored all the features and text of a hotel) which enables us to count the frequency of a term in a document and in all documents (i.e. *docEnum*).
Therefore we can compute the tf-idf value with the following formula:

$$w_{i,tf-idf} = num\_document\ occurrences * \frac{1}{\log num\_total\ occurrences} \tag{4.1}$$

where *num_ document occurrences* is the term frequency for a single document and *num_ total occurrences* is the term frequency for all documents. Finally we calculate the norm of the tf-idf weight as:

$$\|n_i\| = \sqrt{\sum w^2_{i,tf-idf}} \tag{4.2}$$

After the extraction of the stems, concerning features and free text, and the computation of their weights for each items (i.e. hotels), the content manager module manages their insertion into the database in the two tables stem-item flat matrix and stem-item tfidf matrix. This interaction with the DB is

defined through a remote interface for the Java application exposed by the PoliVenusSystem EJB, in detail the *HotelContentManagerRemote* interface of the *HotelContentManager* stateless session bean.

4.5 Test system

The core of the framework is the test system and defines the web application for hotel booking. This module of the framework is based on different technologies and incorporates different languages. It is also extremely modular, factor which is necessary to perform our tests under different experimental conditions.
The testing system is a client-server application, where the technologies adopted at client-side refer to the web application domain and at server-side the logic is implemented with the Enterprise Java Bean technology. The client-side involves HTML, css, javascript, jQuery and AJAX while the server-side is implemented with servlets, Java Server Pages (JSP), entity beans and stateless session beans (SLSB). We can link the components of the application to the MVC paradigm as follows:

- Model: EJB module
- View: client side module
- Control: servlets, JSP

The EJB module is furthermore divided into three logic blocks of session beans and interfaces.
One block is responsible of the management of all methods to access the information necessary to compose the web application; for example the use of the information concerning the hotels to populate web pages: images, descriptions, features, rooms details and information, location information etc...
The second block is strictly connected with the call of the Matlab environment to generate recommendations. It manages the information of the user session during the web application usage and communicates these data in a specific format to the Matlab module; it afterwards receives the response from the module with the list of recommendation ready to be presented to users in the web application.
The third block manages the capture of all information deriving from the tracking activity: the user is tracked during all interactions with the application to analyze data and infer on the variables of our experiments.

The client-side module is divided in two functionalities, a graphic functionality which is obtained with HTML, css, javascript, and jQuery, and a logic functionality which is running in background to retrieve information about users and to send this information to the EJB component to store it into the database. This feature is achieved with the union of Ajax and servlet technology.
Finally the Control of the application, which is composed primary by servlets that both manage the navigation logic of the web application and the asynchronous messages from the client logic. The servlets are divided into three modules, depending on their functionality. One module, the activity analysis, conveys the activity of the user from the client side to the EJB module, and therefore the DB. The second module, namely the controller, manages the web application functionalities without the RS support and finally the last module, i.e. the recommender controller, provides the same functionalities of the second one, with the addition of the recommender support.
In order to experiment the extended conceptual module and to provide answers to the research questions, we realize an architecture which enables the study of RS under different conditions.
Therefore we develop our software to change different variables of the application, with the purpose to achieve a multi-testing environment as expressed in chapter 3.
The involved variables are:

- **RS use**
- **RS characteristics**
- **task**
- **availability**

Other aspects which are crucial in the development of the application are:

- the elicitation technique
- the tracking activity
- the moment and the format of recommendations

We introduce now how the dimensions of the extended conceptual model are implemented, describing in detail the role of model-view-controller components of the framework and how they interact with each other to change the variables of the tests.

4.5.1 RS use

The first variable in our tests represents the possibility to perform tests with and without the RS support inside the application.
The organization of the software is oriented to manage the following issues: on one side the application with RS support introduces new logic components at all the levels of the application (i.e. at level of view at client side, at level of controller within the servlets, and finally at level of the business logic in the EJB module). This is because the RS support needs introduction in the presentation layer of the web application, with the recommendations to be showed to users, in the controller because, as the web pages with the recommendations are different, also the controller must implement the management of the recommender results and use different control strategies between the web pages; finally at the EJB module level it is required to add a recommender feature which enables the interaction with the Matlab environment.
On the other side we must consider that the test system, with or without RS support, shares common logic functionalities: the information about the hotels and their characteristics to be showed in the web application, the management of the user tracking, the begin and the end of the test which correspond to the homepage, the booking simulation page and the final survey of the test. Therefore there is a reuse of some functionalities in both application with and without RS support.
In relation to these considerations, we implement our system with two different modules in the view and in the controller layers and we opt for an integration of recommender functionalities in the logic component, i.e. the model.
At client-side, we divided the web pages generated by the servlets with Jsp in two groups: one group of Jsp has the normal layout of an OTA application, while another in addition has the integration of the results of the recommender used. The *entry point* and the *exit point* of the web pages are common and refer to the homepage, where the task is introduced, and the last portion of the application, from the booking to the end, where users confirm their reservation intention and go on to conclude the survey and the test.
Also the scripts integrated in the application at client-side are shared: their function is mainly related to users' activity tracking and the enrichment of graphic components.
At server-side the controllers are separated in two groups also: the first group is related to the management of the web application with no RS sup-

port (controllers), while the second group manages the control of the web pages with the RS results integration (recommender controllers).
The module which is shared is related to the group of servlets which take care of the results of users' interactions with the web site arose by the scripts at client-side.
The logic, in the model layer, shares the most of its functionalities with the web applications with and without RS support. The two components shared are: the package of the common logic of the application, the *mainlogic* component, which provides information about hotels and their details as rooms features, location, policies, amenities, images etc...; and the package *testing activity manager* which persists all the user activity in the area of the database which saves the interaction during the tests.
The RS integration is obtained through an additional module, *recommender system manager*, which is responsible of the interaction with the Matlab environment.

Matlab and MatlabControl API

Matlab (matrix laboratory) is a numerical computing environment that allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.
In our work, we inherit a set of different recommender algorithms all implemented in Matlab because of its possibility to easily manipulate matrices and complex structures.
The algorithms refer to the three main families discussed in the chapter 2: collaborative-filtering, content-filtering and hybrid. They are all organized in a library which exposes a common interface to interact with such algorithms.
The main issue is how to instantiate a communication between the EJB module, and in detail the recommender system manager with the Matlab environment to call the algorithms library. We solve this problem with a Google API named MatlabControl.
Matlabcontrol is a Java API that allows for calling MATLAB from Java by a connection between the Java Virtual Machine (JVM) of the Java environment of the PoliVenus framework and the JVM in which Matlab runs.
This API is developed implementing a JMI wrapper: using the JMI.jar file (Java to Matlab Interface) included in Matlab, which provides functions to support Java, it is created a proxy between the two Java environments (one

in Matlab and the other of the applications which want to communicate with Matlab). Thanks to this proxy, the system can call functions and execute operations through methods exposed by MatlabControl API like a normal command window of Matlab.

Listing 4.6: Code of the Hotel class and the EJB remote interface to persist all objects

```
MatlabProxyFactoryOptions options = new
    MatlabProxyFactoryOptions.Builder().
    setUsePreviouslyControlledSession(true) .setHidden(true).
    setMatlabLocation(null).build();
MatlabProxyFactory factory = new MatlabProxyFactory(options);
MatlabProxy proxy = null;
try {
    proxy = factory.getProxy();
    //move on the workspace
    proxy.eval(PATHMATLABCODEMYHOME);
    //check if already loaded variables(first time always load
        ...)
    proxy.eval(control = exist(hotelIdList););
    double control = ((double[]) proxy.getVariable(control))
        [0];
    if(control == 0){
        proxy.eval(loadVariables;);
    }
    . . .
    proxy.eval(recListCont =
        computeRecommendationIdListFiltering(signals,
        hotelIdList,@+algorithm+,directContentModelGlobal,[],+
        String.va ueOf(numHotelsRecommended)+,+String.valueOf(
        filteredViewed)+,+filteredVector+););
```

The algorithms in the library work on a two phase-processing: a first offline phase, where they build, in batch processing mode, a heavy and complex model of users based on the data collected; a second online process where the signals of the active user are passed from the application in a specific format and the algorithms operate in realtime execution, sending back the recommendations to the application.
Therefore while the first phase is computed periodically, a part from the system, the second phase is executed during the usage of the application, everytime a request of recommendations is generated by the recommender module of the system.
The online phase depends on the recommender system manager module of the EJB, which interacts with Matlab with the MatlabControl proxy object. The offline phase exposes the following interface to create a global model of

the users:

$$globalModel = CreateModel\_algorithm(URM, ICM, param); \quad (4.3)$$

where the function *CreateModel_ algorithm* makes different operations in relation to the algorithm specified, the *URM* input parameter is the user-rating matrix, the *ICM* parameter refers to the item-content matrix and *param* is a struct with the possibility to specify some extra parameters concerning the execution of the algorithm[6].
The realtime phase instead exposes the following interface:

$$recommendedList = onLineRecom\_algorithm(userProfile, globalModel, param) \quad (4.4)$$

where the output parameter is the recommendations of the algorithm, the *userProfile* is the vector of signals which express the user preferences and ratings, the *globalModel* is the output parameter generate previously by the batch processing for that algorithm, and *param* is is a struct which specifies execution parameters.
To exploit these interfaces, we instantiate a portion of the database to create the structures used by the algorithms. We operate on the data of the database with the other client application of the framework (content manager) to extract the information concerning the users' ratings and the content of the hotels. This because we need to create a structure which contains a relation between users and their ratings, namely URM, and another structure which puts in relation the item (i.e. the hotel) and all its features, namely ICM.
Therefore we create two tables, user-item matrix Venere and user-item-matrix Tripadvisor where we put the column index of the hotels[7], the row index of the user and the rating that the user at that index expresses referring to the hotel at that index.
We create a matrix in Matlab with these tables with the matrix market format of export, and we scale the ratings in order to refer to common values (5 scale values of ratings where 5 is the maximum and 0 is the minimum). This matrix is further translated in order to have values around the zero (positive and negative) and it is finally generated the global collaborative model.
The same procedure on the database is executed to obtain the item-content matrix. In this case we used the client application content manager to execute the task in batch processing.

[6]For example the latent size of the matrix of the model during the expansion

[7]The index refers to another table where at the index is associated the identifier of the hotel

Data are divided into weighted features, in the stem-item flat matrix table, and the free text on which stemming algorithms are processed and then on each stem is computed the tf-idf value. The result of this processing is stored in the stem-item tfidf matrix table. These two tables are exported in the matrix market format and are joined in a matrix inside Matlab, from which is computed the content global model.
The last issues we have to overcome are related to the online interface and the results provided. The first issue is the sorting of the results of the algorithm.
The recommended list generated by the online interface presents results which are not ordered in relation to the predicted rating of the user. Another problem is that to be ready, the list of recommendations should be referred to the identifier of the hotels and not to the index of the column of the matrix.
The third issue is that while in some domains, like movies, we may want to exclude items already viewed, the same is not true for the tourism domain. In fact we may desire to see again a hotel which we already saw in the past but it was not available for some constraints.
Finally, we want to obtain in our recommendations only items which respect the constraints fixed by the user, e.g. range of price, category of the hotel etc...
For these reasons we implement a Matlab function to wrap the online interface exposed by the library and manage all the aforementioned issues. The interface of the function we call from the EJB module is the following:

$$\begin{aligned} function[idsOut, ratingsOut] = \\ computeRecommendationIdListFiltering(userProfile, \\ ids, onLineFunction, globalModel, onLineParam, N, \\ filterViewedItems, availableIDs) \end{aligned} \tag{4.5}$$

where the output parameters are the ordered list of hotels identifiers (*idsOut*) and the predicted rating of the user (*ratingsOut*); the *userProfile* is the vector of signals sent by the application to Matlab, *ids* is the list of identifiers for the list of hotels considered for the recommendations, the *onLineFunction* is the name of the online interface whose name depends on the algorithm, the *globalModel* is the model generated from collaborative algorithms[8], the *onLineParam* is a struct of special operative parameters[9], N is the number of recommendations requested by the application, *filteredViewedItems* is a

[8] In the case of content algorithm this parameter is an empty matrix
[9] In our case is an empty matrix

boolean which indicates if we want to include or exclude viewed items, and finally *availableIDs* which represents the list of items which respect the constraints of the user (filter conditions). After cooperating with the Matlab algorithms library, the recommender system manager module receives the list of recommended hotels and provides it to the servlets of the recommender controllers module which generates the server pages of the recommender group to the user (therefore final HTML web pages). If the selected test is with no RS support, the recommender system manager module is disactivated and the normal items and features are provided to the servlet of the controller module by the mainlogic module. The controllers (i.e. servlets) then generate server pages (therefore final HTML web pages) with no recommendations.

4.5.2 RS characteristics

Other variables we want to modify in our tests are related to the RS characteristics. Into this dimension fall the *RS type* and the *Preference elicitation method*. Actually our framework can vary only the RS type, while the elicitation technique used is the same for every variant of tests.
We discuss now the RS types enabled by the system for the testing, we will discuss later about the latter.
The RS type refers to the algorithm used from the RS to generate recommendations and therefore it is an important operative factor in our research. In this work we argue many times about the importance of changing algorithm to evaluate how users react about this change.
The variation of the type of algorithm used can be done at two different levels: it is possible to choose between different algorithms, ranging from collaborative to hybrid families; it is also possible to create a combination of different recommendation lists with a *hybrid interleaved strategy*. For example a content algorithm produces a recommended list whose items are presented interleaved with the items of a collaborative recommendation list. In our system, the selection of the strategy occurs in the homepage, where it is possible to specify this parameter with the URL. The choice of the strategy for the current test is made through the query string:
http://.../homepage.html?s=i&t=1&d=true where the parameter *s* indicates the strategy and the values *i, s* indicates the hybrid (interleaved) presentation, or single algorithm presentation.
The second level of choice is the selection of the algorithm. The selection is expressed in a *configuration file*, where the names of the algorithms with which the system will call the Matlab library interfaces are written. In the

case of an interleaved strategy, there must be specified two names. An example of this file is reported below:

Listing 4.7: The configuration file

```
/**WRITE THE ROW BELOW THE NAME OF THE ONLINE M FUNCTION OF THE
    COLLABORATIVE ALGORITHM**/
onLineRecom_pureSVD
/**WRITE THE ROW BELOW THE NAME OF THE ONLINE M FUNCTION OF THE
    CONTENT ALGORITHM**/
onLineRecom_directContent_knn
```

With this settings it is possible to choose the type of RS to use during the test session, moreover it can be decided if to use a single algorithm or a combination of two.
The result of the choice are managed by the recommender controllers and the recommender system manager. While the combination of algorithms, single list or interleaved combination, is managed by the recommender controllers which request the recommender system manager one or two recommendation lists, the recommender system manager module reads the configuration file and calls the Matlab library interface with the name of the selected algorithm/s.

4.5.3 Task

As argued in the chapter 3, one specific need of the extended conceptual model is to verify the reaction of users towards RSs when performing different tasks which are oriented to different goals. To study these variables under this constraint, our frameworks must manage different tasks between which to choose at the start of the test.
In order to maintain flexibility, we define a table in our database where we insert all the tests we want to perform. This table has the following fields:

- test identifier
- kind of task
- name of the test
- request
- is recommender
- is available (the resource)
- is last

- is active
- distance

Test identifier identifies the test. It is a value manually assigned, which must be spaced from the other identifiers[10].

Kind of task is a string representing the scenario. In our case we have two typologies of tasks, based on the business and holiday scenario; therefore the value of task in our case is business and holiday.

Request It is the string representing the task assigned to the user. In our case the task for the business scenario is:
book a room in a hotel in Rome. The period of the reservation is from 16 July to 18 July. The characteristics of the hotel are:

- three stars or superior category
- single room price for one night not superior to 120 €
- single or double room

Is recommender indicates whether the test is with RS support or not.

Is available condition specifies if the system simulates the hotel unavailability or not (see next section).

Is last is a boolean which indicates whether the test is the last test performed by a user or not.

Is active indicates whether the test is considered active or not, i.e. in an automatic sequence selection the system should consider the test (active) or jump it and select another one.

Distance indicates the absolute distance (in module) between a test and another, taking as reference the identifier of the tests.

[10]This condition is due to the fact that the id is inserted in the homepage URL so the user can modify it and get a valid test if identifiers are not spaced enough

The policies which assign the tests to users are two: a sequential order and a manual choice. In the sequential order, the user who starts the test, reaches the homepage and the system automatically selects the "next" test with a sequential order. The system in fact, depending on the value of the distance field, selects the next active test respect to the last one executed. After the selection, the test is assigned to the user and is marked as the last and the distances between all tests are updated.
In the manual choice, the tests are selected manually by the URL of the homepage of the application. For example
*http://.../homepage.html?s=i&**t=1**&d=true*
indicates in the query string that the test assigned is the one with identifier 1. With this strategy the administrator has full control on the application and on the tests assigned.
The assignment is concluded when the table "assignments", which represents the relationship between the user tested entity and the test, has the tuple with the link between the actual session_id and the test identifier. This way the user, which is identified with his/her HTTP session, is linked uniquely to a test.
The assignment involves all the layers of the framework: the EJB module, more specifically the *testing activity manager* component, is responsible for the detection of the selected test and the updating of the test attributes; the choice of the test, is sent back to the servlet controller which stores the test in the HTTP session and passes it to the homepage as an XML response of an Ajax asyncronous request. The XML response is parsed within a Javascript and the div object of the homepage is filled with the text of the task.
Finally to verify that users respect the constraints imposed by the test, we implement some controls for dates, hotel typology, and city which trigger in the last phase of the test, at the booking confirmation right before the final survey. If the user does not respect the constraints, the reasons of the mistake are showed and he/she is asked to return back and change preferences according with the task.

4.5.4 Availability

The availability or unavailability of the items, i.e. hotels, is a concept related to the "bounded resource" theory. In order to study this theory the framework must implement different conditions of items availability. The items we take in consideration are the rooms, and therefore in our simplification the entire hotels. The typologies of unavailability we implement are three:

1 preferred hotel always available (i.e. at any checkin checkout dates)

2 preferred hotel not available for a specific range of dates

3 preferred hotel never available for any range of dates

The strategy to select the range of dates and the unavailable hotel is based on the "tentative" of booking. If the current test has the parameter *is available* set to 0 (typology 2) or 2 (typology 3), when the user confirms the hotel and expresses the desire of making the reservation, the system redirects the user to a list of hotels, explaining that the hotel is not available for that dates. Subsequently, if the unavailability typology is the number 2, the system stores the selected checkin and checkout dates and, in correspondence of this period, the hotel previously chosen will be unavailable (this to grant the consistency of the system), while with the typology 3, the hotel chosen will always be unavailable, independently from dates.
This bounded resources simulation is accomplished in the control layer: both the servlets controllers and recommender controllers can exploit two objects saved into the current session.
The first object is an instance of the UserSession class, which stores all the attributes of the session: the period, the city, the number of rooms and guests, all filter attributes used and the sorting menu, the type of visualization of the list of hotels, and if the examined hotel is a recommended one.

Listing 4.8: UserSession and CheckConsistency classes

```
public class UserSession {
    private String city;
    private String date\_checkin = ;
    private String date\_checkout = ;
    private int days = 1;
    private int guests = 2;
    private int rooms = 1;
    private int offset;
    private String visualization\_type = lista;
    private String filter;
    private String price\_range;
    private String rating;
    private String type;
    private String city\_zone;
    private boolean recommended\_hotel;
    private Test test;
    private String algorithm;
    . . .
}

public class CheckConsistency {

  private int hotel;
  private String checkin;
  private String checkout;
  private boolean already\_blocked;
  private boolean blockable;
}
```

The second object is an instance of the CheckConsistency class, whose attributes are the identifier of the unavailable hotel, the checkin and checkout dates, a boolean, already blocked, to verify if there is already a hotel unavailable for the current session, and another boolean, blockable, to verify if this test expects the bounded resource simulation or not.
Thanks to these two objects stored in the HTTP session, the system can manage the bounded resources simulation and at the same time grant the consistency of the information provided to the user.

4.5.5 The elicitation technique

The elicitation we implement in our framework is based on implicit techniques. Although we do not actually study this characteristic of the RS as a variable of our conceptual model, it is important to deepen this matter to understand how the RS interacts with tested users.

At first a comparison between explicit and implicit techniques is done: an explicit elicitation technique is a strategy to obtain the information of the user in an explicit way. For information we mean a set of preferences, needs, and constraints which determine a user profile which can be exploited to infer recommendations. The most famous explicit information request techniques are used by the *conversational RS family*, where the user interacts directly with the system which can therefore build a user profile from explicit questions.
Another way to profile the user is to ask before the task to rate some items.
As aforementioned in the chapter 2, there are three types of information which can be used to infer about a user modeling.
One type is the *long-term signals* that users leave to the system while they interact. This long-term information represents a classification of users' tastes and needs and it is considered long lasting in the user model. More in detail these preferences can be summarized as an overall rating that users give to items.
Conversely, the second type of information, the short-term, is based on *contextual information* that is not related to the person but the surrounding: the particular period in which the recommendations are provided (for example Christmas or summer holiday), or the location (for example a car, at office) etc...
This kind of information is exploited to build contextual models which can express an actual, strong preference towards some items and therefore more useful in some circumstances than the long-term one.
Finally the third type of signals that a user can release is the *metadata information*, which express not a direct information about items but on what characteristics the user is more susceptible (i.e. location or the price ...).
Therefore it is possible to weight in different ways the features on which is based the RS analysis.
Although in our work we use only long-term signals, we also provide some recommendations which exploit the context in which they are provided. Our context is not considered as the set of external parameters independent from the user actions but related to the detail page of a specific hotel he/she is examining. Therefore when the user is in a detail page of a hotel, we neglect previous preferences (long-term or historical ones) to improve the influence of the actual page seen (contextual).
In our work we consider an implicit elicitation which is based on the discovery of user's interests, depending on his/her interaction with the system. This method of elicitation, in addition to being more complicated than explicit one, is also more "realistic" for the e-booking domain, as users do not

want to provide information about themselves.
Our implicit elicitation technique is based on *signals* that the user provides to the system when he/she interacts with it. In our system, this interaction consists of the objects (e.g. links, buttons etc...) of the interface that the user clicks. Everytime this event happens, the system records it and assigns a score to the item which the event refers to (i.e. the hotel).
The main issues of this elicitation technique are the following:

- set a range of values for the signals
- the structure of the user profile
- how to evaluate the interaction history

The first issue depends on the level of navigation where the user generates the signal for the system. Our web application, inspired by the main OTA in the sector of the e-Tourism, is composed by two different levels of depth:

- the list of hotels
- the detail of a single hotel

We map the signals generated in the list of hotels less important than the signals of a detail page of the hotel. In particular we apply a scale of weights from 3 to 5. The objects we refer to in the web page with the list of hotels are:

- all the links that sends to the detail page
- the button "check availability"
- the popup menu links in Google maps

For all these signals, except the check availability button, we attribute a weight of 3. For the check availability button instead we attribute a weight of 4. This is because we think that the button with its label is more representative of the interest of the user on the hotel then other signals.
For the detail page, we manage the following events:

- link to open the text of the reviews
- button to check available rooms
- button to book the room (not definitive)

To the first object we attribute a value of 4, a high interest on the item, to the signal; for the button to check the rooms availability or to book, we give a weight of 5, strong interest, to the signal.
Hence, with all these signals, we can build a user profile contextual with the pages visited by the user.
The structure of the user profile is a map composed by a couple key-value, where the key is the hotel identifier for which a signal is generated; the value is the weight for that hotel[11]. This object-oriented representation can be exported into the Matlab environment as a vector with one row, i.e. the actual user profile, and the number of columns equal to the number of items. For each item in the vector it is assigned the weight corresponding to the signal saved by the system during the interaction. Therefore we have a vector with one row and many columns where there are many zeros (for items with no signal generated) and some weights ranging from 3 to 5.
This process of elicitation involves all the layers of the framework, from the model to the view layer. In fact signals are captured at client-side by HTML objects functions (e.g onclick() etc. . .) or specific Javascript for more elaborated components.
This signals are sent both synchronously or asynchronously to the recommender controller module at server side, in the controller layer.
These servlets then generates a user profile through a method exposed by the remote interface of the recommender system manager module, in the model layer. This method manages the mapping item-signal and generates the vector in Matlab environment.
The last issue concerns the evaluation of the interaction history to build the user profile. In our web application, the RS support can take over in three moments of the interaction:

1 in the list of hotels through the sort menu

2 in the list of hotels after the booking tentative, as a consequence of the unavailability

3 in the detail page of a hotel

In order to experiment different strategies, we provide the possibility to choose between a user profile which doesn't consider the past history and takes into account only the current item (i.e. only the last signal is considered), and another type of user profile which considers the history of rated items, decreasing the weight of the oldest ones. This can be obtained in our

[11] If more than a signal is generated for the same hotel, it is kept the highest value

implementation by multiplying for 0,5 the weight of every value in the map everytime a new signal occurs. Therefore all weights are correlated to the moment when they are generated.
Referring to the three moments of recommendation, we use both strategies of user profiling (long-term and contextual): for recommendations provided as a consequence of unavailability or in the detail page, we use a contextual user profile in order to attribute a strong importance to the single hotel actually examined by the user; in the case of recommendations requested in the list of hotels, we implement a long-term strategy of user profiling where we consider all signals of the different hotels generating during all the session.

4.5.6 User activity tracking

The user activity tracking involves all the layers of the framework: the client-side, the servlets of the activity analysis module and finally the testing activity module of the EJB.
This process is monodirectional, from the client, which captures concretely the activity, to the server, which redirects all data to the EJB component and hence the database. In order to simplify the client-side, we decide to implement some functions in javascript to capture interaction and send data with Ajax asynchronously to the server, rather than use specific client software to analyze the browsing activity[12].
The parameters we store in the database tables are the *objects of interaction* and the *space coordinates*.
The first parameter refers to all the objects "active" in the web pages. The term active indicates the set of objects for which is enabled a javascript function which handles some events like the click or the mouse passage. These events are sent to a specific servlet of the activity analysis module which collects all the received parameters on the event handled and sends these information to the testing activity module of the EJB. All these data are at the end persisted into the database.
The second parameter captured during the interaction, namely the space coordinates, refers to the position of the mouse in the web page in terms of X-coordinate and Y-coordinate. These two coordinates are registered as a consequence of a click event. In the handler of this event are specified the lines of code to record these coordinates and send them asynchronously with Ajax to a specific controller: the servlet *heatmap controller*.
This information is redirected to the session bean *testing manager* which

[12]Softwares like SurveilStar make the client too heavy to be managed from a computational point of view

User id	Session id	Page id	Page type	Object	Object type	Timestamp
1	6D9E9E	homepage.html	homepage	start request	start button	2012-07-08 17:00:19
2	6D9E9E	index.jsp	index	Milano	location	2012-07-08 17:00:25
3	6D9E9E	lista hotel.jsp	hotel	156	image link	2012-07-08 17:00:29
4	6D9E9E	overview hotel.jsp	detail	156	location tab	2012-07-08 17:00:31
5	6D9E9E	overview hotel.jsp	detail	156	box review	2012-07-08 17:00:39
6	6D9E9E	overview hotel.jsp	detail	prenota review	link button	2012-07-08 17:00:41
7	6D9E9E	overview hotel.jsp	detail	Milano	location	2012-07-08 17:00:47

Table 4.4: User activity table

provides a method through a remote interface to store the coordinates in a specific table of the database, namely *heatmap analyis*. In this table are

Figure 4.5: Heatmap analysis

summarized all the the behaviors of users in relation to the space of the web application. With this spatial information, we implement a script which receives the X-Y coordinates and builds a graphic colored area which is proportional in dimension and colors to the density of the user activity. The main div is overlayed with this area which is also partially transparent. This map, called *heatmap* for its colors[13], is complementary to the analysis effectuated with the active objects of the pages: the heatmap is a representation of the activity in all available space, while the active objects inspection indicates what objects are more attractive for users in the usage of the website.

4.5.7 The moment and the format of recommendations

As aforementioned, to build our web application for e-Booking in the tourism domain we refer to the most important websites of OTA of this domain (e.g. Venere.com, Expedia.com, Hotels.com).
A common property that these websites share is the conceptual separation of the content into a list of items and a detail of the item. In detail their levels of research of an hotel are two, a level in which is showed a list of hotels, with some features provided to filter those results, and a single hotel level, where are presented all the features of the specific hotel.

[13]Red indicates high activity density, dark blue low activity density

Following this organization, we also structure the web application in two

Figure 4.6: Hotel list web page

levels, one with the presentation of a list of hotels in relation to expressed constraints of search, and another level where the details of a hotel are presented to users.
In the list of hotels we provide users two possible interactions over such list: a filtering mechanism, based on category, budget, location, rating and name filters, and a menu for sorting such results on price, opinion and stars.
Beyond this we introduce the possibility to see the location of the hotels in

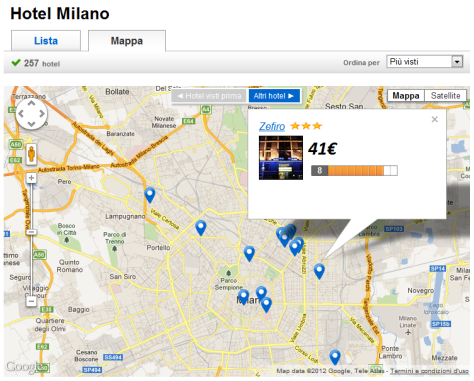

Figure 4.7: Map visualization of the list of hotels

a map.
In the detail page, we define another division based on general descriptions, location description and reviews of the users.

In this context, following the guidelines for recommendations suggested by Pu et al. [35], we decide to insert the recommended items both integrated into existing components and into a dedicated area. In the first solution, we opt for an integration of recommendations into the list of hotels, with the addiction of another menu option, *"recommended for you"*. Due to the space

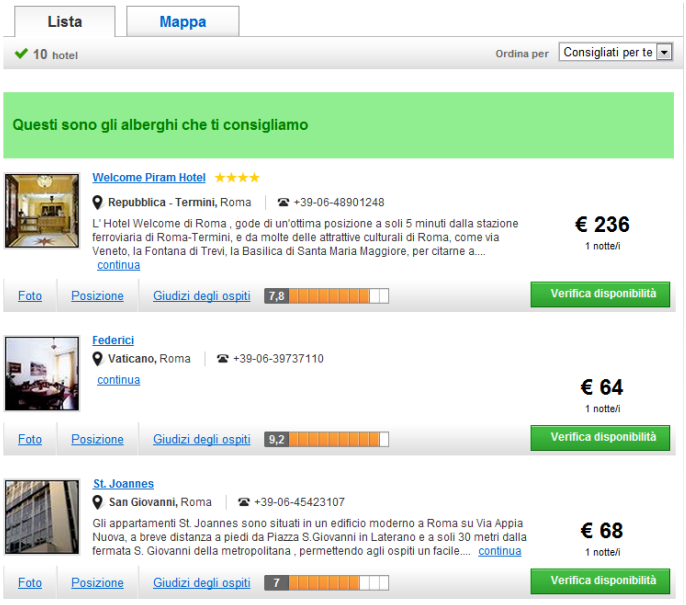

Figure 4.8: Recommendations in the list page

available, the recommendations are introduced with the same information of the normal list content: a short description, a thumbnail, information about user ratings, and links to its detail page. The second solution is implemented considering the lateral free space in the detail page. Here we organize this area with a list of recommendations organized in boxes and well visible to users. In this case, due to space limitations, we must simplify the content inside the recommendations. Therefore every recommended hotel presents a little thumbnail, the name, the location, the price and the rating information. From this box it is possible to access the detail of the page.
Finally the application, when requested the availability check by the user, shows in the detail pages of hotels the combination of rooms available for the selected checkin and checkout dates and the rooms and guests selected. After confirming all the options, the user reaches a summary page which contains all the selected booking option. This page represents the definitive passage for booking: after accomplishing these options the user cannot change the preferences anymore and the booking is completed.

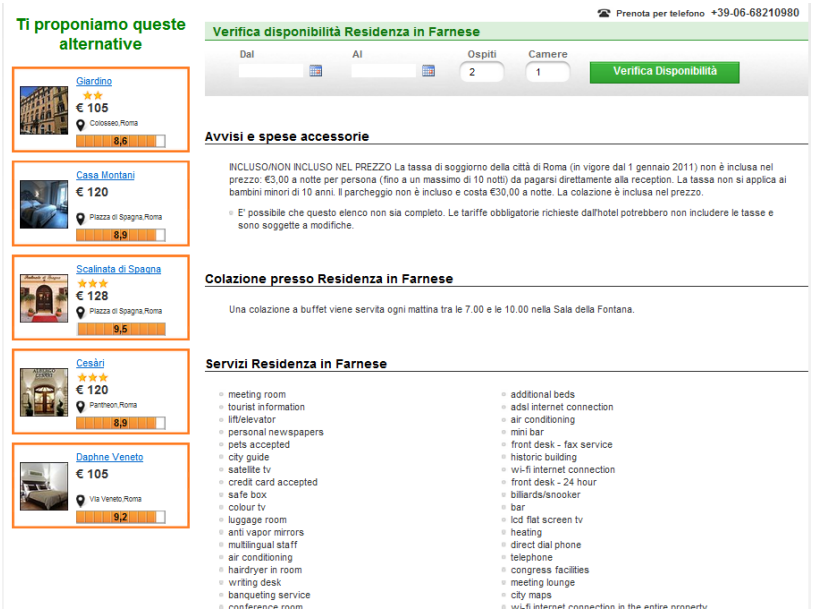

Figure 4.9: Recommendations in detail page of a hotel

The user is therefore redirected to the survey which stores the impressions about the experience with the application.

Figure 4.10: The final survey

Chapter 5

Design of the preliminary empirical study

In order to provide answers to our research questions and to validate our propositions, we propose in this section different scenarios users are subjected to. These scenarios have been developed in relation to the extended conceptual model, with the purpose of investigating the new conceptual construct (namely the task), the new relationship between evaluation of RSs and user decision making, and the new feature under which items are considered (namely the product availability).
In the next paragraph we at first give a description of the scenarios and of the variables involved in our experiments in relation to the variables of the model, providing a detailed description of the different empirical studies we intend to perform. Then we explain the metrics and the instruments to execute the experiments, ending with the planned execution.

5.1 Scenarios and variables considered

Although our conceptual model and the derived software framework let us investigate on such vast number of postulates, for now we refer to a small number of statements in our empirical work.
In detail we test the following propositions postulated in Xiao and Benbasat model:

P1: RS use influences users' decision effort;

P2: RS use improves users' decision quality;

P14: RS type influences users' evaluations of RSs;

In addition we try to answer the following research questions:

i How do RSs influence users' decision making process when users have different tasks?

ii Do the limited availability of on-line items and the corresponding time-pressure on the user influence users' evaluation of RSs?

iii Do users' evaluation of RSs influence the decision making process?

To answer such research questions, we state three propositions we want to test (i.e. test propositions TP) with our experiments:

TP1: Users' evaluation of RS influences perceived quality of the decision process

TP2: Different tasks influence in different ways the user decision making process when using the RS

TP3: Under bounded resources conditions, RS use influences users' perceived quality of the decision process and decision outcomes effort at a higher degree then under unlimited conditions.

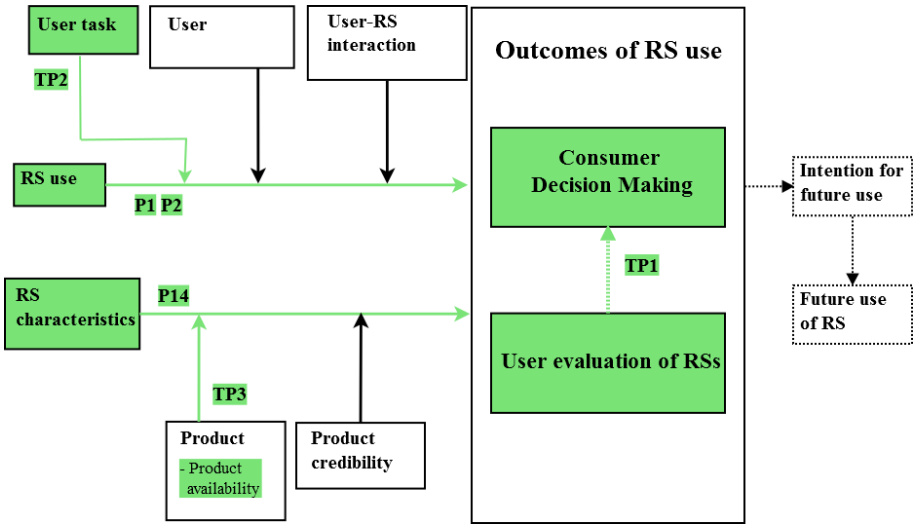

Figure 5.1: Constructs assessed with the empirical study

In order to provide answers to the research questions and validate our propositions we decide to perform different experiments, depending on two different scenarios.

Business scenario The first scenario proposed is related to a business goal.
We ask the user to book a room in a hotel in Rome. The period of the reservation is from 16 July to 18 July. The characteristics of the hotel are:

- three stars or superior category
- single room price for one night not superior to 120 €
- single or double room

The main aim in this scenario is to evaluate the impact of RS assistance under bounded rationality and *strict goal* conditions. The first constraint, i.e. bounded rationality, is due to the large amount of items the user must face (there are about one thousand and five hundred hotels in Rome), while the second, i.e. the strict goal, indicates specific restrictions, typical of this kind of journey.

Holiday scenario In this scenario we ask users to book a room in a hotel in Rome. The characteristics of the hotel and room are:

- double room
- no constraints about dates
- no constraints about hotel's category or price of the room

Under these conditions the aim is to evaluate how users perceive the RS support under bounded rationality constraint but with no restrictions about the features of the item, i.e. *soft goal* task.
The typical scenario which can realize such requirements is the holiday planning, where the user has no specific constraints about the dates and can choose his/her preferred policy to plan the vacation: for example an economic holiday with the partner or a travel with the family etc ...
These two scenarios enables us to compare how users react with respect to RS support under bounded rationality condition but with different type of tasks, one with strict goal and another will soft goal. In fact it can be realistic to think that a user, who is stressed by numerous requirements, may perceive better and accept more a RS, considering its support necessary to avoid the repeat of all the screening phase. Under this light the effect of reducing the cognitive effort can be referred to an evaluation of the RS impact on the user decision making.

Beyond this consideration, there is also the possibility to expand the typical functionality of RS of recommend similar items with the ability to suggest new interesting items for the user (for example exploiting collaborative recommendations). Therefore evaluating the user reaction under different tasks in relation to the features of a RS is another aim of these scenarios.
On the other side it is also interesting to compare the results in terms of the evaluation that a user gives to a RS under different kind of tasks.
Another aspect we want to consider is how is perceived the RS support when, although the complete space of items is vast, the user faces a *bounded resource* condition, for which the preferred choice may not be available.
Therefore our challenges are explored defining the following variables in our experiments:

i RS use

ii the task to be performed by tested users

iii RS characteristics

iv item availability

For each of these variables we test different combinations per each scenario.

5.1.1 RS use

This condition, in our empirical design, is identified with the possibility to test users with two applications: one which simulate a normal OTA use, with no support of a recommender system; the second which replicates an OTA application with the support of our recommender system.
In our testing environment we enable the coexistence of the RS support functionality with the normal use of the system. Hence we have the possibility to test the system with no RS support and the system in which the user may request the help of the recommender, enhancing the complexity of the analysis.

5.1.2 The task of the scenarios

The second variable is the task, defined as the triple action, object, goal.
Starting from the action, we ask our users to make a reservation of a hotel. This must be performed in our software platform which works in the aforementioned modes.
The object identified is a hotel room, which the user must choose respecting

the action of the task assigned to him.
The room of a hotel is a very complex[1] object, which in addition shares some of its property with the hotel itself. Therefore the object of our task can be seen as a hierarchy composed by a highest level represented by the hotel (which, containing all rooms, has all the properties of its sub-elements), and a lower level defined by the rooms, in which some properties of the hotel may be not valid (for example the features associated to the top class rooms, in the case we search a middle class room in the same hotel).
Because of the complex and hierarchical nature of the object, the activity of the user is a very "rich" task, where the screening and the comparison of the decision process can be executed at two different levels of depth.
The goals we consider in our research are two, *business goal* and *holiday goal*. In the first, i.e. the business goal, we imagine to test our system and make our analysis on a typical scenario which occurs in a work environment: a person who must travel alone for job purposes. In this context we impose strict limitations on the task, specifying a range of price for the hotel, a category of the hotel, a location, and the dates of check-in and check-out. With these constraints, this goal is defined, using the terminology of goal oriented requirements engineering [44], as *strict goal*: "a state/target that the user must reach". For example, "having a passport delivered." is clearly an hard-goal, because it defines a precise condition to reach the state target.
In the second goal, namely the holiday goal, we relax the test conditions of our system as we refer to a typical scenario of holiday planning. In this activity we imagine a travel for a vacation; in this scenario therefore we eliminate previous constraints as we let the user book in a vacation period but with flexible dates, with no specified category of hotel as well as no specification about the price[2]. This goal, in contrast with the business one, is defined as a *soft goal*, which is used to specify, at a qualitative level, not sharply-cut objectives.
Their precise definition requires the development of further details, whose specification is left to the users.

5.1.3 RS characteristics

The RS characteristic of which we enable the comparison is the algorithm used to perform recommendations.
As we describe in the next chapter concerning the framework architecture, we

[1] We follow the definition of Xiao and Benbasat discussed in the first chapter

[2] Because our database has only information related to hotels in Italy at the moment, we ask and offer the possibility to book in Italy

enable the possibility to test a multitude of different families of algorithms, ranging from collaborative and content filtering to hybrid.
At the moment, in our tests we use:

- a collaborative algorithm, namely the pureSVD;
- a content algorithm, namely the k-nearest neighbor algorithm (KNN);
- an hybrid obtained by a mixed combination;[3]

Beyond the possibility to switch between different algorithms, we consider other three aspects, which at the moment are implemented as single solutions but in future works can be area of further analysis and comparisons:

- *the moment in which recommendations should be proposed during the interaction*
- *the format of recommendations*
- *the integration strategy between the application and the RS*
- *the elicitation technique*

The moment of recommendation The first issue is very important since the moment of the interaction between the user and the RS determines an impact on the behavior of the user in performing the task and on the decision making process.
Although the scarcity of studies in which is discussed a strategy of integration between RS's support functionality and web applications in coexistence, we identify three points to introduce recommendations:

- the search list level, where recommendations can be requested by users through a menu
- the search list level, after room unavailability occurs
- the detail page of a hotel, as suggested proposals by the RS

The first introduction level indicates a direct request from the user to access recommendations of the RS. The interaction is enabled through a menu which can order results by price, rating from users, category of the hotel, and "suggested for you" criteria.
The second level of recommended items is always in the list of results, but

[3] In the chapter 2 are explained the different techniques for obtaining an hybrid recommender algorithm

in this case such list represents alternatives which are presented after the system has signaled a room unavailability. This situation can occur when a user, who is interested in a hotel, examines its detail page to verify its characteristics. After selecting the period of reservation, the user can be redirected to the search list level because no rooms are available[4].
The difference with the previous situation is that the list represents a different logic level: while the first is obtained as a consequence of the interaction desired by the user, the second is directly proposed by the system to decrease the frustration of failing the reservation, offering items which are similar as much as possible. Therefore the main support of RS in this case is the avoidance of repeating all the screening and comparing stages again from the beginning.
The third point of introduction is identified in the detail page of a hotel. In this case the recommended items are inserted in lateral boxes to catch attention on new items, and therefore providing a *seek for inspiration* functionality.

The format of recommendations The format of recommendations is another fundamental parameter which is taken into account. As proved in different studies [35], the appropriate balance between text, pictures and other factors can influence user's perception of the usefulness, ease of use, and expertise of a RS. Beyond these aspects this variable is really important also to inspire users a sense of trust towards the system, because omitting some attributes in the presentation can be perceived as a tentative of deception.
It should also be noticed that the format of recommendation is in strict relation to the integration strategy of recommendations into the application.
As a consequence of these assumptions, we defined two types of format: one full and one reduced.
In the first type, we include in the recommendation all the features which an item has in the search list of the web application. For our application they are the name of the hotel, a short overview, the location, a representative image in a box with reduced dimension, a users' rating indicator, the minimum price to book and the category label.
In the second type, because of space limitations, we presented recommendations with a reduction in the number of attributes: in our application they are the name, the category, the location, the minimum price to book, the

[4]This event happens under the simulation resource scarcity, see the variables of the empirical studies

users' rating indicator. Our intention is to provide all essential elements for decision, eliminating not priority attributes.

The integration strategy As argued from Pu and Chen [35], the two main integration strategies of a recommender are the integration between existing components of an application and an integration in a dedicate area of an application. According with the paragraph which explains the moment in which recommendations occur, we followed both strategies: in the search list level we opt for an integration with existing components of the application, in order to not upset the interaction with the website and to make recommendations more familiar respect to the other outputs of the application.
In the detail page of a hotel we propose the recommendations in an dedicated area, where a reduced number of important features are enclosed in a box which form the list of suggested items.

The elicitation technique The last characteristic connected with RSs in our research is the elicitation method. In our work we develop an implicit elicitation technique based on input signals given by user during the interaction and captured via software by the framework. These signals reflect the interest of users on some features or preferences and are distinguished with different weights. These weights are calculated basing on the in-depth search of the user: the more deeply he/she analyzes the content of a hotel, the heavier are signals generated during the navigation, i.e. the more interested is the user in the item. Such signals are collected through the navigation in different points: at the level of the list hotels or in the detail page. The components which generates input for the recommender, i.e. such signals, are all parts of the content of a hotel: its description link, the picture, the button for checking availability, a map popup etc ....
The typology of signal, its representation and its contribute to form the user profile vector it is discussed into the framework chapter.

5.1.4 Resource availability

The variable of the resource availability is declined into two dimensions:

- item availability
- time scarcity[5]

[5]At the moment this feature has not been implemented yet, although the system is set

The first parameter indicates that a room, which is the object considered in our task, can be unavailable for the selected period. In our research the problem of resource availability is simulated with three different strategies: in the first strategy, the hotel has rooms which are always available in every period of booking; in the second strategy, the first selected hotel has all rooms which are not available only in one specific range of dates, which corresponds at the first booking tentative of the user. The third strategy is implemented denying users the availability of the rooms of the hotel selected at first booking tentative in every range of dates.
The purpose of the introduction of this variable into our experimental studies is to stress users about their preference choice. This solicitation is used to evaluate the influence of RSs on the decision making process and the user's evaluation of RS, under constraints of bounded resources discussed in previous paragraphs.
The time scarcity parameter defines a dynamic status of the items where, as time passes, the item can modify its status from available to unavailable. In our tests, controlling the session time, we expect to modify the status of hotels which may become unavailable in a range of dates depending on some established policies[6].
Hence, considering the resource availability variable in our tests, we aim to modify the decision making process of Xiao and Benbasat into a more dynamic process in which the stages of the process, namely screening phase and the comparison phase, are cyclic. Therefore, as stated in our third proposition, we expect that under these constraints RS moderate both user's behavior and his/her perceived usefulness of the RS itself.

5.1.5 Tests' variables summary

Here we summarize all tests combinations and all variables involved in the empirical design in table 4.3 In the table, the condition of unavailability is related only to the item, not the time, with a policy of implementation based on the tentative (it is expressed in the paragraph above). The three values of the variables refer to the unavailability strategy. The field "yes, at every tentative unavailable" refers to the item, which is always not available during the whole session with the user. The field yes for the unavailability variable indicates that an item is unavailable in the dates corresponding the first

up for the purpose

[6]In time-independent resource availability simulations discussed above, we use a policy based on the tentative: the dates in which rooms are unavailable are set up equal to dates of the first tentative

Scenario	RS Use	unavailability-bounded resources
business	no	no
business	no	yes
business	yes	no
business	yes	yes
holiday	no	no
holiday	no	yes
holiday	yes	no
holiday	yes	yes
holiday	no	yes, at every tentative unavailable
holiday	yes	yes, at every tentative unavailable

Table 5.1: Summary of variables combination for tests

booking tentative. The field no for unavailability indicates no constraints for bounded resources analysis.
RS use indicates if in the section it is adopted the RS support in the different parts of the navigation.
The scenario refers to the type of task to be performed. In order to measure our three research questions, namely RQ1, RQ2, and RQ3 to which correspond the three proposition TP1, TP2, and TP3, in addition to three propositions, P1, P2, and P14, of the Xiao and Benbasat model, we have to define some measurements metrics for the following variables:

- user decision making
- user's evaluation of RS
- user-product related factors
- the task
- product availability

The metrics derived from those variables, in addition to the RS characteristics taken in consideration expressed above, enable us to study the impact of such variables on the user decision making and his/her evaluation of RS under the different conditions of our tests. In the figure 5.1 we highlight which parts of the extended model we actually study in the experiments.

5.1.6 Decision making metrics

In the chapter 3 we define as the main variables of the construct the *decision processes* and the *decision outcomes.*

Decision processes

The decision processes variable we consider in our work is the decision effort. In our model such variable can be measured with three metrics:

- decision time
- extent of product search
- amount of user input

All of these parameters are collected via software by the framework.

Decision time is computed as the difference between two timestamps: the begin of the task and the end of reservation, i.e. the moment when the user decides to make the purchase with an irreversible simulated transaction. This variable is stored, like all other parameter traced, into the testing partition of the database of the framework.

Extent of product search The second variable, the extend of product search, is defined as the number of alternatives that have been searched and for which detailed information has been acquired. We measure it through the tracking of the user's navigation in the website: hotels for which detailed information is searched are calculated as the number of accesses at their detail page.

Amount of user input is the number of information provided from the user to get results from the system. This amount is calculated as the sum of the interaction between the user, the menu, and the filter. The menu enables a different ordering of the presented results by price, judgment from other users, category of the hotel (e.g. 3 or 4 stars), and eventually (for RS use) the suggested for you option which provides the ordered list of recommendations. The filter instead enables the user to change the budget of the searched hotels, the location of the city zone, the category, and the type of structure (i.e. bed&breakfast, residence etc ...).

Decision outcomes

The decision outcomes is evaluated by a set of variables:

- choice

- decision quality(objective)
- decision quality(subjective)

Choice The first variable is defined as the final choice of the user between the alternatives. It is saved into the db at the moment of booking submission.

Decision quality objective/subjective The second variable is the decision quality, which is the objective/subjective quality of the consumer's purchase decision. We study the objective decision quality in two ways: through software and through some questions of our final survey. We analyze the preference matching score, which is measured by the score of how the chosen alternative matches the consumer's preferences; we provide a measurement of the quality of consideration set, i.e. the average of the alternatives that consumer considers for purchase, of the choice of non-dominated alternatives, i.e. if the product chosen is an optimal choice or not in the set of the considered alternatives. The questions of the survey related to this analysis are introduced as follows:

Q5: *How much are you satisfied about your final choice with respect to your needs? [not much/fairly/very much]*

Q6: *Would you have preferred to book a hotel with different characteristics but you were not able to find it? [yes/no]*

Q7: *If yes to Q6, would you have preferred a hotel (more answers are feasible): (i) cheaper, (ii) with more stars, (iii) in another city zone, (iv) in other dates*

Q13: *The set of proposed hotels is: [predictable/with original and unexpected items/very surprising]*

Q23: *When you travel for holiday, what are the priority criteria with which you choose a hotel? (low priority/ medium priority/high priority): (i) price, (ii) offered services (stars), (iii) location, (iv) suited for people who I travel with (e.g., alone, partner, friends, children...), (v) other (specify)*

Q24: *When you travel for business, what are the priority criteria with which you choose a hotel? (low priority/ medium priority/high priority): (i) price, (ii) offered services (stars), (iii) location, (iv) other (specify)*

Q26: *If you imagine to travel for a holiday, are you making the hypothesis to travel: (i) alone, (ii) with the partner, (iii) with friends, (iv) with the family (children included)*

The last sub-dimension of the subjective decision quality that we analyze is the product switching, i.e. whether the consumer after a purchase decision changes his/her mind and switches to another alternative. We measure this parameter counting the number of times users, before the definitive booking page, try to change the product.
The last variable measured is the decision quality subjective, which indicates the confidence of a consumer in RS's recommendations. We study this variable with the following question of the survey:
Q12: *How much do you think proposed hotels match to your "character"? [not at all/fairly/very much]*

5.1.7 User's evaluation of RS

This construct is evaluated with the following variables:

- satisfaction
- perceived usefulness
- perceived ease of use

Satifaction The first, satisfaction, is the user's satisfaction with RS and it is measured in the survey with the following questions:
Q5: *How much are you satisfied about your final choice with respect to your needs? [not much/fairly/very much]*

Perceived usefulness is the user's perception of the utility of a RS. We study this variable with the following question of the survey:
Q29: *How much do the suggested hotels match with the search criteria? (i) not much, (ii) fairly, (iii) very much*

Ease of use is the user's perception of the effort to use the application. We measure such variables with three questions of the survey:

Q10: *The hotel selection process has been: [easy/hard/very hard]*

Q30: *Do you think that the navigation through the web application was easy and clear? [yes/no]*

Q31: *Do you think that the web application layout was intelligible? [yes/no]*

5.1.8 User-product related factors

This variable is related to the perception that users have about the product. It can be divided into two factors:

- product expertise
- perceived product risks

Product expertise This factor indicates the degree of user's knowledge about the item. We measure this degree with the following questions:

Q1: *Have you already stayed in the city the hotel is located? [yes/no]*

Q2: *Have you already stayed in the area where the hotel is located? [yes/no]*

Q3: *Have you already stayed in the selected hotel? [yes/no]*

Q19: *Average number of journeys with accommodation per year for business purpose*

Q20: *Average number of journeys with accommodation per year for holiday purpose*

Perceived product risks is defined as the user's perception of uncertainty of buying a product. We measure this variable with the following question:
Q27: How much do you think that the characteristics of the reserved hotel will correspond to the real one? (i) not much corresponding, (ii) fairly corresponding, (iii) very much corresponding

5.1.9 Product availability

The first of our extensions is integrated into the product construct. As Xiao and Benbasat do not consider at all this variable in their model, it must be analyzed appropriately with some measurements dedicated.
To accomplish these measurements we exploit both the software and the survey: with the software we are able to establish if the test is performed with bounded resources conditions (and in detail which of the three types

of unavailability) while with the survey we study the influence that this condition has on the users with the following question:
Q4: Would you have preferred to book another hotel you were aware of, but it was not available? [yes/no]

5.1.10 Task

The second extension is also measured in a dedicated way: as it determines a new construct, the task is measured in a dedicated way by software, which enables us to know which test is assigned to user. As we already said before, the test is composed by a specific task and therefore we can map this variable into the evaluation of decision making, decision outcomes, and user's evaluation of RS.

5.2 Planned execution

In this section we explain the operative modes of the experiments, specifying the instruments, the participants' characteristics, the location of the tests and the strategy of reward. Since this is a work in progress, we will describe the main characteristics of the experiments we are carrying on.

5.2.1 Instruments

In this section we provide some information about the instruments with which we execute the experiments.
All the tests are performed with our software platform, namely PoliVenus, which is located in a linux server at Politecnico di Milano university. As described in chapter 4, this framework enables the study of RS in e-tourism under multi-dimension analysis. The conceptual model which inspires us to plan and execute the experiments is our extended model derived from the Xiao and Benbasat work. Our frameworks measures a set of variables through objective and subjective metrics. The objective metrics are obtained through software, measuring some events (as click or storing the reservation etc ...), while the subjective metrics are achieved by a survey[7].

5.2.2 Participants

The participants of our tests come from three distinct groups: a group of 200 students from 2 courses of The Politecnico di Milano university; they are divided in 100 of the HCI (human computer interaction) course and

[7]The complete list of questions is provided in the appendix section

the remnants of the Enterprise Systems course. Another 10-20 people are employees of an IT company, 30 people of the national post company, and a last group 10-20 people are recruited randomly with different strategies.
To the first group belong students between 22 and 26 years old, mainly male gender. To the second group belong people who are between 26 to 45 years old. The third group is composed by people with age between 35 and 60 years old, and finally the last group is composed by people between 25 and 26 years old.
The main technique of recruiting for the tests is by e-mail.

5.2.3 Location of the experiments

The location of the experiments is not established beforehand. Since the part of the framework which performs the test is a web application on a server of Politecnico di Milano, the tests can be accomplished at every hour and in every location.

5.2.4 Reward strategy

Finally, to increase the quality of our tests we define a raffle: we extract randomly a participant who receives the price. The price is established to be the accommodation for the chosen period in the chosen hotel. We decided this type of price in order to encourage users to execute as much seriously as it is possible the task assigned. Moreover we expect users to make the test in a more realistic way.

Chapter 6

Future work

In this section we show the possible future works to expand the research in this area. In our guidelines for future works we range from software aspects, such as new features for the framework, to conceptual aspects related to the design of the logic model.

6.1 Future work on the framework

As described in previous chapters, our work focuses on the possibility to experiment RS effects under multi-variables testing environment. For this purpose we develop the framework, to enable the study of RS in the tourism domain under different empirical conditions.
From this consideration we pinpoint that there is still the need to increase the variables of analysis of the framework, in order to compare RSs effects on users.
The first issue that should be taken into account is the comparison between different styles of recommendation presentations. In our work we introduce two possibilities, i.e. a normal list of recommended hotels integrated in the application and the boxes of recommendations in the detail page of a hotel. We do believe, as other studies suggest, that the format and the content of recommendations can produce a big influence on user behavior in the interaction with the system; therefore an extension of the framework which considers different recommendation formats is very welcomed.
Another aspect to focus on is the revolution of the concept of filter. Actually all the OTA websites (and also our web application) use filters as sharp selectors of the items. In our application, the recommender list is generated respect to a filtered set of hotels. This constraint can dramatically limit the

benefit of recommender, since they have power to suggest novel of interesting items which may not respect the previous constraints imposed by the user. Therefore to overcome this problem, there should be considered the hypothesis to relax the constraints imposed by filters in order to change them from sharp characteristics to *light preferences*. With this point of view, the filters are not considered as software instruments that execute a sharp query on the set of items, but express once more an interest of the user; interest which must be considered when processing recommendations.
Of course this opens two new issues:

- how to not alter the perception of correct functioning
- how to exploit these additional preferences

The first issue is due to the fact that with sharp filters, the user perceives more control on the items provided by the system: for example if a user filters the hotels basing on their rating indicating 4 stars, he/she is sure that the result of this action is a list composed exclusively by hotels with 4 stars. It is therefore more difficult for the user to accept that the system will also propose hotels with 3 or 5 stars because "however the user can be interested in them". This unexpected result can be recognized as a bad functioning with the consequence of a decrease in expertise and trustfulness of the system. Therefore there is an absolute need in finding a way to integrate this functionality from a point of view of interaction and design with the user, letting him/her understand that the system knows the user and therefore produces value added that he/she can exploit.
The second issue is related to the way that RSs can use this added knowledge about the user. In our research we exploit a single type of signals generated by the user: a rating expressing the preference of the user respect to an item. Beyond this typology of signal, there are other two types of signals: the preferences on metadata and on the contest.
The second type of signals refers to metadata, i.e. the possibility to collect information about the preference criteria over the items. If the system understands that the user is searching hotels basing on the position of the hotel, then it can attribute more importance on this meta-information.
The third type, the contextual signals, indicates the collection of contextual information of the actual circumstance in which recommendations are provided not in relation with the user but external. For example the particular period in which the session occurs, e.g. Christmas etc... This *contextual information* can be used by the recommender to generate recommendations contextual with the circumstances in which the user is involved in.

It should be noticed that while the first and the second signals type are long lasting, the third type is temporary. Hence it is important to understand how to record and extract this information and how to insert this knowledge into the algorithms.
The last consideration we want to expose as future work is the research of another form of interaction with the user. We think that this new interaction should enable different form of analysis and item search. The user should be able to view items' features at different level of depth while searching, rather than discover them visiting the next depth level with the details. We therefore advice to address this issue with a new strategy of presenting items, inspired by a *faceted search*. By definition, the faceted search, also called faceted navigation or faceted browsing, is a technique for accessing information organized according with a faceted classification system, allowing users to explore a collection of information by applying multiple filters.
A faceted classification system classifies each information element along multiple explicit dimensions, enabling the classifications to be accessed and ordered in multiple ways rather than in a single, pre-determined, taxonomic order.
Facets correspond to properties of the information elements. They are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in a database such as author, descriptor, language, and format. Thus, existing web-pages, product descriptions or online collections of articles can be augmented with navigational facets.
Exploiting the usage of facets it becomes possible not only introduce hotels

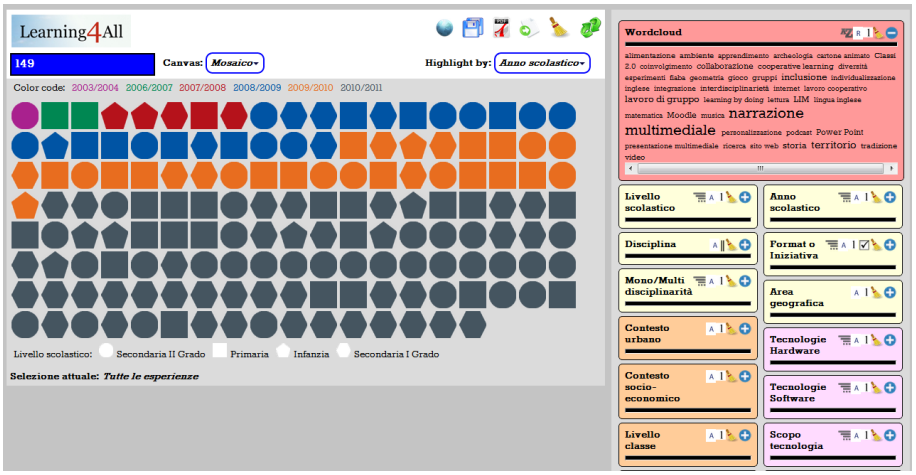

Figure 6.1: An example of an educational portal with facet technology

to users in a different way, but also produce other types of signals as expression of preferences explicitly indicated by users themselves while searching. In order to introduce in future works the faceted search, we already provide the framework with SOLR, a search engine which has the possibility to configure the documents with fields which can be further searched and returned to compose a faceted interface.
The last extension we suggest is the implementation of different elicitation techniques. In the actual framework we collect users' preferences by implicitly storing the signals users leave in their interactions with the web pages. We think that a comparison between implicit and explicit methods should be enabled. An example are the conversational RSs, where systems interacts in a human way with users and "explain" their choices and motivations for recommending items. This communication on one side let the system reach the needs and preferences of the users and on the other side increases the feeling of trust and expertise towards RS. In an optic of RSs' effects evaluation it is important to compare all these different strategies.

6.2 Future work on the conceptual model

As mentioned in other chapters, our works on the previous Xiao and Benbasat model expand the variables of analysis of RSs. Beyond our work, two considerations must be stated concerning two areas particularly relevant and complex.
The first area of future works in the model is the *trust* factor. Of course Xiao and Benbasat indicate this variable as an important component of user's evaluation of RSs and formulate postulates and research questions about it. Still we need to deepen the study on this important factor both in experimental condition as well as in the conceptual parameters related to it. Therefore we must discover under which testing conditions we are able to explore fully the trust of RSs and with what instruments we should study this factor.
A second aspect that should be considered for future works is the extension of the *bounded resource* variable. At the moment in fact we study this variable by simulating the unavailability of rooms and hotels. An extension of the work can be the evaluation of other strategies not depending only on the concept of tentative: the unavailability is obtained on the object for which the first booking tentative occurs.
Another parameter that can play an important role in the simulation of bounded resources is the *time*: the object can be considered unavailable as time passes as a consequence of resource scarcity. Hence the flow of time plays an important role and should be inserted as new parameter in the

model and in the framework as a sub condition of resource availability.
The other sub dimension of the resource availability can be the variation of price, in the sense that changing the booking conditions (e.g. period, city etc...) prices can vary over time. This consideration opens the possibility to study the items in a more dynamic way, adapting the extended model.

Bibliography

[1] D. Jannach E. Teppan M. Zanker A. Felfernig, S. Gordea. A short survey of recommendation technologies in travel and tourism. *OEGAI*, 25/2, 2007.

[2] Tuzhilin Adomavicious. Networks of design.

[3] Fishbein Martin Ajzen, Icek. Attitude-behavior relations: A theoretical analysis and review of empirical research. *American Psychological Association*, 1977.

[4] Ulrike Bauernfeind. The evaluation of a recommendation system for tourist destination decision making.

[5] Izak Benbasat and Robert W. Zmud. The identity crisis within the is discipline: Defining and communicating the discipline's core properties. *MIS Quarterly*, 27(2):pp. 183–194, 2003.

[6] Dimitrios Buhalis and Rob Law. Progress in information technology and tourism management: 20 years on and 10 years after the internet and the state of etourism research. *Tourism Management*, 29(4):609 – 623, 2008.

[7] Burke. Knowledge-based recommender systems.

[8] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002. 10.1023/A:1021240730564.

[9] Robin D. Burke. Hybrid web recommender systems. In *The Adaptive Web*, pages 377–408, 2007.

[10] Mi-Hea Cho and SooCheong (Shawn) Jang. Information value structure for vacation travel. *Journal of Travel Research*, 47(1):72–83, 2008.

[11] Lorcan Coyle and Pádraig Cunningham. Exploiting re-ranking information in a case-based personal travel assistant.

[12] Jun-Ho Jang Dev Jani and Yeong-Hyeon Hwang. Personality and tourists' internet behaviour. *EnterProceedings*.

[13] B. J. Fogg. Persuasive technology: using computers to change what we think and do. *Ubiquity*, 2002(December), December 2002.

[14] BJ Fogg. A behavior model for persuasive design. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 40:1–40:7, New York, NY, USA, 2009. ACM.

[15] Nader Mirzadeh Francesco Ricci, Bora Arslan and Adriano Venturini. Itr: A case-based travel advisory system.

[16] NADER MIRZADEH HILDEGARD RUMETSHOFER ERWIN SCHAUMLECHNER ADRIANO VENTURINI KARL W. WÖBER FRANCESCO RICCI, DANIEL R. FESENMAIER and ANDREAS H. ZINS. Dietorecs: A case-based travel advisory system.

[17] Gerald Haubl and Kyle B. Murray. Recommending or persuading: the impact of a shopping agent's algorithm on user behavior. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, EC '01, pages 163–170, New York, NY, USA, 2001. ACM.

[18] Katerina Kabassi. Personalizing recommendations for tourists. *Elsevier*.

[19] B. E. Kahn and J. Baron. An exploratory study of choice rules favored for high-stakes decisions. *Journal of Consumer Psychology*, 4(4):305–328, 1995.

[20] Maryon F. King and Siva K. Balasubramanian. The effects of expertise, end goal, and product type on adoption of preference formation strategy. *Journal of the Academy of Marketing Science*, 22(2):146–159, Spring 1994.

[21] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011. 10.1007/978-0-387-85820-3-3.

[22] Fabiana Lorenzi and Francesco Ricci. Case-based recommender systems: A unifying view. In Bamshad Mobasher and Sarabjot Anand, editors, *Intelligent Techniques for Web Personalization*, volume 3169 of *Lecture Notes in Computer Science*, pages 89–113. Springer Berlin / Heidelberg, 2005. 10.1007/11577935-5.

[23] Roberto Montanari Marcella Guagliumi Maurizio Salvoni Luca Tonelli Luca Console, Ilaria Torre. Mastrocaronte a multiagent adaptive system for tourist recommendations onboard the car, which observes the needs and tailors the helps.

[24] Eck M., Hadenfeld J., Eierman M.A., Niederman F., and Adams C. Dss theory: A model of constructs and relationships. *Decision Support Systems*, 14(1):1–26, May.

[25] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.

[26] D. Harrison McKnight and Norman L. Chervany. What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology. *Int. J. Electron. Commerce*, 6(2):35–59, December 2001.

[27] D. Harrison McKnight, Larry L. Cummings, and Norman L. Chervany. Initial trust formation in new organizational relationships. *The Academy of Management Review*, 23(3):pp. 473–490, 1998.

[28] Stuart E. Middleton, Harith Alani, and David De Roure. Exploiting synergy between ontologies and recommender systems. *CoRR*, 2002.

[29] Fabio Del Missier and Francesco Ricci. Understanding recommender systems: Experimental evaluation challenges.

[30] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. A taxonomy of recommender agents on theinternet. *Artif. Intell. Rev.*, 19(4):285–330, June 2003.

[31] Franca Garzotto Paolo Cremonesi and Roberto Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: an empirical study. *TIIS-ACM*, 2010.

[32] Eli. Pariser. *The filter bubble : what the Internet is hiding from you.* Penguin Press, New York, 2011.

[33] Rex E. Pereira. Influence of query-based decision aids on consumer decision making in electronic commerce. *Information Resources Management Journal (IRMJ)*, 2001.

[34] Pearl Pu and Li Chen. Trust building with explanation interfaces. In *IUI*, pages 93–100. ACM Press, 2006.

[35] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 157–164, New York, NY, USA, 2011. ACM.

[36] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[37] Salton. *Automatic text processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[38] Anali´a Amandi Silvia Schiaffino *. Building an expert travel agent as a software agent.

[39] Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[40] Vanitha Swaminathan. The impact of recommendation agents on consumer evaluation and choice: The moderating role of category risk, product complexity, and consumer knowledge. *Journal of Consumer Psychology*, 13(1-2):93 – 101, 2003. Consumers in Cyberspace.

[41] Nava Tintarev. Explanations of recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 203–206, New York, NY, USA, 2007. ACM.

[42] Peter Todd and Izak Benbasat. The influence of decision aids on choice strategies: An experimental analysis of the role of cognitive effort. *Organizational Behavior and Human Decision Processes*, 60(1):36 – 74, 1994.

[43] Francesco Ricci Ulrich Rabanser. Recommender systems: Do they have a viable business model in e-tourism?

[44] A. van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249 –262, 2001.

[45] Paolo Viappiani, Pearl Pu, and Boi Faltings. Conversational recommenders with adaptive suggestions. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 89–96, New York, NY, USA, 2007. ACM.

[46] Bo Xiao and Izak Benbasat. E-commerce product recommendation agents: use, characteristics, and impact. *MIS Q.*, 31(1):137–209, March 2007.

[47] Kyung-Hyan Yoo and Ulrike Gretzel. Creating more credible and persuasive recommender systems: The influence of source characteristics on recommender system evaluations. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 455–477. Springer US, 2011. 10.1007/978-0-387-85820-3-14.

[48] Wayne Zachary. A cognitively based functional taxonomy of decision support techniques. *Hum.-Comput. Interact.*, 2(1):25–63, March 1986.

Appendices

Appendix A

PoliVenus EJB - Session Beans

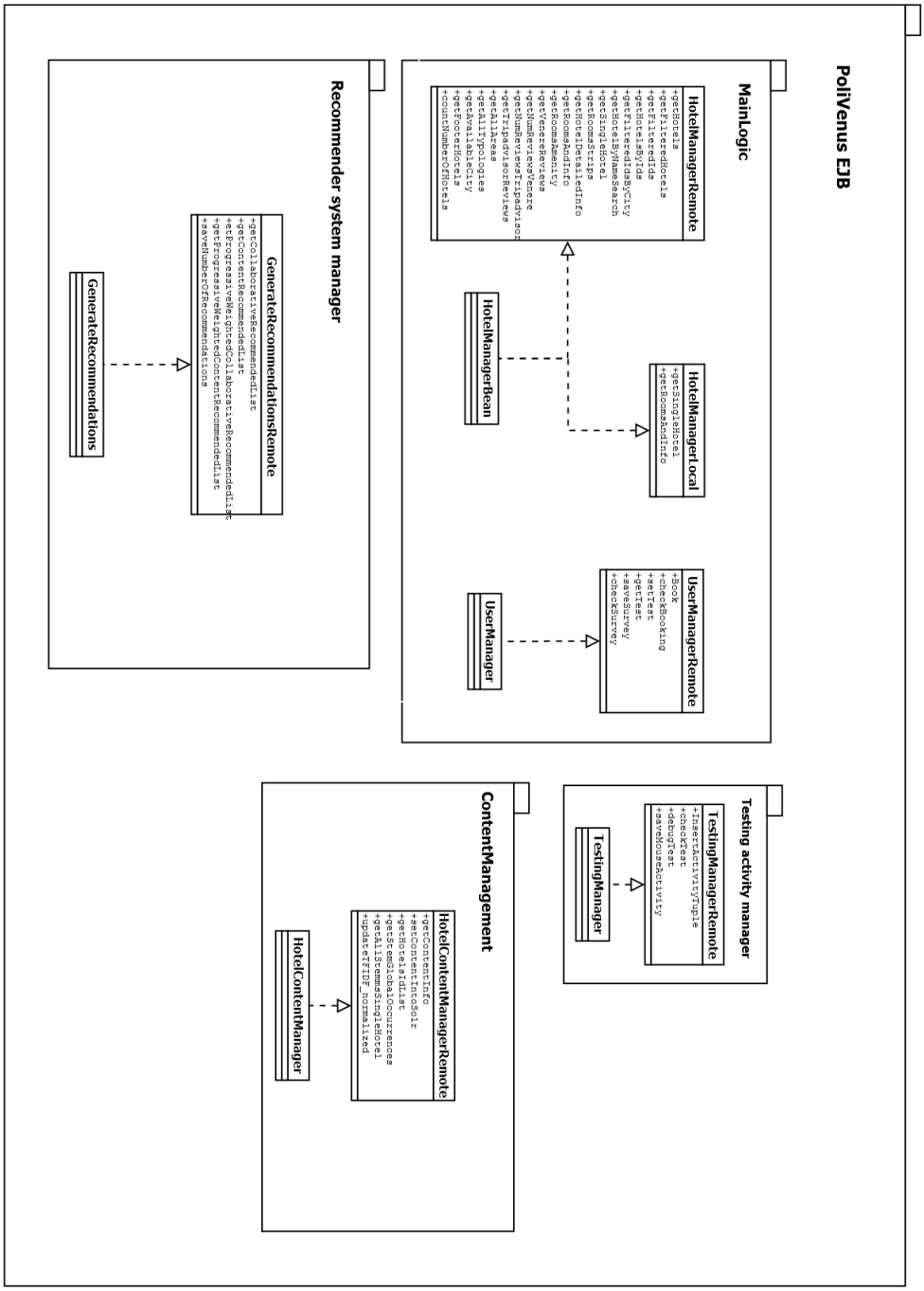

Figure A.1: PoliVenus EJB - Session Beans

Appendix B

Database ER schemes

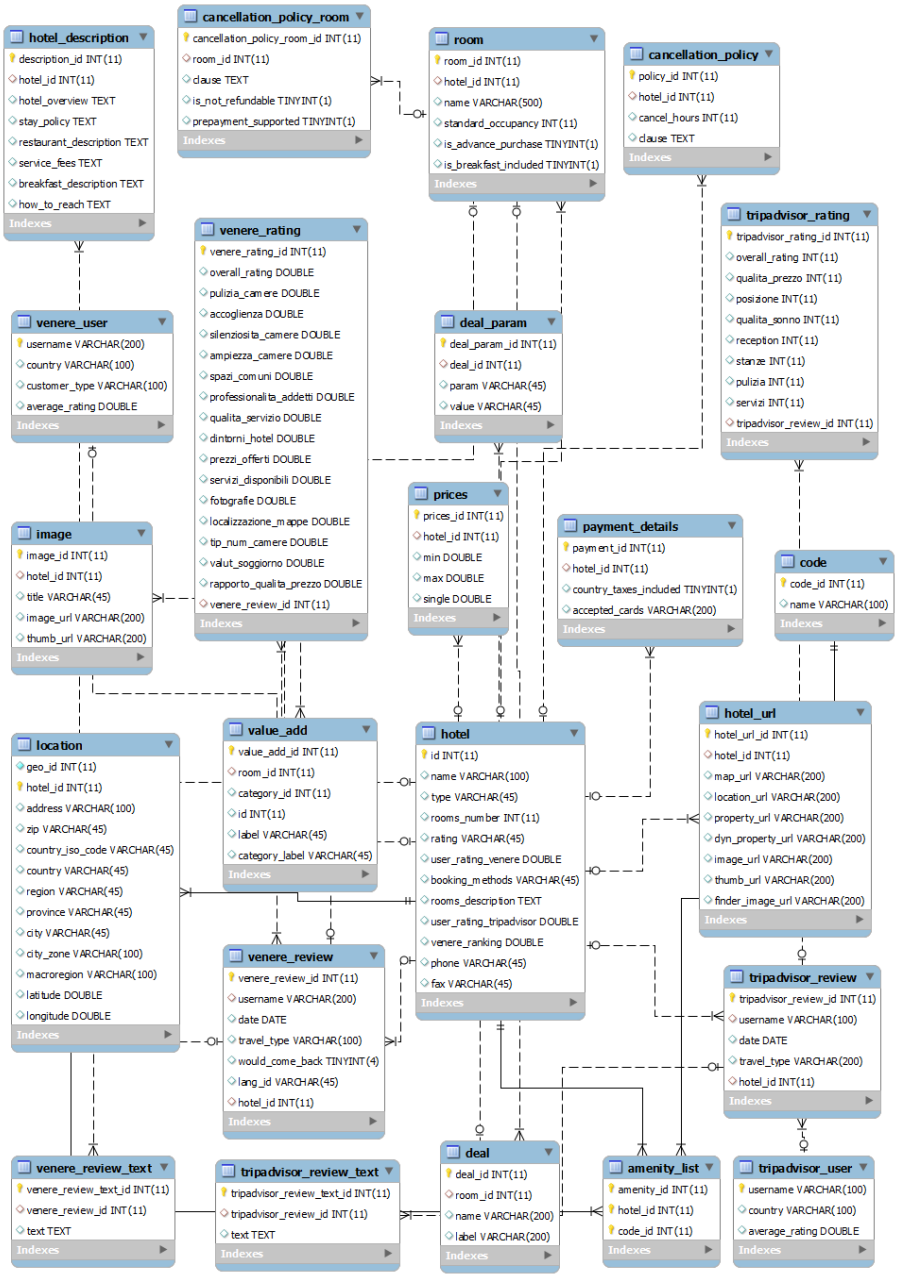

Figure B.1: Hotel catalog with reviews extracted

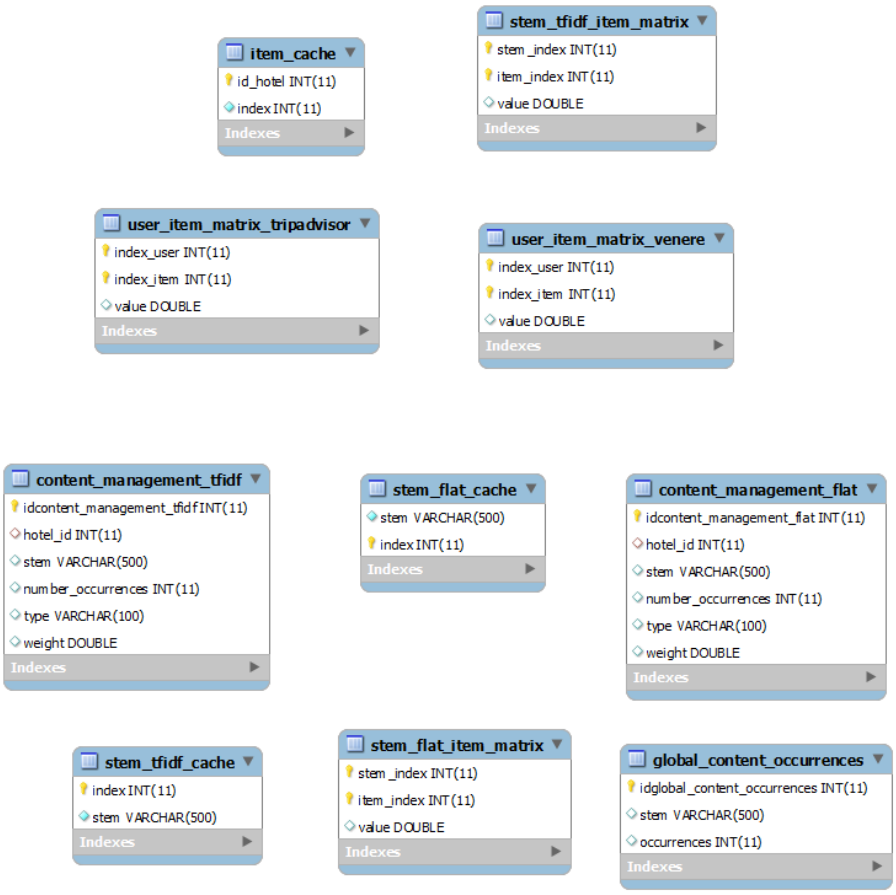

Figure B.2: Tables used from Matlab Environment

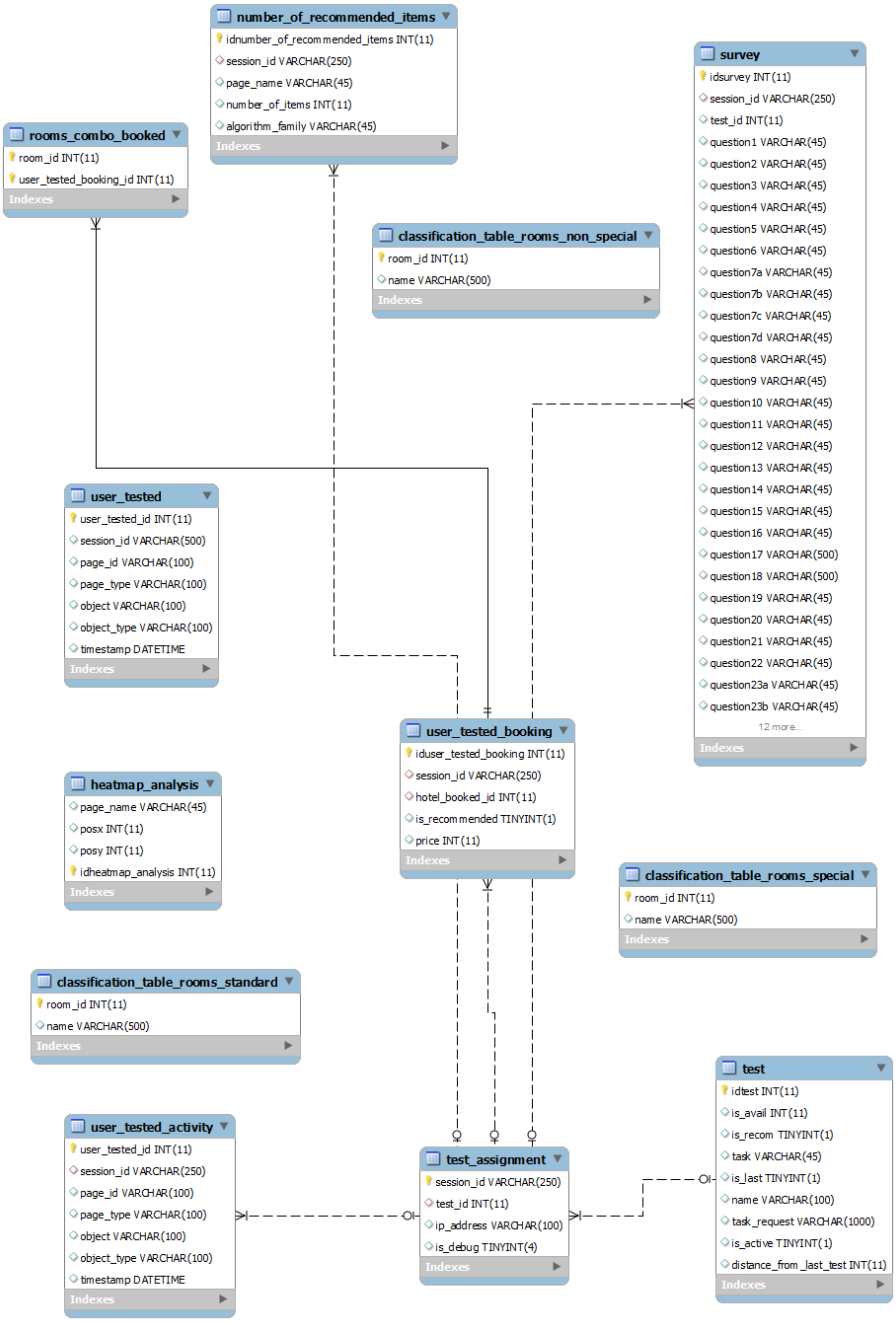

Figure B.3: Tables used to collect user activity

Appendix C

Logic model postulates

Postulate	statement
P1	RS use influences users' decision effort
P2	RS use improves users' decision quality;
P3	RS type influences users' decision effort and decision quality;
P4	The preference elicitation method influences users' decision quality and decision effort;
P6	Recommendation content influences users' product evaluation and choice;
P7	Recommendation format influences users' decision processes and decision outcomes;
P8	Product type moderates the effects of RA use on users' choice.
P9	Product complexity moderates the effects of RA use on users' decision quality and decision effort.
P10	Product complexity moderates the effect of included product attributes on users' choice.
P11	Product expertise moderates the effect of RS use on users' decision quality;
P12	Perceived product risks moderate the effects of RS use on users' decision quality and decision effort;
P13	User-RS similarity moderates the effects of RS use on users' decision quality and decision effort;
P14	RS type influences users' evaluations of RSs;
P15	The preference elicitation method influences users' perceived ease of use of and satisfaction with the RSs;
P16	The ease of generating new or additional recommendations influences users' perceived ease of use of and satisfaction with RSs;
P17	User control of interaction with RSs' preference-elicitation interface influences users' evaluation of the RSs;
P18	The provision of information about search progress, while users await results influences users' satisfaction with RSs.
P19	Response time influences users' satisfaction with RSs;
P20	Recommendation content influences users' evaluations of RSs;
P21	Recommendation format influences users' perceived usefulness of, perceived ease of use of, and satisfaction with the RSs;
P22	Product type moderates the effects of RA use on users' trust in and perceived usefulness of RSs.
P23	Product expertise moderates the effects of RS use on users' evaluations of RSs;
P24	Product expertise moderates the effects of RS type on users' evaluations of RSs;
P25	User-RS similarity moderates the effects of RS use on users' trust in, satisfaction with, and perceived usefulness of RSs;
P26	User's familiarity with RSs moderates the effects of RS use on trust in the RSs;
P27	The confirmation/disconfirmation of expectations about RSs moderates the effects of RA use on users' satisfaction with the RSs;
P28	Provider credibility, determined by the type of RS providers and the reputation of RS providers, influences users' trust in RSs;

Table C.1: Xiao and Benbasat postulates

Appendix D

Survey

Questions
Q1) Have you already stayed in the city the hotel is located? (yes/no)
Q2) Have you already stayed in the area where the hotel is located? (yes/no)
Q3) Have you already stayed in the selected hotel? (yes/no)
Q4) Would you have preferred to book another hotel you were aware of, but it was not available? (yes/no)
Q5) How much are you satisfied about your final choice with respect to your needs? (not much/fairly/very much)
Q6) Would you have preferred to book a hotel with different characteristics but you were not able to find it? (yes/no)
Q7) If yes to Q6, would you have preferred a hotel (more answers are feasible): (i) cheaper, (ii) with more stars, (iii) in another city zone, (iv) in other dates
Q8) How long (in minutes) do you think you have spent for booking the hotel?
Q9) The time required to choose the hotel is: (reasonable/overmuch/unexpectedly short)
Q10) The hotel selection process has been: (easy/hard/very hard)
Q11) Do you think the range of hotel available is: (poor/broad/very broad)
Q12) How much do you think proposed hotels match to your character? (not at all/fairly/very much)
Q13) The set of proposed hotels is: (predictable/with original and unexpected items/very surprising)
Q14-Q18) Age, Gender, Nationality, Educational qualification, Occupation
Q19) Average number of journeys with accommodation per year for business purpose
Q20) Average number of journeys with accommodation per year for holiday purpose
Q21) Did you already know Venere.com/it? (yes/no)
Q22) Have you ever used Venere.com/it to make reservations in the past? (yes/no)
Q23) When you travel for holiday, what are the priority criteria with which you choose a hotel? (low priority/ medium priority/high priority): (i) price, (ii) offered services (stars), (iii) location, (iv) suited for people who I travel with (e.g., alone, partner, friends, children...), (v) other (specify)
Q24) When you travel for business, what are the priority criteria with which you choose a hotel? (low priority/ medium priority/high priority): (i) price, (ii) offered services (stars), (iii) location, (iv) other (specify)
Q25) Where have you made this system use session? (for example at home, at the office, at university etc...)
Q26) If you imagine to travel for a holiday, are you making the hypothesis to travel: (i) alone, (ii) with the partner, (iii) with friends, (iv) with the family (children included)
Q27) How much do you think that the characteristics of the reserved hotel will correspond to the real one? (i) not much corresponding, (ii) fairly corresponding, (iii) very much corresponding
Q28) Do you think that the system, showing you the hotels, has acted in your interests?
Q29) How much do the suggested hotels match with the search criteria? (i) not much, (ii) fairly, (iii) very much
Q30) Do you think that the navigation through the web application was easy and clear? (yes/no)
Q31) Do you think that the web application layout was intelligible? (yes/no)
Q32) Space for free comments

Table D.1: Questions of the survey