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

E-TOURISM RECOMMENDER SYSTEMS

Relatore: Prof. Paolo Cremonesi
Correlatrice: Prof.ssa Franca Garzotto

Tesi di Laurea di:
Massimo Quadrana, matricola 766440

Anno Accademico 2012-2013

A mio papà Franco,
continuiamo a viaggiare assieme.

Sommario

I Sistemi di Raccomandazione (RSs) sono utilizzati in numerosi sistemi di e-commerce per aiutare gli utenti a trovare prodotti interessati in grandi cataloghi fornendo loro delle raccomandazioni personalizzate che si adattino ai loro gusti ed interessi.
Lo scopo di questa tesi è di sviluppare un RS per il dominio dell'e-tourism. Nell'e-tourism la disponibilità degli hotel dipende dalle circostanze in cui avviene la prenotazione e varia col tempo. Gli hotel "migliori" sono solitamente i primi a non essere disponibili. Mentre tradizionalmente i RSs presumono che la disponibilità dei prodotti sia potenzialmente illimitata, noi studiamo l'effetto della disponibilità variabile degli hotel sulle raccomandazioni fornite agli utenti.
Altro aspetto innovativo è l'utilizzo di un meccanismo di implicit elicitation per estrarre le preferenze del cliente per poter così fornire loro delle raccomandazioni personalizzate. Abbiamo quindi valutato la qualità delle raccomandazioni utilizzando sia esperimenti offline che online, basandoci su differenti condizioni di disponibilità degli hotel.
Abbiamo potuto osservare come fornire raccomandazioni personalizzate mantenga gli utenti più soddisfatti anche in condizioni di scarsa disponibilità di hotel. Infatti più del 70% degli utenti si è dichiarato soddisfatto delle propria esperienza quando il sistema ha fornito loro raccomandazioni personalizzate. Questo può essere di grande utilità, anche economica, per le agenzie di viaggio online.
Inoltre, occorre notare che l'e-tourism comunemente implica un maggior rischio da parte degli utenti rispetto ad altri sistemi di e-commerce. Gli utenti sono quindi più propensi a fidarsi delle recensioni di altri utenti rispetto alle descrizioni fornite dallo stesso gestore del servizio su un determinato hotel. Abbiamo quindi introdotto una tecnica per analizzare le recensioni degli utenti e riassumerne il contenuto in un singolo valore numerico. Le informazioni così estratte dalle recensioni testuali potranno essere utilizzate potenzialmente per costruire dei RSs più accurati.

Abstract

Recommender Systems (RSs) are used in many e-commerce applications to help users in discovering interesting items over huge catalogs of products by providing them personalized recommendations that fit with their interests and preferences.
The purpose of this work is to develop a RS for the e-tourism domain. In e-tourism hotel availability depends on contextual circumstances and varies over time. The hotels that are missing are often the "best" ones. Traditionally RSs assume that items are potentially always available. We studied the influence of variable item availability over the recommendation process.
Differently from many current applications in e-tourism, we used implicit elicitation to assess users' preferences and to provide them personalized recommendations. We evaluated the recommendations using both offline and online methods in different experimental conditions of item availability.
We obtained that personalized recommendations make users the most satisfied in condition of scarcity of hotels, with more than the 70% of satisfied users. This fact can be of great utility and economical impact for online travel agencies.
Moreover, e-tourism implies an higher risk for users than e-commerce applications. Hence, users tend to rely more on other users' reviews than on the descriptions given by service providers over hotels. We describe here a technique analyze user reviews and to summarize them into numerical ratings. This information can be potentially used to build more accurate RSs.

Ringraziamenti

Vorrei ringraziare in primo luogo il Prof. Paolo Cremonesi e la Prof.ssa Franca Garzotto per avermi guidato in tutte le fasi di questo lavoro. Il loro supporto professionale ed al contempo amichevole è stato importante per una mia crescita sia culturale che umana.
Vorrei ringraziare sentitamente tutti i miei amici, vicini e lontani. Sono stati anni difficili ma intensi e pieni di soddisfazioni che non posso che condividere con tutti voi. Grazie per non avermi fatto mai mancare il vostro sostegno anche quando sono rimasto a lungo distante da casa. Grazie per donarmi quei momenti di serenità, allegria e spensieratezza che donano sapore pieno alla vita.
Un grazie speciale va a Barbara, grazie per accogliere tutti le mie idee, i progetti e le incertezze, e donarmi continuamente nuove e rinnovate energie indispensabili per continuare.
Infine il più grande dei ringraziamenti va alla mia famiglia, a mia madre Antonella ed alle mie sorelle Ilaria e Sara. Grazie per essere le solide fondamenta da cui partire per costruire il mio futuro.

Massimo

Contents

List of Figures

List of Tables

Chapter 1

Introduction

I Sistemi di Raccomandazione (RSs) sono utilizzati in numerosi sistemi di e-commerce per aiutare gli utenti a trovare prodotti interessati in grandi cataloghi fornendo loro delle raccomandazioni personalizzate che si adattino ai loro gusti ed interessi.
La maggior parte dei RSs basano i loro suggerimenti sui ratings numerici dati dagli utenti ai prodotti precedentemente acquistati. I RSs funzionano bene in molti domini dell'e-commerce, quali i mercati di film e libri.

Lo scopo di questa tesi è quello di sviluppare un RSs per il dominio dell'e-tourism, in collaborazione con Venere.com, una delle imprese leader nel settore dell'e-tourism e parte del Gruppo Expedia.
Tradizionalmente i RSs assumono che i prodotti siano potenzialmente sempre disponibili; l'unica variabilità considerata è l'aggiunta di nuovi prodotti (items) al catalogo offerto. Contrariamente, nell'e-tourism gli hotel non sono sempre disponibili; il catalogo degli hotel disponibili può ridursi a causa di molte circostanze contestuali, come accade ad esempio nei periodi di alta stagione. Nella letteratura sono stati studiati problemi simili ma intrinsecamente differenti: le deviazioni introdotte dalla popolarità dei prodotti, la distribuzione non casuale dei ratings mancanti e il problema dei nuovi items.
I RSs nel dominio dell'e-tourism fanno uso di explicit elicitation perchè semplifica l'acquisizione delle preferenze dell'utente al fine di fornire raccomandazioni personalizzate. Ciò nonostante, i meccanismi di explicit elicitation spesso richiedono la registrazione dell'utente presso il sistema e il tracciamento delle sue attività attraverso le varie sessioni di lavoro. Questo è spesso poco desiderabile, visto che gli utenti sono riluttanti nel fornire informazioni personali al sistema.
Inoltre, nel dominio dell'e-tourism gli utenti considerano seriamente le recensioni degli altri utenti prima di effettuare la propria prenotazione. La

prenotazione di hotel implica rischi maggiori per l'utente rispetto a molte altre attività di e-commerce; per questo gli utenti tendono a fidarsi maggiormente delle opinioni espresse da altri utenti piuttosto che della descrizione dell'hotel data dal fornitore del servizio.

I contributi innovativi di questa tesi sono: (i) lo studio del comportamento dei RSs in condizioni di limitata disponibilità di hotel dovuta a fattori temporali o contestuali; (ii) l'uso di implicit elicitation in una singola sessione utente; (iii) lo studio di una metodologia per riassumere le recensioni degli utenti in un singolo valore numerico.
Per lo svolgimento di questo lavoro abbiamo largamente impiegato PoliVenus [20], un framework sviluppato dal Politecnico di Milano in collaborazione con Venere.com.
Utilizzando le informazioni raccolte in uno user study precedente, abbiamo identificato l'algoritmo di raccomandazione personalizzato più adatto allo scopo ed abbiamo progettato il sistema di implicit elicitation da utilizzare per raccomandare hotels agli utenti. Abbiamo analizzato in dettaglio il processo di fruizione degli hotels in modo da poter simulare la disponibilità limitata di hotels in alta stagione. Abbiamo quindi confrontato gli effetti di raccomandazioni personalizzate e non personalizzate sulla soddisfazione degli utenti, utilizzando una valutazione sia offline che online. Infine abbiamo implementato un sistema in grado di predire i ratings dati dagli utenti a partire dall'opinione da loro espressa nelle recensioni testuali.

Abbiamo potuto osservare come la disponibilità degli items abbia un impatto notevole sulle prestazioni degli algoritmi non personalizzati. La percentuale degli utenti soddisfatti cala del 50% quando gli hotel raccomandati sono tra i più semplici, come ad esempio quelli più popolari.
Quando invece le raccomandazioni sono personalizzate per gli utenti, la loro soddisfazione si mantiene costantemente sopra il 70% in entrambe le condizioni sperimentali di disponibilità completa e limitata. Questo può essere di grande aiuto per i fornitori dei servizi al fine di offrire un buon servizio ai propri clienti anche nelle situazioni dove questo è più complesso e difficile, con un conseguente grande impatto economico.
Abbiamo inoltre introdotto un nuovo sistema per riassumere le opinioni espresse nelle recensioni. Questa è una fonte di informazioni potenzialmente molto utile per poter sviluppare nuovi RSs che siano consapevoli delle opinioni espressi dagli utenti sulle caratteristiche dei prodotti per mezzo di recensioni testuali.

A partire da questo studio e dai sui risultati è stato pubblicato il paper *Evaluating Top-N Recommendations "When the Best are Gone"* (Cremonesi, Garzotto, Quadrana). Il paper è stato accettato per la presentazione

come short paper alla conferenza ACM RecSys 2013, la più importante conferenza mondiale sui Sistemi di Raccomandazione.

1.1 Struttura della Tesi

La struttura della tesi è la seguente:
Nel Capitolo 2 viene descritto lo stato dell'arte della ricerca sui Sistemi di Raccomandazione e sull'Analisi dell'Opinione nei documenti testuali. La prima parte riguarda le caratteristiche dei RSs e la loro tassonomia. Nella seconda parte presentiamo l'analisi offline ed online dei RSs. Sono quindi presentati tre problemi simili ma sostanzialmente differenti dal quello da noi studiato, ossia la deviazione dovuta alla popolarità, la distribuzione non casuale dei ratings mancanti ed il problema del nuovo item.
Nella terza parte sono presentati in dettagli i metodi di opinion mining e summarization. In particolare analizziamo il concetto di feature-based opinion mining, una sottoclasse dell'analisi dell'opinione che è particolarmente adeguata allo scopo del nostro lavoro. Per concludere è presentata una panoramica sul framework PoliVenus e sulle sue funzionalità principali.
Nel Capitolo 3 viene presentato il sistema di implicit elicitation che abbiamo adottato per ottenere le preferenze delle utente. Siamo partiti da una tecnica grezza di implicit elicitation che abbiamo successivamente raffinato per ottenere l'insieme finale di componenti e pesi da utilizzare nell'elicitation.
Nel Capitolo 4 viene presentato lo studio sulla disponibilità limitata di hotel nel dominio dell'e-tourism. Viene presentato prima lo studio generico sulla disponibilità limitata degli items, quindi la sua effettiva applicazione nelle simulazioni che seguono.
Nel Capitolo 5 viene presentata la sperimentazione che abbiamo condotto al fine di valutare gli effetti della disponibilità limitata di items sui RSs. Abbiamo effettuato un'analisi sia offline che online sotto la condizione di disponibilità limitata. Vengono quindi presentati i risultati ottenuti da queste analisi.
Nel Capitolo 6 viene presentato lo studio sull'analisi delle opinioni contenute nelle recensioni utenti nel dominio dell'e-tourism. Viene presentata la metodologia che abbiamo adottato per analizzare le recensioni testuali, e viene presentato il modello lineare che suggeriamo per riassumere le recensioni in PoliVenus.
Nel Capitolo 7 sono presentate le conclusioni della tesi ed i possibili sviluppi futuri.
Nell'appendice A vengono riportati un presentazione dell'esperienza utente in PoliVenus con l'uso di implicit elicitation, il questionario finale dell'esperimento

con gli utenti, l'algoritmo di feature-based opinion mining che abbiamo utilizzato e i risultati dettagliati dell'analisi dell'opinione.

Introduction

Recommender Systems (RSs) help users in discovering interesting items over huge catalogs of products by providing them personalized recommendations that fit with their interests and preferences.
Most RSs make their suggestion based on the numerical ratings that users gave to previously purchased products. RSs are known to perform well in many e-commerce domains, such as the movie and the book markets.

The purpose of this work is to develop a RS for the e-tourism domain, in cooperation with Venere.com, one of the leading companies in the e-tourism sector and part of the Expedia Group.
Traditionally RSs assume that items are potentially always available and the only variability over the catalog of products that is considered is the addition of new items. Conversely, in e-tourism hotels are not always available and the catalog can be shrunk by many contextual circumstances, for example in high season periods. In the literature are available some studies over similar but indeed very different problems: the popularity bias, the non-random distribution of missing ratings and the new item problem.
Previous works on RSs for the e-tourism domain used explicit elicitation because it makes easier to infer users' preferences necessary for building recommendations. However, explicit elicitation often requires user registration and cross-session tracking, that are not always desirable because users are reluctant to provide personal information to the system.
Moreover, in e-tourism users take seriously in account hotel reviews before making their reservation. Hotel booking is an activity that implies a higher risk with respect to other e-commerce activities, so users tend to trust more on other users' opinions than to the hotel description given by the service provider.

The innovative contributions of our work are: (i) the study the behavior of RSs in case when the availability of items is constrained by temporal or contextual factors; (ii) the extensive use of implicit elicitation in a single user session; (iii) the study of a technique to summarize user reviews into

numerical ratings.
To the purposes of our work we extensively used PoliVenus [20], a framework developed by the Politecnico di Milano in cooperation with Venere.com.
Starting from a previous user study, we identified the most appropriate personalized recommendation algorithm and we designed the implicit elicitation method to be used to recommend hotels to users. We explored the item consumption process in order to define the technique to simulate the bounded availability of hotels in high season. We then compared the effects on users' satisfaction of personalized recommendations against non-personalized recommendations, using both offline and online evaluation. At the end, we implemented a system to predict user ratings starting from the opinions expressed by users in their reviews.

We observed that items availability has a great impact over the performances of non-personalized algorithms. The percentage of satisfied users drops of the 50% when the most trivial recommendations are provided to users, such as the most popular hotels.
On the other hand, the personalized algorithm is able to keep more than the 70% of satisfied users in both the experimental conditions, i.e., fully availability and limited availability. This can be of great utility for service providers to offer a good service to their clients also when the contingent circumstances makes it very harder, with a consequent great economical impact.
We also introduced a novel approach to summarize opinions in user reviews. This information can be a really powerful base of knowledge to develop new RSs that are aware of the opinion expressed that users express in textual reviews over product features.

The study and its results were published in the paper *Evaluating Top-N Recommendations "When the Best are Gone"* (Cremonesi, Garzotto, Quadrana). It has been accepted for short paper presentation at ACM RecSys 2013 Conference, the most important conference over recommender systems worldwide.

1.2 Structure of the Work

The structure of the work is the following:
In Chapter 2 we describe the state of the art of the research in Recommender Systems and Opinion Analysis. In the first part we cover in the detail the characteristics of RSs and their taxonomy. In the second part we analyze offline and online evaluation of RSs. We present three problems that are similar but yet different to our, namely the popularity bias, the non-random

missing rating distribution and the new item problem.
In the third part we present in detail the opinion mining and summarization tasks. We focus our attention on feature-based opinion mining, a subclass of opinion mining techniques that is especially suitable to the purposes of our study. At the end, we present an overview on the framework PoliVenus and on its core functionalities.
In Chapter 3 we present the implicit elicitation mechanism we adopted to infer user's preferences. We started with a rough elicitation technique that we subsequently refined to obtain our final elicitation components and weights.
In Chapter 4 we present our study over bounded availability of hotels in e-tourism. We first present our study over the bounded condition and how we simulate it in the subsequent experiments.
In Section 5 we present the experimentation over the influence of bounded availability on RSs. We performed both offline and online evaluation under the bounded condition. We finally present the results we obtained from each analysis.
In Chapeter 6 we present our study over the opinion analysis of user reviews in the hotel domain. We explain the methodology we adopted in review processing, and we present the linear weighting model we suggest for summarizing reviews in PoliVenus.
In Chapter 7 we present the conclusions we derived from the study and the future works.
In Appendix A we report an overview over PoliVenus user experience with implicit elicitation, final survey of the user experiment, the feature-based opinion mining algorithm we adopted and some detailed results of the opinion mining analysis.

Chapter 2

State of the art

In this chapter we present the state of the art knowledge in Recommender Systems and Opinion Summarization.

In the first part we explore the nature of Recommender Systems (RSs). We present the most recent and relevant works in algorithms and evaluation methods for RSs. We focus our attention on their application in the e-tourism domain. We then describe in detail PoliVenus, the system that have been extensively used in our study.

In the second part we present the state of art knowledge in Opinion Mining and Summarization. With the explosion of World Wide Web and Social Networks, a huge amount of opinionated text have become available. E-tourism is no exception. Among the several techniques have been proposed by researchers so far, we used feature-based opinion mining to precess hotel reviews, because its results are more suitable to be integrated with Recommender Systems.

2.1 Recommender Systems

The explosion of e-commerce applications induced industry and academia to develop new RSs, in order to reduce the burden on users in finding new interesting products. Example of applications include recommending books and other products by Amazon.com [35], and movies by Movielens [46]. In these applications users have the capability to rate the product they have purchased. In this way, they provide to other users and to the system their opinion on the product they have just purchased.

In its most common formulation, the recommendation problem consists in estimating the rating for the items unseen by the user. Once we the

ratings of unseen items have been estimated, we can recommend to the user the items with highest ratings.

The recommendation problem can be formulated as follows [1]:
Let C be the set of all users and let S be the set of all possible items that can be recommended. The set of possible items S and users C can be very large, even millions of items in some applications. Let u be a utility function that measures the usefulness of item s to user c, $u : C \times S \to R$. R is a totally ordered set, because it contains only nonnegative integers or real numbers within a certain range. Then, for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes user's utility:

$$\forall c \in C, s'_c = \underset{s \in S}{\operatorname{argmax}}\, u(c, s) \tag{2.1}$$

Usually the utility of an item is represented by its rating. However, in general it can be any arbitrary function. The central problem of RSs lies in that utility u is not usually defined in the whole $C \times S$ space, but only on one subset of it. So u needs to be extrapolated to the whole space. In RSs, where utility is usually represented by ratings, the initial $C \times S$ is represented by an User-item Rating Matrix (URM).

Several kinds of RSs have been proposed by researchers in the last decade. They can differ in many characteristics, such as the method they adopt to estimate ratings, in the utility function and in the type of recommendations.
RSs can predict the *absolute* values of ratings that individual users would give to the yet unseen items, or they can predict the *relative* preferences of users (preference-based filtering); in the latter case, RSs will try to predict the correct relative order of items, rather than their individual ratings.
According to the type of recommendations, to RSs can be all classified into two main categories:

- non-personalized RSs;
- personalized RSs.

Non-personalized RSs essentially do not take in account user's model to provide recommendations; thus, non-personalized RSs recommend the same items for every user. For example, they can present to users unseen items sorted out by popularity (e.g., the *TopPop* algorithm) or by average rating (e.g., the *MovieAvg* algorithm).

On the other hand, personalized RSs build a specific user's model to provide personalized recommendations. Such RSs can be classified, according on how recommendations are made, in three main categories [1]:

Content-based recommendations, which provide the user items similar to the ones the user preferred in the past;

Collaborative recommendations, also known as social filtering, which provide the user items that people with similar tastes liked in the past;

Hybrid approaches, which combine the above methods to provide more accurate recommendations.

There exist other types of RSs, such as knowledge-based RSs [8, 9, 45], case-based RSs [39, 59] and context based RSs [34]. Each type of RSs tries to limit the drawbacks of previous RSs by putting additional knowledge in the recommendation process, e.g, the relationships among needs and preferences of users [9]. Still, content-based, collaborative and hybrid approaches are the most used in practice and the last RSs can be considered as special cases of these three main categories to some extent.

2.1.1 Content-based Methods

Content-based recommenders provides recommendations for categories the user liked in the past, by matching up the attributes of a user profile in which preferences are stored, with the attributes of an item, in order to recommend to the user new interesting items [39].
Frequently content-based RSs focus on recommending items containing textual information, such as description and reviews. Thus several information retrieval techniques can be used to build user profiles from such textual data. Typically this process involves the collection keywords in bag-of-words (BOW), where are considered textual words along with their own frequency information, discarding any grammar/semantic connection [5]. Usually words are preprocessed using tokenization, stopword filtering and stemming [11]. The importance of a word k_i in a document d_i is determined with some weighting measure $w_{i,j}$. One of the best-known weighting measures in Information Retrieval is *term frequency/inverse document frequency (TF-IDF)* measure, 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 [61].
The TF-IDF weight of a word is therefore computed as:

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.2}$$

where, assuming that $f_{i,j}$ is the number of times keyword k_i appears in document d_j,

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{2.3}$$

and

$$IDF_i = \log \frac{N}{n_i} \tag{2.4}$$

Once items are represented in term of weighted BOW, each user can be represented as a vector of weights $(w_{c1}, \ldots, w_{ck})$, where each weight denotes the importance of a keyword k_i to user c. These weights can be computed from user's ratings with several techniques, such as averaging, Bayesian classifiers, etc. [1]. Recommendations can be computed using the vector similarity between the user profile vector $\vec{\mathbf{w}}_c$ and item vector $\vec{\mathbf{w}}_s$, using similarity measures such as the cosine similarity

$$u(c, s) = \cos(\vec{\mathbf{w}}_c, \vec{\mathbf{w}}_s) = \frac{\vec{\mathbf{w}}_c \cdot \vec{\mathbf{w}}_s}{\|\vec{\mathbf{w}}_c\|_2 \times \|\vec{\mathbf{w}}_s\|_2} \tag{2.5}$$

For example, let us consider some keywords that are used by content-based algorithms. In Table 2.1 are shown some of the keywords extracted from hotel descriptions and reviews in the hotel booking domain, with their respective TF-IDF scores.
If the user c has previously liked hotels associated to k_i keywords (e.g., station, restaurant, bus), such keywords will be represented with high weights w_{ci} into the user profile $\vec{\mathbf{w}}_c$. Consequently, the content-based RSs, using its own similarity measure, would assign a high utility $u(c, s)$ to hotels s that have high weighted such k_i keywords in their own item vectors $\vec{\mathbf{w}}_s$.
In general, content-based recommender systems use item attributes to recommend items "similar" to items the user liked in the past [1].
Among the several content-based algorithm available in literature, we mention LSA [5] and DirectContent [14], a simplified version of LSA.

Limitations of Content-based RSs

Two are the main issues in content-based RSs: *overspecialization* and the so called *new user problem.*
Overspecialization regards the attitude of content-based RSs in recommending items that are too similar to user preferences; this can result in recommendations to the user that are obvious and too homogeneous, while *diversity* is really a desirable feature in recommender systems.
Content-based RSs also needs users to rate a sufficient number of items in order to understand users' preferences. Therefore, they can hardly provide useful recommendations to new users.

Stem	TF-IDF
luxury	0.2250
restaurant	0.0568
directly	0.0387
satisfy	0.0264
east	0.0261
bus	0.0248
time	0.0240
meeting	0.0236
opening	0.0231
foot	0.0215
offer	0.0205
find	0.0191
station	0.0181

Table 2.1: Example of stem TF-IDF matrix in hotel booking domain.

2.1.2 Collaborative Methods

Collaborative RSs try to predict the utility of items for a particular user based on the items previously rated by other users [1]. They ignore the content and try to exploit the "social" aspect of communities of users.
They are based on these two assumptions:

- there are groups of users with similar tastes, which rate the items similarly;
- correlated items are rated by a group of users similarly.

Collaborative algorithms can be subdivided in the following two categories:

- Neighborhood models (user-based and item-based models)
- Matrix factorization models

Neighborhood models try to exploit the similarity relationship between users (or items) to estimate the utilities, namely the missing ratings in the user-item rating matrix.
More formally, the utility $u(c, s)$ of an item s for a user c can be estimated based on the utilities $u(c_j, s)$ assigned to item s by those users $c_j \in C$ who are similar to user c; this approach is also called *user-based*, since it is based on between-users similarity.
Another possible approach is to estimate utility $u(c, s)$ based on utilities $u(c, s_i)$ assigned by user c to items $s_i \in S$ who are similar to item s. This approach is called *item-based* and it is based on between-items similarity.

Notice that this concept of similarity is completely different from the one used content-based RSs. Since no content information is used, here we are saying that whatever the content of an item is, such item is considered somehow similar to another because the community has expressed the same evaluation for both items [5].
Neighborhood models predict the missing ratings by averaging the rating of the k-Nearest-Neighbors of the current user (or item).
In practice, user-based collaborative filtering is not used due to the poor quality of results and their computational requirements. Item-based is generally more scalable, since the number of items is usually lower than the number of items, and it makes easier to explain the reason of the recommendation to the user in terms of items previously rated by him/her.

Another approach to collaborative RSs is *matrix factorization*, or *latent factor*. Instead of studying users and items in the space of ratings, they are represented as vectors in a common low-dimensional "latent factor" space. In such space, users and items are directly comparable and the rating of a user c on an item s can be estimated as the proximity (e.g., the inner-product) between the respective latent factor vectors.
These methods are also called SVD models, because they rely on Singular Value Decomposition (SVD) to move users and items from their original space to the reduced "latent factor" space. An User-item Rating Matrix $\boldsymbol{R}$ of dimensions $n \times m$ can be factorized as:

$$\hat{\boldsymbol{R}} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^T \tag{2.6}$$

where $\boldsymbol{U}$ is a $n \times l$ orthonormal, $\boldsymbol{V}$ is a $m \times l$ orthonormal matrix and $\boldsymbol{S}$ is a $l \times l$ diagonal matrix containing the first l singular values, sorted in decreasing order. The number l of singular values corresponds to the number of *latent factors* to be considered into the model. The rating of a user c about item s can be predicted as:

$$\hat{r}_{cs} = \boldsymbol{u}_c \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{v}_s^T \tag{2.7}$$

where $\boldsymbol{u}_c$ represents the c-row of $\boldsymbol{U}$ and $\boldsymbol{v}_s$ the s-row of $\boldsymbol{V}$. Since $\boldsymbol{U}$ and $\boldsymbol{V}$ have orthonormal columns, by multiplying both terms of (2.6) by $\boldsymbol{V}$, we can state that:

$$\boldsymbol{u}_c \cdot \boldsymbol{\Sigma} = \boldsymbol{r}_c \cdot \boldsymbol{V} \tag{2.8}$$

where $\boldsymbol{r}_c$ is the c-row of $\boldsymbol{R}$ (i.e., the profile vector of user c).
Consequently, (2.7) can be reformulated as:

$$\hat{r}_{cs} = \boldsymbol{r}_c \cdot \boldsymbol{V} \cdot \boldsymbol{v}_s^T \tag{2.9}$$

In order to predict all the ratings of the user c, we can extend the previous equation as:

$$\hat{r}_c = \boldsymbol{r}_c \cdot \boldsymbol{V} \cdot \boldsymbol{V}^T \tag{2.10}$$

which depends on the profile vector $\boldsymbol{r}_c$, since the product between $\boldsymbol{V}$ and $\boldsymbol{V}^T$ can be precomputed when training the model.
SVD factorization allows the collaborative RSs to provide recommendations to any user, even if his profile $\boldsymbol{r}_c$ is new or it has been updated since the model has been created.
It allows to represent users into a low-dimensional space of size $l \times l$, much less than the $n \times m$ original space. Moreover, it reduces the noise of data, because it allows to discard low-magnitude singular data that are usually associated to least-informative data, typically noisy [23].
Finally, it strengthens the relationship among data, because related vectors are represented closer in the l-dimensional space that in the original space. It also allows to discover hidden dependencies among users or items.
Among the many SVD-based algorithms that are currently available we cite PureSVD, which bases its estimation process on the conventional SVD factorization, by treating missing items as zeros. It achieves very good performances in recommendations, often better neighborhood based collaborative RSs in many conditions [13, 15].

Limitations of Collaborative RSs

Collaborative RSs limit the problem in diversity in recommendations of content-based ones. They do not rely on any kind of content information, but only on other-users' preferences; thus, they can deal with any content and can recommend any items, since there is no need of the content of recommended items to correspond to any item the user previously liked.

On the other hand, item-based collaborative RSs are affected by the *new item* problem. Since such systems recommend the items most correlated to those preferred by the user, a new item item cannot be recommended because nobody has rated it so far.

They can be also limited by the *sparsity* of user-item rating matrix. Usually the number of rating in such matrix is very small compared to the number of ratings to estimate; this can prevent the recommendations to users with unique tastes, because there will not be enough users (or items) enough similar to him.

As a consequence of the previous limitations, collaborative RSs are seriously affected by the so called *cold start* problem. In fact, due to the

scarcity of user ratings, any brand new systems will not be able to provide any accurate recommendations. Content-based systems are also affected by such problem, but they can mitigate its effects thanks to their content-aided approach [5].

2.1.3 Hybrid Methods

We have analyzed before the most important characteristics of content-based and collaborative RSs. We have seen that every RSs has its own drawbacks and limitations. We can overcome such limitations by combining the results of different RSs. For instance, collaborative RSs that suffers of cold start problem can be aided by content-based ones in providing recommendations in the early stages.
These types of RSs are called *hybrid Recommender Systems*, because they merge the results of different RSs to create a new "hybrid" recommender. According to the different strategy adopted in the hybridization process, they can be classified into:

- Parallelized hybrids
- Pipelined hybrids
- Monolithic hybrids

In *parallelized hybrids* (Figure 2.1) different RSs work independently and produce separate recommendation lists. Their results are then combined to produce the final recommendations to be provided to the user.
Different strategies can be adopted in combining the results of different recommenders:

Weighted: the results of all RSs available in the system are combined to predict the score of an item by, e.g., performing a linear combination of the recommended scores.

Switched: the different recommendation techniques are switched according to a switching criteria; e.g., a content-based RSs can be used first, and then the system switches to a collaborative RSs when the content-based RSs fails to predict ratings over a fixed confidence threshold.

Mixed: the results of more than one recommendation technique are combined and are presented together to the user. One example of combination is *interleaved hybridization*, which interleaves the results of two different recommenders, such as a content-based and a collaborative RSs.

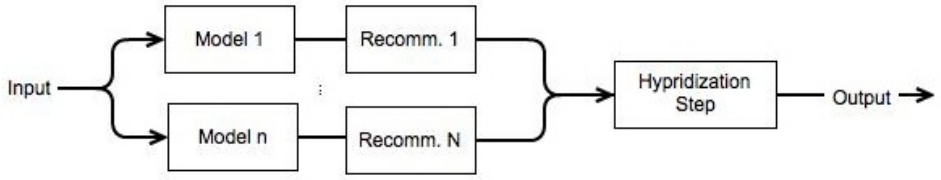


Figure 2.1: Parallel hybridization.

Pipelined hybrids use one RS as the input for a second recommender system. The second RS can be used to refine the results of the first, or the model of the first can be used as input for the second RS.
Monolithic hybrids merge the models of more RSs to create a new "augmented model" to be used in the final recommendation stage.
A detailed survey on hybrid recommender systems, with a more comprehensive taxonomy, can be found in [9].
Mixed interleaved hybrid RSs are known to behave well in the hotel domain, so we will adopt them in our work [14].

2.2 Recommender Systems Evaluation

Recommender Systems are especially used in domains where it is necessary to facilitate users in finding interesting items among large amounts of digital contents. An excessively wide offer would burden users in searching the contents they needs or they are interested into. Thus, recommendations can potentially improve the decision making process by reducing the information overload and, in general, by helping users to choose among a vast set of alternatives.

As we have already seen, most RSs operate estimating the utility the user would give to an unseen item, using a statistical model derived from content information about items or from the peer knowledge available in the community.
From now on we will consider utility as the rating. Consequently, the utility estimation corresponds to the prediction of the rating that the user would give to an unseen item.

Whatever is the type of RS, it is necessary to evaluate the effectiveness of the recommendation process, and to investigate on its influence on the users' decision making process. The quality of a RS can be defined either in terms of *system-centric* or *user-centric* evaluation.

We discuss here over the best known practices in RSs evaluation. In particular, we consider the so-called Top-N Recommendation task.

2.2.1 System-centric Evaluation

In system-centric evaluation, usually called *offline evaluation*, the RS is evaluated against a pre-built ground truth dataset. Users do not interact with the system during the evaluation, but it is based on the comparison between estimated users' ratings and the ratings previously collected from real users on the same items [14].

The original dataset is subdivided into a *training set* and a *test set*. The former is used to compute the statistical model used by the RS. The latter is used to assess the quality of the previously computed model in terms of *accuracy* metrics and *error* metrics. The estimation of such metrics can be performed with several methods, such as *bootstrapping*, *k-fold cross-validation* or *leave-one-out cross-validation*. Still, *10-fold cross-validation* is widely considered as the best evaluation technique [31].

Error metrics can be computed when the RS estimates the exact rating of an item for a user. The most commonly used metrics are the Root Mean Squared Error (RMSE),

$$RMSE = \sqrt{\frac{\sum_{j,i}(u(c_j, s_i) - \tilde{u}(c_j, s_i))^2}{\|testset\|}} \tag{2.11}$$

and the Mean Absolute Error (MAE),

$$MAE = \frac{\sum_{j,i} |u(c_j, s_i) - \tilde{u}(c_j, s_i)|}{\|testset\|} \tag{2.12}$$

where $u(c_j, s_i)$ and $\tilde{u}(c_j, s_i)$ are respectively the real and estimated ratings of item s_i for user c_j.

Often we are more interested in the *ranking* capabilities of RSs rather than in their precision in estimating ratings. In fact, after a RS has ranked the items according to their estimated ratings, it selects a subset, namely the "best bet" subset, to be recommended to the user; this is indeed a classification process, which divides items into *relevant* and *non-relevant* ones, and the evaluation of classification quality can be more interesting and informative then the mere precision of the system.
Hence, we can apply here the classical accuracy measures used in classification: the *recall*, i.e., the conditional probability of suggesting an item that is relevant for the user; the *fallout*, i.e., the conditional probability of suggesting an item that is irrelevant for the user.
These metrics are commonly evaluated considering the top N items with largest predicted ratings, the so-called *Top-N Recommendation task* [18].

For each recommendation we have an *hit* if the relevant item is in the list. Therefore, recall can be computed as

$$recall(N) = \frac{\#\{\text{relevant items in the list}\}}{\#\{\text{recommendations}\}} \tag{2.13}$$

Similarly, fallout is measured counting the number of non-relevant items int the recommended list. Fallout can be computed as

$$fallout(N) = \frac{\#\{\text{non-relevant items in the list}\}}{\#\{\text{recommendations}\}} \tag{2.14}$$

In general, small error metrics means good capabilities of the RS in predicting missing ratings. However, very good performances in prediction may not correspond to satisfying recommendations to users. Accuracy metrics helps to evaluate the quality of the items recommended to users, and many times are more helpful than error metrics in assessing the real quality of a RS. Moreover, error metrics cannot used with recommenders that exclusively produce a ranking of items, e.g., mixed hybrid RSs.

2.2.2 User-centric Evaluation

In user-centric evaluation, usually called *online evaluation*, users interact with a running recommender and receive recommendations. The evaluation process collects both *subjective* and *objective* measures.
Subjective measures are collected by asking the user (e.g., through interviews or surveys), and regards subjective aspects of the interaction with the system, such as choice satisfaction, choice risk, perceived time, etc.
Objective measures are instead automatically collected by the system, by recording user's interactions with the system and then by analysing system logs. Examples of objective measures are the elapsed time, the menu interactions, the number of explored pages, etc.
In a word, user-centric evaluation focuses on the human-computer interaction process, or User eXperience (UX).
The research in this field is still preliminary, due to the intrinsic difficulty of performing user studies in the RS domain.

2.2.3 System-centric vs. User-centric Evaluation

System-centric evaluation is widely performed since it is immediate, economical and easy to perform in several domains and with multiple algorithms.

On the other hand, user-centric evaluation is difficult to perform, since involves real users, and it is in general more costly and resource demanding than system-centric one. Thus, several RSs are evaluated only against the system and without involving users thanks to the minor overall efforts of the former kind of evaluation.

But system-centric metrics may not be able to capture non-accuracy metrics such as novelty (i.e., how recommendations are perceived as new). So recently many researchers have argued that the system-centric evaluation of RSs in e-commerce applications does not always correlate with the perceived value of recommendations by the users.

Among them we cite two studies in the movie [13] and the hotel [14] domains. Both studies tested some personalized and non-personalized RSs in both offline and online experiments. The goal of both studies was to compare measures of *user's perceived quality* (e.g., user global satisfaction, accuracy and novelty of recommendations) against measures of *objective statistical quality of RSs* (recall and fallout) in the respective domains of application. The results of these studies suggest that RSs evaluation, and especially the relationship between system-centric and user-centric evaluation, is strongly domain-dependent.
In the movie domain personalized algorithms, which have the best performances in offline evaluation, have results comparable to the less sophisticated non-personalized algorithms in online evaluation[13]. That is not the case for the hotel domain, where offline metrics are good predictors for online metrics [14].
This because each domain has its own characteristics and peculiarities that influence the interests of users on items and, consequently, change users' perception on the quality of the recommended items.

2.2.4 Popularity Bias

A factor of non-secondary importance that must be taken in account in RSs evaluation is the distribution of ratings in the User-item Rating Matrix. In the majority of applications the URM is very sparse. For instance, in Table 2.2 are reported the statistics of Movielens and Netflix datasets, two of the most popular datasets for RSs evaluation in the movie domain. These two datasets have really low densities, respectively 4.26% for Movielens and 1.18% for Netflix dataset.

In [15] the distribution of rating in such datasets have been exhaustively studied. As expected, ratings are not distributed at random into the URM. Instead, the URM is affected by the so called *popularity effect*. In many

Dataset	Users	Items	Ratings	Density
Movielens	6,040	3,883	1M	4.26%
Netflix	480,189	17,770	100M	1.18%

Table 2.2: Statistical properties of Movielens and Netflix datasets.

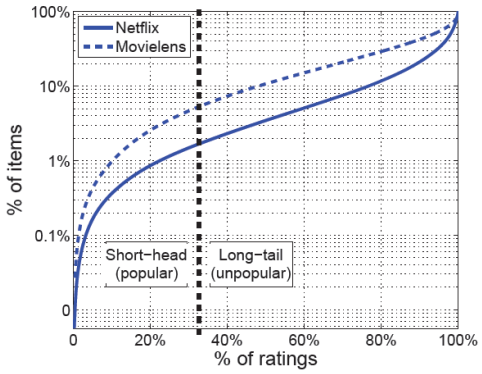


Figure 2.2: Rating distribution for Netflix and Movielens datasets. Items are ordered according to popularity (most popular at bottom).

commercial systems the majority of the ratings are condensed in a small fraction of the most popular items [3].
Figure 2.2 show the rating distribution of Netflix and Movielens datasets. We can observe the about 33% of ratings collected by Netflix involve only the 1.7% of most popular items; analogously, the same percentage of ratings involves the 5.5% of most popular ratings in Movielens. This small set of very popular items is usually called the *short-head*, and the remaining set is called the *long-tail*.
Recommending less known items is less trivial than recommend the most popular ones. It also adds novelty and serendipity to the users. Therefore, one must take in account the distribution of ratings before commenting the results of the evaluation of a RS.

In [15] the authors also studied the accuracy of RSs in *recommending non-trivial items*. To this purpose, they analyzed Netflix and MovieLens datasets. They subdivided the test set into two separate subsets, T_{head} and T_{long}, which corresponds to items taken respectively from the short-head and the long-tail of each dataset. Then several personalized collaborative RSs and non-personalized RSs were tested against both the entire test set and the T_{long} dataset. By doing this, the authors were able to evaluate the performances of RSs in recommending non-trivial items, namely the items that are in the long-tail.

The experimental results show that the more non-personalized algorithms perform very well when recommending items in the entire test set, even better than several more sophisticated collaborative RSs. Conversely, they significantly down-perform respect to the initial condition in recommending items in the long-tail. This because non-personalized algorithms trivially recommend the most popular items. When they are forced to recommend items in the long-tail, they are not able to provide useful recommendations at all.
On the other hand, collaborative algorithms perform well in every condition. When recommending items in the long-tail, they are still able to provide useful recommendation since they are less biased by item popularity.

This fact should suggests to take care about the construction of the test set in the evaluation process, since a careless construction could bias the entire offline evaluation toward non-personalized algorithms. This is the so-called *popularity bias*.

The study of item popularity seems similar to the purpose of our work, but it is indeed different. The study of item popularity solely aims to describe the influence of item distribution over the offline evaluation of RSs. It shows that offline metrics are often biased toward popularity-based algorithms, which are able to provide to users only the most trivial recommendations.
We are instead interested in modelling the item consumption process and in studying the behavior of RSs when item availability varies over time. As we will see, this process is not exclusively affected by the popularity of the items.

2.2.5 Non-Random Missing Ratings

The process through which the users select the items they choose to rate, i.e., the rating observation process [43], has a great impact on the distribution of ratings in the URM. Users do not observe items at random. Consequently, ratings are *not missing at random* in the URM.
Instead, the distribution of ratings in the URM is skewed by two effects:

- *popularity effect*, that refers to the short-head/long-tail distribution we have talked in the previous Section;
- *positivity effect*, which refers to the propensity of users to rate more often the items they like, and to rate less the items they do not like.

Many studies in RSs are founded on the assumption that ratings are missing at random. But this incorrect assumption about missing data can bias the

models used by several recommendation algorithms.
For example, the study in [43] shows that collaborative algorithms (neighborhood based and matrix factorization) can achieve significantly better performances in rating prediction and ranking if a non-random missing data model is considered when training the RSs.
Another recent study [54] studies the effect of (1) ignoring missing ratings, and (2) treating them as negative ratings, on the ranking performances of a RS.
The study shows that (1) ignoring missing ratings leads to a dramatically biased evaluation in the presence of the positivity effect, and (2) considering missing ratings as negative biases the evaluation toward models that favor popular items, because the popularity effect has a greater order of magnitude than the positivity effect. These properties suggest that choosing the importance of missing data can be crucial when training a recommender algorithm with user-selected items.

Studying the impact of non-random missing ratings over the evaluation of RSs still differs from the interests of our work. The studies we mentioned before focus on the distribution and management of missing ratings in URMs, while we are interested in modeling the distribution of *missing items*, that is indeed another issue.

2.3 Time-Evolution of RSs

Finally we want to present another important phenomenon in Recommender Systems. Many e-commerce applications offer huge catalogs of products of users. They help users in their decision process by recommending them interesting items through collaborative RSs. Especially in the movie domain, collaborative filters are preferred to content-based ones principally because they rely on explicit or implicit opinions expressed by users rather then explicit content description [40].
However, collaborative RSs are affected by the cold-start problem we described in Section 2.1.2.
Moreover, RSs evolve over time. At each point of time we can distinguish between *old* existing users and *new users*, as well as between *old* and *new items*:

- *new users* have very few ratings available to describe their profile when they register with the RS;

- *new items* have no ratings when they are added to the catalog.

Therefore, at each point of time the system suffers from some cold-start issues.
A study in [16] focuses on the time-evolution of RSs in the Interactive TV domain. The authors modeled the time-evolution process of RSs when new users and new items are continuously added to they system over time. Moreover, they evaluated the quality of two collaborative recommender systems, one item-based and one matrix factorization (SVD-based) method, over time. They also analyzed the time evolution of the algorithms with respect to item popularity (Section 2.2.4).
Their results show that item-based algorithms perform better with respect to SVD-based algorithms in the early stages of the cold-start problem. However, SVD-based algorithms, when used with a large-enough number of latent features, can outperform item-based algorithms over time if the dataset does not present a long-tail behavior.

The time-evolution of RSs is a really interesting problem and of great practical impact, because it is really hard to predict how RSs will behave in real non-static environments.
All the prior studies in this field consider the time-evolution of RSs as a purely *additive* process, in which new users, new items and new ratings are inserted in the system.
Conversely, in our study we are interested in analyzing the effects of the *subtractive* evolution of the datasets. In this case neither items nor users nor ratings are added the system, instead many items becomes unavailable due to contingent situations depending on time or on context.

2.4 Elicitation Methods

In the e-tourism domain RSs can help travellers in finding attractive tourism products and destinations that best fit with traveller's expectations and attitudes. They can be used to enhance the quality of online travel agencies (OTAs), by increasing the overall satisfaction of users over the entire e-commerce process and, consequently, bringing better economic results to companies.
RSs achieve these results by personalized user profiling. RSs try to infer users' interests and preferences to provide them personalized recommendations. Such information as well as information about the users' previous experiences is stored in a user profile.
A recommender system can maintain an individual user model or some users models that represent classes of users (i.e., stereotypes). The way user models are stored may change the degree of fitting of the recommendations to

current user's interests.
Whatever the kind of user model is maintained, it is necessary for the recommender system to acquire the information necessary to build the user model. This information can be elicited explicitly or implicitly from the user, or in both ways.

In *explicit elicitation* the user directly develop and maintain its online profile. This allows the recommender system to build a high quality user model and, consequently, to provide very good recommendations. However, the number of questions required to build an accurate user model may burden the user. Furthermore, users may not be able to describe accurately themselves and their preference.

In *implicit elicitation* instead the user profile is inferred from user's previous interactions with the system. Implicit user modeling is considered more reliable and less intrusive than explicit user modeling. However, the hypotheses generated by the system for each user may not be accurate. Furthermore, the system may not have observed the user's behavior for a sufficient time to produce accurate hypotheses on user's preferences.

In e-tourism explicit elicitation is often preferred to implicit elicitation. This because hotels have a much more complex structure than, for example, books or movies, and it is hard to establish reasonable user profiles by observing only user's interactions with the system. Therefore, many currently available approaches use explicit elicitation to acquire the preferences and requirement of users before providing recommendations.
For example, Trip@vice [58] provides complete travel collaborative recommendations based on an explicit user profile. During the first stage of interaction with the system, the user is asked to provide some information about personal and travel characteristics. This information is used to create a set of collaborative features, that include group composition, means of transport, type of accommodation, budget, etc. User's preferences over collaborative features are then used to build the user model and compared with stored items' features to provide collaborative recommendations.
Another example of explicit elicitation in e-tourism is adopted by ADVISOR SUITE [28]. It uses a conversational approach between the user and the system. The user profile is built from the answer to a series of questions that the system poses to the user over its preferences.
DieToRecs [59] uses both techniques of elicitation. Explicit elicitation is performed using the conversational nature of the system during all the user session, by asking the user to provide some preferential features, such as nationality and travel purpose). Meanwhile, implicit elicitation gathers information when the user selects a package suggested and the system stores

the content preferences expressed with this choice.
In [34] is proposed a context-based travel recommender system. It aims to provide useful recommendations to users at the cold-start by asking them information over their nationality and travel intent. Moreover, users are asked to express their preference over a set of features, such as hotel location, services, food, and rooms. This information is used to provide recommendations that are aware of the current context of interest of the user.

Implicit elicitation is used by the Personal Travel Assistant (PTA) [12] to recommend flights to users. The system uses implicit features that describe user's preferences over previously viewed travel offers, and to represent user's overall travel preferences. Such preferences are continuously refined with every user-interaction, and then used to provide collaborative recommendations. PTA was developed for PDA or mobile phones, where the conversational approach used in explicit elicitation is not feasible.

2.5 Opinion Mining and Summarization

Before the World Wide Web there was a little amount of opinionated text available. When an individual needed to make a decision, he/she typically asked for opinions from friends and familiars. When an organization wanted to investigate over the general public opinion over their products, it needed to conduct surveys or opinion polls.
But the Web has dramatically changed this perspective. Now users can post reviews of products on blogs, forums, discussion groups, etc. This is often called *user-generated content*. Together with the explosion of the user-generated content available online, also the behavior of users and company has changed. Now consumers can directly use the reviews available online to guide their purchases, and companies can extract consumer opinions about their products directly from user-generated content available on the Web.
An interesting set of examples of the influence of peer-opinions over consumers are reported in [49]. For instance, it has been discovered that consumers have a greater willingness to pay (from 20% to 99% more) for a 5-star-rated item than a 4-star-rated item; moreover, about the 80% of readers of online reviews report that reviews had a great influence on their purchase (studies were conducted over more that 2000 American adults each).

User-generated content can be categorized into two main types: *facts* and *opinions* [37]. Facts are objective expressions about entities, events and their properties. Opinions instead usually are subjective expressions that describe people's sentiments and feelings toward entities, events and their properties.

The concept of opinion is very broad, but practical applications focuses on opinion expressions that express positive or negative sentiments. The automated process of opinion discovery over huge datasets of user-generated content is called *sentiment analysis*, or *opinion mining*. Once collected, opinions can be used to extract, summarize and organize user-generated content to be easily accessed by a human reader. This process takes the name of *opinion summarization*.

We examine here the main aspects of opinion mining and summarization. Because of their significant impact over the decision making process, we want to investigate here over some possible connection between the analysis of opinionated text and the role of recommender systems in real e-commerce applications.
We therefore present here the *problem of sentiment analysis*; then we present the state-of-the-art methods in *sentiment and subjectivity classification*, in *feature-based sentiment analysis* and in *opinion summarization*. At the end, we present the state-of-the-art RSs that use opinion mining in their processing.

2.5.1 The Problem of Sentiment Analysis

Sentiment analysis or opinion mining is the computational study of opinions, sentiments and emotions expressed in text [37]. In general opinion are can be expressed on anything, e.g., a product, a service, an organization, etc. This is the *object* of the opinion.
Usually *specific opinions* are expressed over a set *features* of the object. Features can be subdivided into:

Explicit features, if a feature f or any of its synonyms appear in as sentence s.

Implicit features, if neither f nor any of its synonyms appear in s but f is implied through a *feature indicator*.

The *holder* of the opinion is the person of organization that expresses the opinion, such as the authors of posts in reviews and blogs. An *opinion* on a feature f is a positive or negative view, attitude, emotion or appraisal on f from the opinion holder. Its *orientation* indicates whether the opinion is *positive*, *negative* or *neutral*.
Using these elements it can be defined the *feature-based sentiment analysis model* [26, 36, 38].

Model of an object: An object o is represented with a finite set of features, $F = \{f_1, f_2, \ldots, f_n\}$, which include the object itself as a special

feature. Each feature $f_i \in F$ can be expressed with any one of a set of *synonym* words or phrases $W_1 = \{w_{i1}, w_{i2}, \ldots, w_{im}\}$ or indicated by a set of feature indicators $I_i = \{i_{i1}, i_{i_2}, \ldots, i_{iq}\}$.

Model of an opinionated document: A general opinionated document d contains opinions on a set of objects $\{o_1, o_2, \ldots, o_q\}$ from a set of opinion holders $\{h_1, h_2, \ldots, h_p\}$. The opinions on each object o_j are expressed on a subset F_j of features o_j.

An opinion can be any of the following two types:

1. **Direct opinion,** which expresses an opinion from the holder directly over a feature of the object.
   It is represented as a quintuple $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$, where o_j is an object, f_{jk} is a feature of the object, oo_{ijkl} is the orientation of the feature of object, h_i is the opinion holder and t_l is the time when the opinion is expressed by the holder.

2. **Comparative opinion,** which expresses a relation of similarities or differences between two or more objects, and/or object preferences of the opinion holder based on some of the shared features of the objects. They are usually expressed using the comparative or superlative form of an adjective or adverb.

Comparative opinions should be handled because they are frequently used by product review authors. But authors of hotel reviews, that is our domain of interest, usually focus on the specific hotel they describe. They usually do not compare the characteristics of different hotels in their reviews. Therefore, this is not an essential task for our analysis and we do not consider it here.

The goal of sentiment analysis or opinion mining can be formulated as follows: Given an opinionated document d_i, opinion mining aims to

1. discover all opinion quintuples $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$ in d_i and

2. identify all the synonyms (W_{jk}) and feature indicators of each feature f_{jk} in d_i.

Opinion mining does not only aims (1) to infer positive or negative opinions from text, but also (2) to discover other pieces of information which are important for practical applications.

As we have mentioned before, user-generated content can be categorized into facts and opinions. Therefore, in a typical document some sentences

express opinions and some do not.
Sentences can be therefore classified into *objective sentences*, which express some factual information about the world, and *subjective sentences*, which instead express some personal feelings or beliefs. The process of classifying sentences according to their objectivity/subjectivity is called *subjectivity classification*.

Opinions expressed into subjective sentences are called *explicit opinions*, while opinions implied into objective sentences are called *implicit opinions*. Sentences which express an explicit or implicit positive or negative opinions are called *opinionated sentences*. They can be subjective or objective sentences. Clearly identifying implicit opinions is a more challenging task respect to identifying explicit ones.

2.5.2 Sentiment and Subjectivity Classification

Among the several topics in sentiment analysis, sentiment and subjectivity classification are perhaps the most studied. Subjectivity classification is the process that aims to classify sentences as opinionated or not opinionated. Once opinionated sentences have been recognized, they need to be further classified into positive or negative sentences, according to the opinion they express. This process is called *sentence-level sentiment classification*, because sentences are considered as the basic information unit. When instead document (e.g, a product review) is the basic information unit, we are talking about *document-level sentiment classification*.
Previous studies in sentiment and subjectivity classification used both supervised [50, 64] and unsupervised [17, 60, 63] learning techniques.

However, compound sentences can express multiple opinions over different objects. In this cases, sentence-level classification is not applicable. Moreover, also objective sentences may imply opinions. Therefore, it is necessary a more flexible approach to mine opinions in this greater variety of cases.

2.5.3 Feature Based Sentiment Analysis

Classifying opinionated text can be useful in many cases, but it does not provide the sufficient level of detail required in many applications. For example, a positive opinionated document does not mean that the author has positive opinions for every feature of the object described in the document. Analogously, an overall negative opinion does not imply that the author dislikes everything about the object.

Typically user-generated opinion documents contains both positive and negative aspects of the object, although the general opinion on the object can be positive or negative. To reach the necessary level of detail it is necessary to work at feature level.
Opinion holder, object and time extraction usually is performed by the system through Named Entity Recognition and Coreference Resolution [62]. Time as well can be extracted easily from system's timestamps and it is useful in analyzing the temporal variation of opinions.
Feature based sentiment analysis is essentially composed of two sub-tasks.

Feature Extraction

The feature extraction process aims to identify object features from documents, often online product reviews. In the most general case, documents are free-format text (i.e., the users writes freely with no prior schematization).
Free-format reviews can contain any kind of sentences. They can have any kind of complexity and usually contain a large amount of noise. To extract features from free-format reviews an unsupervised learning method is described in [26]. This method requires a large amount of reviews, and consists of two steps:

1. *Frequent feature extraction*

2. *Infrequent feature extraction*

Frequent feature extraction consists of finding frequent nouns and noun phrases in reviews. Reviews are preprocessed using stemming [52], stopwords filtering, approximate string matching [48] and Part Of Speech Tagging [11]. POS tagging is used to identify nouns and nouns phrases. Their occurrence frequencies are counted and only frequent ones (i.e., the ones with frequency greater than a minimum threshold) are kept. Frequent feature can be extracted using any of the state-of-the-art algorithms in frequent itemset and association rules mining mining, e.g., the classical Apriori algorithm [2].
In frequent feature extraction items are words and transactions are sentences. Frequent itemset mining is therefore able to discover frequent words and word phrases in a set of sentences. But mere frequent itemset mining does not always generates useful or genuine features, and it can also extract uninteresting and redundant ones. Thus feature pruning is required to filter incorrect features [27]:

- *Compactness pruning*: it considers frequent feature phases (i.e., features with more than one word). Since frequent itemset mining does not consider the position of words in a sentence, it may generate non-genuine features. Therefore compactness pruning aims to eliminate candidate features phrases whose words do not appear together.

- *Redundancy pruning*: it does only consider features of size 1, and removes frequent features that are redundant in the dataset, i.e., such frequent features that have superfeatures that are also frequent.

This approach is justified from the following consideration: when people comment on product features, their vocabulary usually converges, and most product features are nouns or noun phrases. Therefore, by mining frequent nouns and noun phrases we are able to catch important feature. Conversely, irrelevant contents in reviews are often diverse and thus infrequent.

However, frequent features may not be enough to correctly characterize opinions in review. Sometimes the same opinion word can be used to describe different feature objects, which can be both frequent or *infrequent*. Therefore, opinion words that modify frequent features can be used to detect interesting features that are not frequent in reviews.
Other solutions have been proposed in feature extraction which main use the concept of topic modeling [44].

Opinion Orientation Identification

Once object features have been extracted, we want to determine the orientation of opinions expressed on them in a sentence. One of the best known approaches are *lexicon-based approaches* [19, 26].
These methods use *opinion lexicons* to infer opinion words orientations. An opinion lexicon is a corpus or a dictionary of words annotated with their orientations. Generally opinion lexicons are generated from dictionaries such as WordNet [26], or other recent opinion lexicon as SentiWordNet [4].
Given a sentence s, the lexicon-based approach is basically subdivided into four steps:

1. *Identification of opinion words and phrases*
   This step identifies all opinion words and phrases. Each positive and negative opinion word is assigned to score +1 and -1 respectively, while context dependent opinion words are assigned to score 0.

2. *Negation handling*
   In this step the previously computed scores are revised according to a

set of rules that consider the effects of negations in natural languafe [37].

3. *But-clause handling*
   A sentence containing "but" (or words with similar behavior such as "except for" and "with the exception of") is handled by applying the following rule: the opinion before "but" and after "but" are opposite to each other.

4. *Opinion aggregation*
   At the end an opinion aggregation function is applied to the resulting opinion scores of each opinion word $\{op_1, op_2, \ldots, op_n\}$ to determine the final orientation of the opinion of each object feature $\{f_1, f_2, \ldots, f_m\}$ in the sentence s.

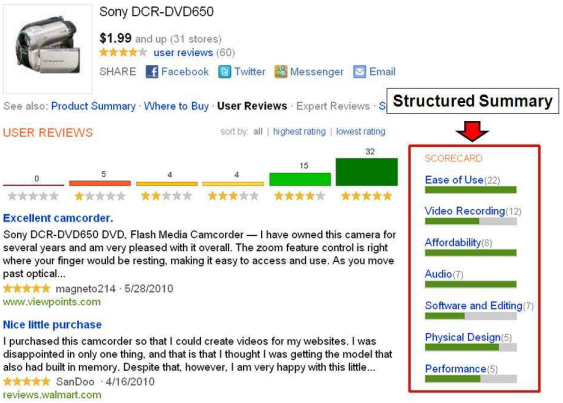
Natural language is really complex and the previous set of rules may not cover some special cases that may occur in user-generated content. Nonetheless, feature-based sentiment analysis is a general method able to detect opinion in many real contexts.

2.5.4 Opinion Summarization

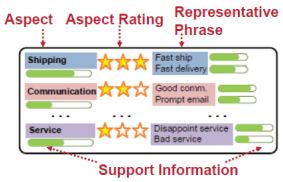
The born of World Wide Web bringed to everyone the possibility to express their opinion on whatever product we are interested on. This has produced an immense amount of opinionated data that it is almost impossible to manage "as is". Opinion mining provides a very effective base of knowledge in order to generate a concise and digestible summary of the opinions expressed in big amounts of user-generated content over a specific product, individual or organization.

In the previous Section we have presented a general framework to analyze opinionated text and extract opinions from it. The simplest form of opinion summary is the aggregation of the sentiment scores produced by sentiment analysis. More advanced summarization techniques can produce structured, textual and temporal summaries, which provide a more understandable and detailed analysis of the opinion.

Opinion summarization techniques can be categorized into: *aspect-based* and *non-aspect-based* opinion summarization [30].
Aspect-based opinion summarization is the most common summarization technique. Generally it is composed of (1) *feature-based sentiment analysis* and (2) *summary generation*.
The *summary generation* process aggregates the results of the previous steps to produce a concise and understandable summary.



(a) Statistical summary from Bing.



(b) Summary generated by [41].

One option is to create a *statistical summary* that directly shows the statistics of positive and negative opinions for each feature. Several presentation formats have been proposed in literatures, e.g., list [26] and graph [41] formats. Figure shows an example of statistical summary from the website Bing [1]. Another option is to perform a text selection to reach a better level of understandability. For example in [51] they propose to rank words associated to features and show the strongest opinionated word for each aspect. But also short representative phrases can be automatically generated, as shown in [41]. They propose to aggregate ratings using clustering topic modeling, then the sentiment scores for each aspects are averaged and presented with representative phrases.

Non-aspect-based Opinion Summarization is usually combined or incorporated into the first, although it is based on different principles. Since aspect-based format is strictly correlated with the feature based sentiment analysis we are interested in, we do not explore it here.
A comprehensive survey on both kinds of summarization is available here [30].

2.5.5 Opinion Mining and Recommender Systems

Sentiment analysis and opinion mining can act as in interesting source of information for Recommender Systems. As we explained in Section 2.1, content-based RSs use content information associate to items to recommend to users items of its interest. On the other hand, collaborative RSs exploit

[1] http://www.bing.com

the interconnections between users and items to recommend to users items that people with similar tastes liked in the past. Feature-based sentiment analysis can be a very valuable source of structured information to improve collaborative filtering performances, by adding qualitative information to explicit user ratings.

However, only a little research have been done in this direction so far. In [33] is presented a preliminary study about user rating enrichment via the analysis of user-generated content in educational repositories.
In [34] is proposed a context-aware recommender system that uses the sentiment analysis of hotel reviews [37] to model user tastes according to their nationalities and travel intents in order to provide better recommendations in the cold-start phase.

2.6 The Framework PoliVenus

In our studies we have extensively used the framework PoliVenus. PoliVenus has been developed by the Politecnico di Milano [20] in cooperation with Venere.com [2]. It provides a powerful environment based to test and evaluate RSs in the hotel domain.
It is composed of different functional units, namely the dataset building and data retrieval unit, the content management and processing unit, and the testing unit.
The dataset building and data retrieval unit collects data from Venere.com and crawled from Tripadvisor.com [3]. Such data regards users' ratings, reviews and hotels' descriptions.
The content management and processing unit manages such data to create the necessary base of knowledge for the Recommender Systems. From textual data (description and reviews), the TF-IDF schema necessary for content-based algorithms is extracted. User ratings are instead used to create the User-item Rating Matrix.
The test init provides all the necessary instruments to create and manage several experiments. It allows to configure the recommendation algorithm (content, collaborative, hybrid or none) to be used, the task asked to the user and the availability of hotels.

PoliVenus offers an highly configurable environment to perform user-centric evaluation of recommendation algorithms. It reproduces the same user-experience of Venere.com (the payment is only simulated), with the ad-

[2] http://www.venere.com/
[3] http://www.tripadvisor.com/

dition of recommendations that are presented to users in different moments of the user-experience. It can be schematized in the following parts:

- the *homepage*, in which the user selects the desired destination.
- the *hotel list* page, in which the system presents to the user the list of hotels available in the desired place. Hotels can be filtered according to budget, number of stars, typology of hotel, and location. The user is able to sort the hotels according to price, number of stars, number of views and hosts' opinion.
- the *hotel detail* page, in which the user is able to explore the details over a specific hotel. It contains the information over hotel general description, location and comments. It presents to the user the list of rooms available together with their capacity and prices. The user can make the reservation only through this page.
- the *confirm reservation* page, which offers to users the possibility to revise their reservation, and in case to confirm it.

It can be configured to work with more then 20 recommendation algorithms (content-based, collaborative and mixed hybrid):

Content-based: use an Item Content Matrix extracted from users' reviews and hotel descriptions through a TF-IDF schema. PoliVenus implements LSA, DirectContent k-Nearest-Neighbors and Naïve Bayes content-based algorithms.

Collaborative: Polivenus implements two latent factors algorithms (AsySVD and PureSVD) and 8 neighborhood algorithms, mainly item-based algorithms that use different similarity metrics, such as the cosine similarity and the Pearson similarity.

Mixed hybrid: three mixed hybridization techniques are available, namely maximum rating, average rating and interleaved results.

Non-personalized : three non-personalized algorithms are available, namely MovieAvg, TopPopular and Random algorithms.

The recommendations algorithms in PoliVenus use an URM that collects the ratings over 3164 italian hotels from users in Venere and Tripadvisor. The URM is a very sparse matrix, with more than the 99.97% of missing ratings. In average there are available ~ 77 ratings per hotel and ~ 0.8 ratings per

	Total	Venere	TripAdvisor
Hotels	3164	3164	-
Users	293K	107K	186K
Reviews	246K	81K	165K
Avg. Ratings/User	0.82	0.75	0.88
Avg. Ratings/Hotel	77.72	25.46	52.26
Density	0.000265	0.000238	0.000281
Hotel features	481	481	-

Table 2.3: Statistics of the URM, reviews and the number of features used by PoliVenus.

user. The detailed statistics over the URM are reported in Table 2.3.
The desired type of recommender, as well as the content-based and collaborative algorithms to be used, can be easily configured before running any test. PoliVenus allows to create a set of different test configurations to be tested in user experiments. Each test configuration is stored in a database record, and can be assigned to users in Round Robin fashion, in order to assign uniformly each experimental condition over the set of users. For debug purposes the test configuration can be also configured by changing the URL parameters. Clearly this functionality can be disabled in the real experiment session.
Recommendations are integrated into the user-experience of Polivenus and they are provided to user in two different moments:

1. The list of recommended hotels is integrated into the list of hotels, with the addiction of an option in the menu named *"recommended for you"* (Figure 2.3). Recommendations are introduced with the same information of the normal list content (short description, thumbnail, user ratings and links to detail page).

2. Another list of recommended hotels is integrated in the hotel detail page. Here hotels are presented with a reduced description due to space constraints (Figure 2.4).

During the testing session, the system tracks all user activity (the interaction with objects in the user interface and the spatial coordinates of the pointer) in a log. This log can be analyzed a-posteriori to extract interesting information over, for example, the time elapsed, the number of pages explored by the user and number of user's interactions. This information is useful, for instance, to implement a more effective implicit elicitation technique, but also to discard meaningless users' testing sessions.

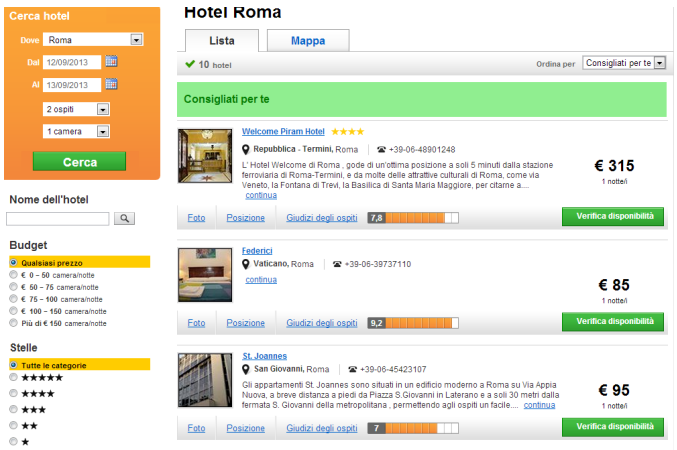


Figure 2.3: Recommendations in the hotel list page in PoliVenus

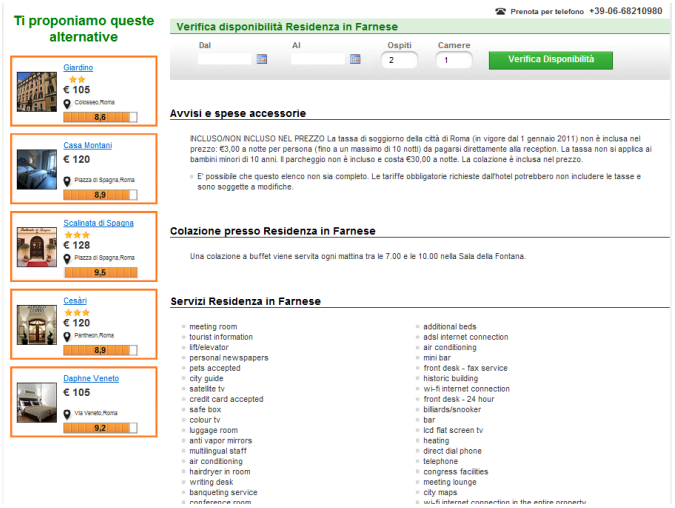


Figure 2.4: Recommendations in the hotel detail page in PoliVenus

Chapter 3

Implicit Elicitation

RSs help user in their decision making process by providing them personalized recommendations. The RSs needs to infer user's preferences and interest in order to recommend items that fit with user's needs. The information needed to perform the recommendation step is stored in an *user profile*.
In implicit elicitation (Section 2.4), each profile is based on the user's interactions with the system. The system does not ask the user for its interests, instead it tries to infer user's preferences by observing him/her activity.
We decided to use implicit elicitation for three main reasons:

- e-tourism involves many personal preferences and tastes that the user is reluctant to supply to the system. Consequently, many currently available e-tourism applications allow also to unregistered users to their reservations. To support users who have *no rating history* or who are *not interested in logging* into the system, an implicit elicitation method is necessary.

- we are interested in exploring a *smooth integration* of personalized recommendations in existing online booking systems. Implicit elicitation is functional to this scope, because it does not requires the introduction of an intrusive add-on into the system. Explicit elicitation instead requires several modifications to the user experience to ask user's preferences and to the system's back-end to handle the information retrieved from the user.

- according to a large number of works, the *lower effort* of implicit elicitation with respect to explicit elicitation increases the perceived effectiveness of recommendations [21, 24, 29, 57].

On the other hand, the analysis of user activity within the system has some limitations:

- the *absence of negative feedback*, because we are only able to detect what are the objects the user has interacted with, but we cannot know why he/she does not used some of functionalities offered the system.
- user activity data is *noisy*, because we are not able to distinguish whether the user interacted with an object for curiosity or guided from real interest.
- the *number of interactions* does not necessarily express an interest from the user, but only it express the frequency of the interactions. This two measures are not necessarily related with each other, because many interactions with an item may not imply user's appreciation for such item. However, the greater is the number of interactions we are able to collect, the higher is the statistical significance of the conclusions we can infer from it.
- implicit elicitation requires appropriate measurements to take in account, for instance, repetitiveness in user's interactions.

The objects in the PoliVenus interface the user can interact with are *links*, *buttons*, *maps* and *pictures*. In Appendix A.1 we report all the objects that are available in the user experience. For each user action over an object, PoliVenus stores the following information record:

- *session id*, which identifies the current user.
- *page id*, which identifies the page from which the action starts.
- *page type*, which identifies the category of the page (homepage, hotel list, hotel detail, etc.).
- *object*, which identifies the object the user has interacted with.
- *object type*, which identifies the category of the object.
- *timestamp*, which stores the date and time of the action.

PoliVenus originally implements an implicit elicitation method that, whenever the user interacts with an object on the interface, it assigns a score to the hotel related to the object. Not every object in the user interface generates an hotel score, but only a selected subset of active objects.
At each interaction with an active object in the user interface, the implicit elicitation method generates a *signal*. Each active object in the user interface has a different weight proportional to extent to which the interaction

of the user with the object expresses an interest for the hotel that is related to the object.
The user profile contains implicit hotel ratings, where each rating is the linear combination of all the signals generated for that hotel. The user profile is used by recommendation algorithms to provide recommendations to new users. Given an active object f, an user u and an item i, the explicit rating $\hat{r}_{ui}$ for the user profile is computed as follows:

$$\hat{r}_{ui} = \sum_{f} w_f \cdot s_{uif} \tag{3.1}$$

where w_f is the weight associated to object f and s_{uif} is the number of interactions of u with i through object f.
Each new signal updates the user profile, and, consequently, it updates the list of recommended hotels. In order to give more importance to most recent interactions, the signals are scaled with an exponential decay function. Whenever a new signal is generated, all previous ratings in the user profile are multiplied by a dumping factor. Formally, whenever a new signal f on item i is collected for user u, all the other signals for the same user are divided by a dumping factor h:

$$\begin{aligned} &s_{uif} \leftarrow s_{uif} + 1 \\ &s_{ujg} \leftarrow s_{ujg} h \text{ (for } j \neq i \text{ and } g \neq f) \end{aligned} \tag{3.2}$$

PoliVenus adopts a decay factor $h = 0.75$. This fact means that at every new interaction, the weight of previous ones in reduced to the 75%.

Originally active objects and object weights were decided manually by assuming a probable degree of importance of each kind of interaction. For example, objects associated with the hotel availability check or booking action were considered as more important than links to hotel gallery and map, and consequently associated with higher weights. However, there was no empirical evidence of the correctness of such assumptions.
To improve the quality of the estimation of user preferences, we analyzed the information records obtained from a previous user experiment with PoliVenus over 240 users. The user experiment was used to evaluate user satisfaction when different types of recommendations were provided by the system [14].
We analyzed what were the real degrees of object interaction of users that successfully concluded the experiment by booking a hotel. The interactions of these users are clearly the most reliable. It turns out that the majority of objects in the user interface are rarely used, a part from the objects connected with hotel availability and booking. Therefore the less utilized

Object	Page	Function	Weight	Figure
Location tab	Hotel details	Link to hotel position on the map	0.7460	A.3
Box review	Hotel details	Link to hotel reviews	0.4874	A.3
Overview tab	Hotel details	Link to hotel detailed description	0.4439	A.3
Union check availability	Hotel details	Union of the links to hotel availability	0.2653	A.3
Big image	Hotel details	Link to hotel photo gallery	0.1929	A.3
Availability button	Hotel list	Link to detail page to check the hotel availability	0.0956	A.2
Name link	Hotel list	Link to hotel detail	0.0942	A.4
Description link	Hotel list	Link to detailed description of the hotel	0.0527	A.4
Review tab	Hotel details	Link to hotel reviews	0.0125	A.3

Table 3.1: Object used for implicit elicitation in Polivenus ordered by signal weights.

objects were deactivated, and some of the remaining ones where grouped according to their semantic function. The resulting schema is shown in Figure 3.1.

From the number of interactions of each active object, we can were able to learn the weights to be associated to each one. Since the user profile should contain the estimated ratings of each user to the specific item he/she has interacted with, we used real ratings taken from the URM of PoliVenus as reference ratings.

If the rating given by user u to item i is r_{ui}, the weight w_f associated to feature f is computed by minimizing the squared error:

$$\underset{w_f}{\operatorname{argmin}}(r_{ui} - \sum_{u,i} w_f \cdot s_{uif}) \tag{3.3}$$

We normalized the number of interactions s_{uif} in order to give the same relevance to every active object. The resulting weights are reported in Table 3.1. Normalization tends to assign higher weights to less used objects such as the location tab, the box review and , because any interaction with such objects is potentially more informative than any interaction with frequently used objects. In other words, we can infer real interest when the user's behavior differs from the common actions that are usually done when exploring the catalog of hotels. A part from these cases, our estimation assigns high weights to objects that can correlated to user interest by common sense, such as the union of availability buttons and the link to hotel gallery in the hotel detail pages.

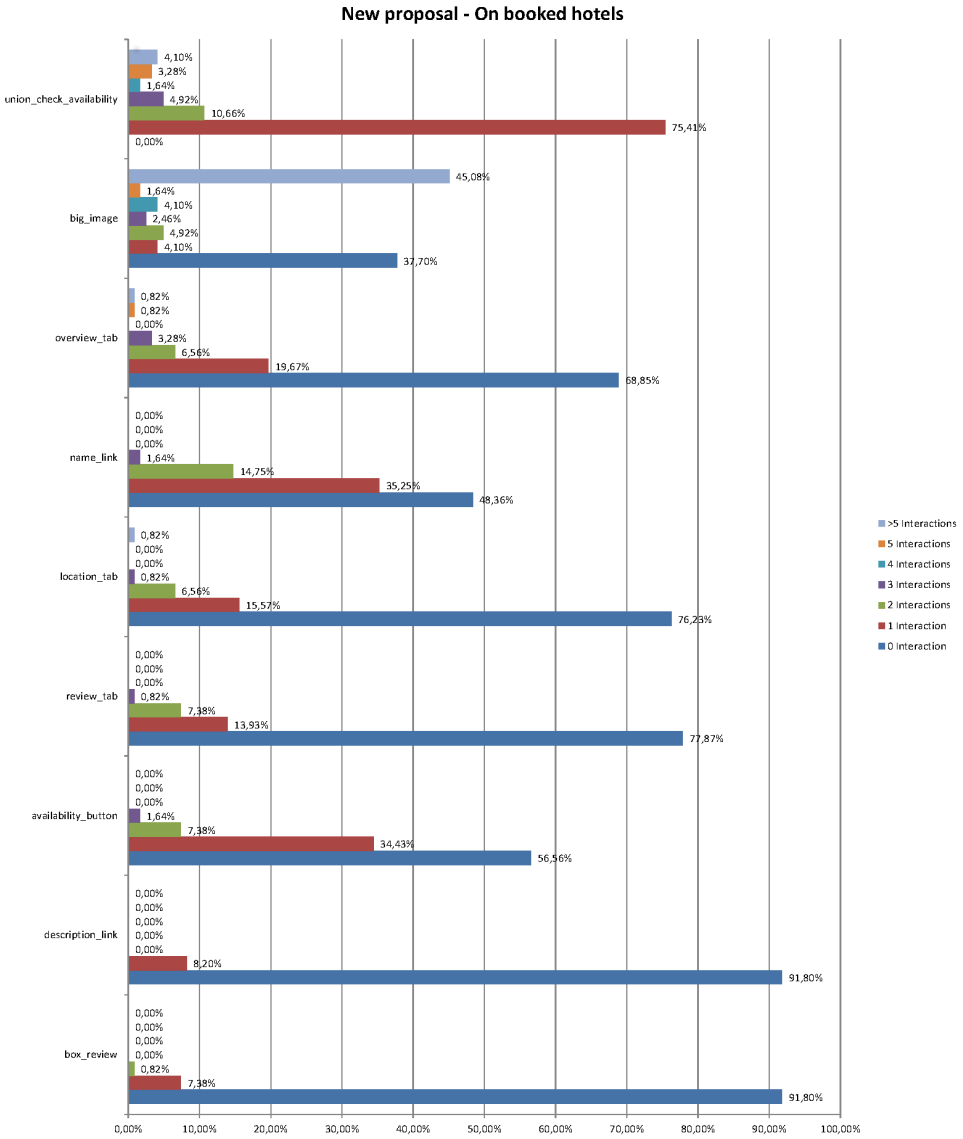


Figure 3.1: Active objects used in implicit elicitation, together with their degrees of interaction.

Chapter 4

Bounded Availability

In this chapter we present our study on simulating bounded availability in the e-tourism domain. We want to study the effects of limited availability of hotels over the perceived quality of recommendations by users. The scarcity of hotels is a real issue in hotel booking, especially in high season periods. Moreover, items are not consumed by users at random. Instead, as the number of available items decreases, the quality of the remaining one tends to decrease, as the "*best*" hotels are the first to be consumed. Therefore, we want to accurately simulate the item consumption process in order to study the behavior of RSs in this circumstance.
We explain here the methodology we adopted to create an effective simulation of limited availability of hotels during high season periods. This is a general method that can be applied to any domain in which the offer of items depends on contextual circumstances and varies over time.

In high season periods of the year, or when the time of booking is close to to the desired time of usage, most of the "*best*" hotels are already booked and the remaining ones are often the less interesting. This because the most popular hotels and the ones with highest ratings are usually the first to be *consumed*, and the remaining ones are often in the less attractive locations of the city or may offer low quality service. We define this condition as "*the best are gone*" condition, because the best items are now unavailable to the current user since they have been consumed previously by other users.

This qualitative description reveals a peculiar aspect of the e-tourism domain. In video-on-demand or e-book business, for example, the electronic format allows a potentially infinite number of costumers to consume them at any time. In e-tourism instead an item can be consumed by a limited number of users, that corresponds to the number of rooms available in the

hotel in a certain day or period.

Items in hotel domain are bounded by their own *capacity*. An item can be consumed until its capacity is reached. After that, the item has to be considered as a *missing item*. The possibility for a customer of consuming items depends on *contextual circumstances* and *varies over time*. For instance, in e-tourism a hotel may become unavailable during high season or when the booking time is close to the desired time of usage.
From this we can derive our definition of *bounded domain*:

Bounded domain It is any domain in which the availability of its items is *bounded* by their capacity, i.e., the maximum number that can be consumed in a certain time.

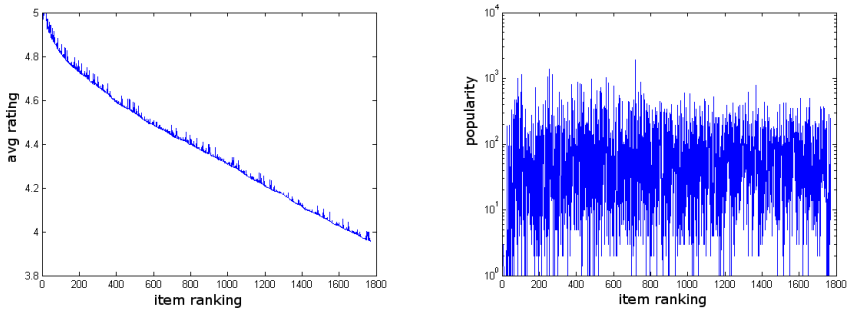
Bounded domains are, for example, e-tourism, the clothing market, and event organization. In general, a domain is bounded whenever the consumption of an item has a *subtractive* effect on the set of items available to users. At any time in which the number of consumptions of an item exceeds item's capacity, this item becomes unavailable to any user, i.e., it is subtracted from the set of items. The item becomes available only when a new capacity of such item is added to system.
This domains are of great practical interest since this bounded conditions affects many real applications. Moreover, it can be seen that as the number of items decreases, also the quality of the remaining ones tends to decrease. This because items are *not consumed randomly*. Instead, often the best items are the first to be consumed.
The main issue concerns the identification of such best items. To identify what are the best items two indicators can be taken in account:

1. Item's average rating
2. Item's popularity

According to common sense, the best items are the one with the largest *average rating*. Given an item i its average rating can be computed as:

$$\bar{r}_i = \frac{\sum_u r_{ui}}{n_i} \tag{4.1}$$

where r_{ui} is the rating given from user u to item i, and n_i is the number of users who rated item i (i.e., its popularity). However average ratings computed over a larger support n_i are considered as more reliable by the users. The two metrics are not necessary correlated, as low popularity may come with high hotel ratings and viceversa.



(a) Average rating. *(b) Popularity.*

Figure 4.1: Distributions of the average rating and popularity in the short-head for $k = 0.01$

To overcome this ambiguity, we adopted the following definition of *shrinked average rating*:

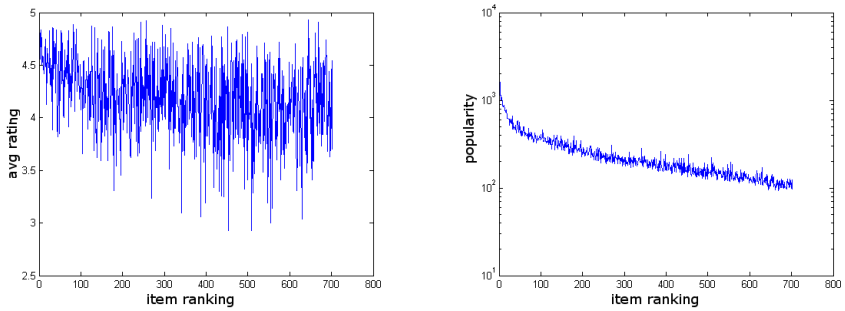
$$\bar{\bar{r}}_i = \frac{\sum_u r_{ui}}{n_i + k} \tag{4.2}$$

where the value k is the shrink factor. Here we defined as the *short-head* as the set of items (hotels) that contains the first 66% of the ratings of the URM. Different values of k produce different behaviors of the value $\bar{\bar{r}}_i$. Analytically it can be seen that for $k = 0$, items are ranked according to the average rating defined in (4.1). For $k \to \infty$, items are ranked according to their popularity. This because the lower is item support, the higher is the weight of the shrink factor at the denominator of (4.2).
But we want to estimate a correct value of k that better comprises the effects of popularity and average rating on the precedent measure. Therefore, we have conducted an in-deep analysis of the impact of the shrink factor over two factors over the short-head:

1. The distribution of the average rating
2. The distribution of popular items

We used the URM in PoliVenus we described in Table 2.3.
We ranked the items according to their shrinked average rating $\bar{\bar{r}}_i$, and then we plotted the values for their average rating $\bar{r}_i$ and popularity n_i. Then we studied the resulting rankings according to different values of k.
Figure 4.1 shows the ranking of the items in the short-head for $k = 0.01$. Figure 4.1a confirms what we expected. For low values of k items are ranked by average rating, since the average rating clearly decreases as the position

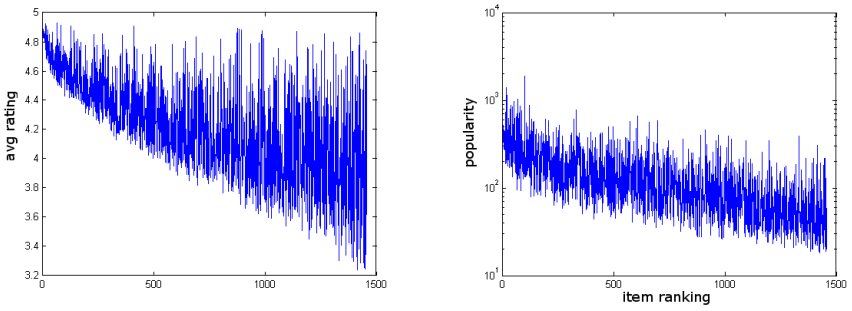
of the item in the short head increases. Moreover, no clear ordering is advisable in popularity of items (Figure 4.1b).



(a) Average rating. *(b) Popularity.*

Figure 4.2: Distributions of the average rating and popularity in the short-head for $k = 1000$

Figure 4.2 shows different results for $k = 1000$. In this case the short-head includes only 702 hotels. Also in this case we have obtained the expected behavior. Now hotels are ranked by popularity (Figure 4.2b), while no clear ordering can be advised in items' average rating (Figure 4.2a).



(a) Average rating. *(b) Popularity.*

Figure 4.3: Distributions of the average rating and popularity in the short-head for $k = 10$

Hence, we tried with intermediate values of k. Figure 4.3 shows the results we obtained for $k = 10$. Now the ordering is not as finely shaped as in the previous cases. However, it clearly shows that items are now ranked according to both average rating and popularity, because both measures tends to decrease as the position of the item becomes greater.

Therefore we used in our experiments a value of $k = 10$, since it allows us to consider both average rating and popularity in defining the short-head. Moreover, it includes in the short head 1458 hotels, almost the 50% of the items in the URM. This are our *best* hotels. This number is closed to the percentage of fully booked hotel as reported by Venere.com during high season periods in Rome. In simulating high season, we removed the hotels in

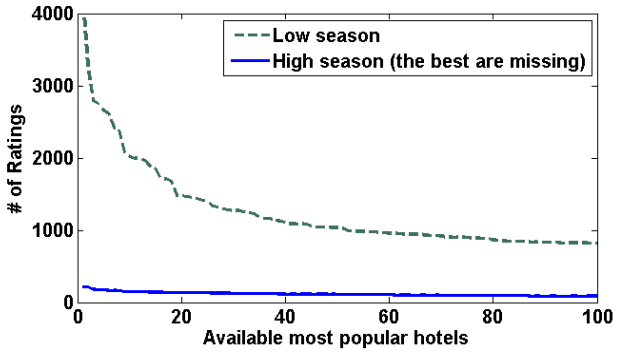


Figure 4.4: Distribution of hotels for both high season and low season.

the short-head by making them unavailable to users. Figure 4.4 reports the popularity n_i of the 100 most popular hotels for both low and high season scenario. Hotels are sorted in descending of popularity. It is clear that in the high season scenario the best hotels, i.e., the short-head, are removed from the dataset. Table 4.1 shows short resume of the experiments in short-head definition.

k	Short-head Size	Ranking
0.01	1768	Avg. Rating
10	**1458**	**Both**
1000	702	Popularity

Table 4.1: Short-head dimension and ranking for different values of the shrink factor k

Chapter 5

Empirical Study

The purpose of this thesis is to develop a recommender system for the e-tourism domain. To create a system that is able to provide useful recommendations to users in this domain, we needed to select a recommender algorithm among the plethora of currently available solutions. Based on a precedent study over the same domain of application, we used a mixed interleaved hybrid algorithm, which guarantees better users' satisfaction.
To provide personalized recommendations, we implemented a concrete mechanism of implicit elicitation to infer user's preferences.
Finally, we considered the influence of the variability of hotel availability over time. We developed an empirical study to assess the impact of bounded availability of items over the recommendations. We present here the structure of the empirical study and the results that we obtained from it.

5.1 The Empirical Study

The goal of the study is to investigate if and how missing (e.g., consumed) items affects the quality of recommender systems trained on popularity-biased datasets. We have already seen that items are not missing at random. Instead, there is clearly a bias toward most popular items in the item consumption process. Such bias makes a few number of hotels (i.e., the most interesting) to be consumed first and, hence, unavailable in high season periods. This is the short-head. Conversely, the remaining hotels (i.e., the less interesting) remain available in high season, and are the only that can be effectively consumed by the user. This is the long-tail.
In this study we are interested into investigate two issues:

1. If non-personalized algorithms have an accuracy comparable to personalized algorithms also in bounded domains, where the time-varying

availability and distribution of items may affect algorithms trained on popularity-biased datasets. We have already discuss about the distribution of ratings in URMs in Sections 2.2.4 and 2.2.5.

2. How these phenomena are influential on the quality of recommendations measured using both user-centric and system-centric metrics

The study is organized in four steps:

1. Preliminary study devoted to identify the *"most appropriate" personalized algorithms* to be used in our final case study.

2. Investigate the results of the previous steps in order to design the implicit elicitation mechanism to be used in our final study (Chapter 3).

3. Explore the process of items consumption and define the technique to simulate our "*the best are gone*" condition (Chapter 4). This step also includes the study of the variation of room prices between high and low season periods.

4. Evaluate the effects of non-personalized algorithms against personalized algorithms chosen in step 1). We want to explore the influence of the bounded condition over the perceived quality of recommendations. The comparison has been performed with both user-centric (online) and system-centric (offline) evaluation.

The preliminary study in step 1) is well detailed in [14]. Here an empirical study over 240 users was performed using PoliVenus. The study considers three algorithms, one content-based algorithm *DirectContent*, one collaborative algorithm *PureSVD* and one interleaved hybrid that combines the two previous. It compares the effects over users' perceived quality of introducing recommendations generated by different algorithms against the one achieved by the baseline scenario without recommendations. The results shown that the adoption of the hybrid algorithm significantly increases the perceived quality of the user experience. Hence, we adopted the interleaved hybridization of DirectContent and PureSVD in our experiments.

5.1.1 Dependent and Independent Variables

In this study we want to study the effects of the bounded condition over both user-centric and system-centric measures. These are the *dependent variables* of the study. User-centric evaluation considers two types of constructs:

Subjective variables: Attributes resulting from user's perception and judgment of the decision making process;

Objective variables: Objectively measurable attributes of the decision process and outcome.

Further details on user-centric evaluation are reported in Section 2.2.2. We used here an adapted subset of the ResQue variables for online evaluation [55]. Subjective variables were measured using a web survey, proposed to participants at the end of their reservation process. We have considered the following *subjective variables*:

Choice satisfaction: The subjective evaluation of the reserved hotel in terms of quality/value for the user. This variable is evaluated through the following question posed to the user in the final survey: "Are you satisfied with your final choice?".

Trust: The perceived degree of matching between the characteristics of the chosen hotel emerging from the use of the system and the real characteristic of the accommodation. This variable is evaluated through the following posed to the user in the final survey: "Will the description of the chosen hotel match its real characteristics?".

Objective variables are instead measured using interaction log data stored by PoliVenus. The *objective variables* we measures are:

Hotel price: The cost of one night for the reserved hotel.

Elapsed time: The time taken for the user to search for hotel information and make a reservation decision.

Extent of the hotel search: The number of hotels that have been searched, for which detailed information has been acquired.

Choice satisfaction is used to measure the user's perceived quality of the interaction with the system and of the recommendations.
Elapsed time and the extent of the hotel search are used to measure the effort of the user during the decision process. They are measured through the tracking of user's navigation in the web site. The elapsed time is computed as the difference between the timestamp on which the user started the task and the timestamp at the end of the task, i.e., the moment when the user makes the purchase via a simulated irreversible transaction. The extent of product search is measured as the number of hotels for which the user as accessed to the detail page, and can be easily extracted from the navigation

log.
Hotel price is used to provide us perspective over the influence of price in the decision process, since it is also influenced by our bounded condition (Section 5.1.2).

Independent variables are *room availability* and *recommendation algorithm*. Room availability has two possible values:

1. *Low season*, or fully availability;

2. *High season*, or limited availability. It corresponds to "the best are gone" bounded condition. In this case hotels from the short-head are unavailable to users.

The recommendation algorithm can be one of the following:

1. *Editorial*, which is our baseline "algorithm". It ranks hotel according to the ranking provided us by Venere.com, which is mainly based on the number of users who booked the hotel.

2. *Hybrid*, which provide interleaved recommendations from PureSVD (Section 2.1.2) and DirectContent. Content analysis takes in account 481 features (e.g., category, price-range, facilities) extracted from the hotels' description and reviews.

3. *Popular*, which ranks hotels according to shrinked average rating of Chapter 4.

Editorial and Popular are non-personalized recommendation algorithms. Instead Hybrid is a personalized recommendation algorithm. We have a total of 6 experimental conditions, each one associated to a different combination of values of the two independent variables.

5.1.2 Price Variability

Another variable that is subject to time variation is room price. In e-tourism prices tend to rise in high season, when many tourist are looking for an accommodation in the city, and to fall in low season periods, when best prices are offered by hotels' owners to attract the maximum number of tourists. Price is an important discriminant in the decisions of users in every e-commerce activity. Therefore it must be somewhat modeled in order to provide users the most authentic experience within our tests.

Price variation over time is a complex phenomena to be modeled. We are not interested into an exact and precise modeling of such phenomena,

but rather into a simple but fair representation of its effects over the rooms' prices in different seasons. Therefore, we have considered the best 30 hotels in Rome and Florence according to the ranking provided by Tripadvisor.com. For each hotel we have extracted the prices for both high and low seasons from Venere.com. Then we applied linear regression to the obtained data to model the prices in high season with respect to low season ones. Figure

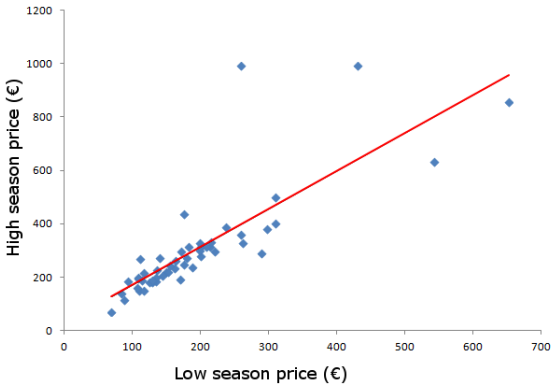


Figure 5.1: Linear regression of high season prices w.r.t. low season prices for 60 sample hotels.

5.1 shows the results of the linear regression. The low season price p_l and the high season price p_h can be linearly combined by $p_h = \beta_0 + \beta_1 p_l$, where $\beta_0 = +27,681$ and $\beta_1 = +1,4223$. The R^2 coefficient of the regression is equal to $0,6593$ (65.93%), which stands for a quite good fit. In fact the prices are well modelled by the regression line, apart from some outliers at very high prices. The β_1 coefficient shows that hotel owners tend to rise prices of the about the $45 - 50\%$ in high season periods.
Hence, we decided to rise the prices of hotels in high season of the 50% in our application to simulate. We used the initial room prices set in Polivenus as low season prices.

5.1.3 PoliVenus Extended

Before proceeding with the experiments, we have extended the functionalities of PoliVenus to handle our bounded condition. We had to modify both the user experience and the back-end of PoliVenus in order to make the experiments the more similar to the real conditions of limited availability of hotels in high season.

The Back-End

PoliVenus already handles all the procedures to call the recommendation algorithms and to present to users the recommendations in the user interface. The only modification needed in the back-end regards the handling of our first independent variable, namely the room availability, and the definition of the periods of high and low season.

To this purpose we have introduce some fields into the *test* table of PoliVenus database. In this table are specified all the characteristics of the task (i.e., the experimental condition) to be done by the users. The experimental conditions are chosen in Round-Robin fashion from active ones specified in such table, in order to have an uniform number of tests among all the participants to the study.
We report here all the fields of each task that can be customized (new added ones are followed by symbol *****):

Test identifer identifies the test. It is manually assigned and must be spaced from the other identifiers (this to impede users to change the test by modifying the URL string).

Kind of task is a string representing the scenario. In the original PoliVenus application it was possible to choose between two scenarios (i.e., work and holiday). In our case this field is useless since we consider only the holiday scenario.

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. This fields refers to a first naive simulation of hotels unavailability in PoliVenus. Also this field is useless in our case and we keep it set to 1 (i.e., available hotels).

Is last is a boolean which indicates whether the test is the last test performed by a user or not. It is used to control the Round Robin assignment of tests to users.

Is active indicates whether the test is considered active or not, i.e., if it has to been considered in the Round Robin assignation of tests.

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

Is high season* is a boolean value that establish whether the current test has to simulate high season availabilities. If set to 1, hotels from the

short-head are unavailable to users in the high season period. If set to 0, all the hotels are always available to users.

High season start* is a datetime field that sets the starting date of the high season period.

High season end* is a datetime field that sets the ending date of the high season period.

High season period is defined by the two respective variables in the table. When high season simulation is active, if the user tries to book an hotel into the high season period, he/she will not be able to book the best hotels (i.e., the ones in the short-head) and the remaining hotels will be presented with an increased price (i.e., the high season price). If the user searches out of this period, all the hotels become available at normal prices (i.e., the low season price). Users are forced to book a hotel within the high season period defined by the respective two variables, whatever the kind of availability (fully or bounded) is set by the experiment.

Other minor modifications were required to the back-end, especially to manage unavailable hotels into high season periods. However, such modifications do not add any new value to the capabilities of PoliVenus described in [20], so we do not report them in detail here.

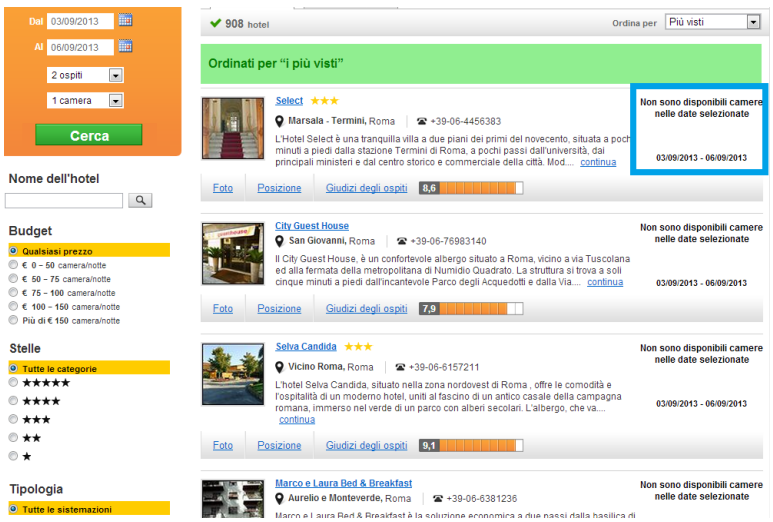


Figure 5.2: Unavailable hotels (in blue*) in PoliVenus User eXperience.*

The User Experience

More significant modifications were required on the original user experience of PoliVenus. We report here the most important ones.
Originally PoliVenus was not designed to permit the users to modify their own check-in and check-out dates. Moreover, the number of hosts was fixed to 2. We have enriched the user experience by allowing users to change the dates of stay and the number of hosts.
These are essential modifications in order to offer a more authentic experience to users. The capability of changing dates is also functional to our study. Users can now explore the whole catalog of hotels without setting their staying dates, as Venere.com does. When an user selects staying dates within the high season period may now find that many hotels are unavailable (Figure 5.2). This can also happen when the user changes the staying dates while watching the detail of an hotel which he/her is interested into (Figure 5.3). When the desired hotel is available in high season, its prices are increased of the 50% (Figure 5.4).

These details significantly increase the level of realism of the booking experience. Our intention was to create the same feeling of frustration and disappointment that tourists feel when they are not able to find an hotel that respects their own tastes and economic constraints. This surely has a significant impact on the overall the quality of the study.

All the other functionalities have not significantly changed from the original PoliVenus, especially regarding the way recommendations are inserted into the user experience and regarding the booking process. The only other difference regards the final survey we proposed to users, that is detailed in the following Section.

5.1.4 The Study Execution

As we have already told, our study considers both offline and online evaluation. In particular, online evaluation required an user experiment to be performed in order to assess the values of our quality metrics for each of the 6 experimental conditions reported in Table 5.1. Each participant was assigned to one experimental condition. Experimental conditions were assigned by the system in a Round Robin fashion in order to equally distribute the users among each case of our interest. The participants were asked to complete a task and then to complete a final survey. The detailed list of the questions in the survey are available in Appendix A.2.

	Room Availability	Recommendation Algo.
1	Low season	Editorial
2	Low season	Hybrid
3	Low season	TopRated
4	High season	Editorial
5	High season	Hybrid
6	High season	TopRated

Table 5.1: The six experimental conditions that were tested in our study.

Participant Recruiting

Participants were picked from current students, ex-alumni and administrative personnel of the School of Engineering and the School of Design of the Politecnico di Milano. They were all aged between 20 and 40 and have some familiarity with the use of the web and had never used Venere.com before the study. This to control the potentially confounding factor of biases or misconception derived from previous uses of the system. The total number of recruited subjects that completed the subject by the deadline was 142, equally distributed in the 6 experimental conditions.

The task

Users were asked to complete the following task:

"Imagine that you are planning a vacation in Rome and are looking for an accommodation during Easter season; choose a hotel and make a reservation for at most two 2 nights; dates and accommodation characteristics (stars, room type, services, and location) are at your discretion. After confirming the reservation (simulated), please complete the final questionnaire."

Easter is considered very high season in Rome. Hence, it is a very valuable period to the purposes of the study. Moreover, participants are left free to explore the accommodations in any desired day, with no restriction on the number of hosts and characteristics of the hotel. Only at the moment of the purchase the constraints of the task have to be fulfilled to complete the operation.

Reward Strategy

To increase the quality of the test we introduced a lottery incentive. We extracted randomly one participant who receives a prize. The prize consisted of a coupon of the value of 100€ to be used to stay in the hotel fictitiously reserved using PoliVenus. The impact of lottery incentives on survey re-

sponse rates have been exhaustively studied in [53]. We expect the users to make the test in a more realistic way.

5.2 Experimental Results

In this Section we present the experimental results of the study. As we have stated before, we have performed both system-centric and user-centric evaluation of the system. This to actually compare the computed quality and the perceived quality of the recommendations. Moreover, with user-centric evaluation we are able to differentiate the perceived quality over various aspects such as the overall satisfaction, trust and effort of users.

5.2.1 System-centric Evaluation

We have exhaustively talked about system-centric evaluation, or offline evaluation, and related metrics in Section 2.2.1. System-centric quality can be assessed using either error metrics (e.g., the RMSE and MEA) or accuracy metrics (e.g., precision, recall and fallout).

Interleaved hybridization does not compute the estimated ratings, but it combines the rankings of the two different recommendation algorithms. Hence, error metrics cannot be used. We therefore analyzed the offline quality using accuracy metrics. In particular we focused on the *recall* and *fallout* metrics.

Recall measures the capability of recommending interesting items to users. Fallout measures instead the attitude of the recommendation algorithm to recommended uninteresting items to users. Hence, we want to keep the recall the highest possible. Conversely, fallout should be the lowest possible to avoid recommending unappealing items to users.

To evaluate these measures we divide the User-item Rating Matrix of Polivenus into two sets, the *training* and the *test* set. The URM contains ratings extracted from Venere.com and Tripadvisor.com. Details over the URM are shown in Table 2.3 in Section 2.6.
Users in the URM were rearranged in random order and then assigned to either the training or the test set, in order to have an equal subdivision of the users in the initial URM. In other words, the training and the test set contain respectively *half* of the original users. The training set was used to train the 3 recommendation algorithms we chose (TopRated, Editorial and Hybrid). Then the accuracy metrics are measured against the test set using leave-one-out cross-validation. We compute accuracies for increasing values of N (i.e., the number of items presented to users by the RS), with values

in $(0, 100]$.
Notice that, in real applications, interesting values of N are often lower than 20, because users cannot actually handle with many recommended items at time. RSs are studied to help the user in the decision process, so providing many items is not very useful. The optimal value of N depends on many factors we do not investigate here.

When estimating the accuracies in high season availability, we removed from the test set all the hotels that belong to the short-head. However, recommendation algorithms are still trained over the original training set, which still contains the unavailable hotels and they are still able to recommend hotels in the short-head. This is what exactly occurs in real applications, where models do not take in account the possible variation over time of items availability. Hence, we impede the recommendation algorithms to recommend hotels in the short-head. This permits us to correctly estimate the metrics we are interested into.

Figure 5.5 shows the experimental results in conditions of *low season availability*, when all the hotels are available with no restriction. The plot of recall (Figure 5.5a) shows non-personalized algorithms actually outperform the personalized hybrid algorithm. For $N \leq 20$ the TopRated algorithm is clearly the best one, followed by the Editorial algorithm. Fallout (Figure 5.5b) shows no significant difference between algorithms. Consequently, al algorithms have almost the same capabilities in filtering uninteresting items.

Figure 5.6 shows the most interesting results. Now the recommendation algorithms are required to recommend hotels when the short-head is removed. By construction (see Section 2.2.4) the short-head contains the best hotels, namely the one with best popularity and average rating. These items cannot be recommended anymore since they are not available to users. The plot of recall (Figure 5.6a) shows that now non-personalized algorithms significantly down-perform with respect to the condition of full availability. In particular, TopPop algorithm is not able to provide any useful recommendation at all for $N \leq 20$. Straightforwardly, TopPop cannot recommend any hotel when the best ones are removed, because it simply recommends the top popular ones.
Instead, the Editorial algorithm is not entirely based on popularity, thus it can recommend some interesting item also in conditions of limited availability.

The hybrid algorithm has the best accuracy. Moreover, it does not significantly change it performances from the previous case. This means that the hybrid algorithm is *more robust* and *less sensitive* to our bounded con-

dition.The hybrid algorithm also has a greater fallout than others (Figure 5.6b). Still, when evaluating the Top-N Recommendation task using RSs, we are more interested in their capabilities of providing useful recommendation rather than excluding less interesting ones. Non-personalized algorithms are not able to provide useful recommendation with our bounded condition. Therefore, the Hybrid recommendation algorithm is the best performing one in Top-N Recommendation task in condition of limited availability of hotels, according to offline evaluation.

5.2.2 User-centric Evaluation

User-centric evaluation allows us to compare the perceived quality of recommendations in the different experimental conditions. We performed an user experiment (Session 5.1.4) and collected the data from 142 participants. We first removed the data referring to subjects who shown apparent evidences of gaming with the testing system. For example, users who interacted with the system for less than 2 minutes are pruned from the dataset, since we considered 2 minutes as the minimum time to complete the task. Similarly, users who left too many questions unanswered are not considered reliable and thus pruned.

In the end, we considered data referring to 125 participants, almost equally distributed in the six experimental conditions. Each experimental condition involves a number of subjects between 20 and 24.

We performed ANalisis Of VAriance (ANOVA) [22] over the 6 experimental conditions, which depends on the type of algorithm and on the hotel availability. ANOVA returns a $p\_value < 0.05$, which confirms us that the independent variables have a *statistically significant* impact over the dependent ones (subjective and objective).

We are now interested into finding relationships between dependent variables in the different experimental condition. To this scope, we performed a multiple pairwise comparison post-hoc test using Turkey's method on the *mean values* of the dependent values. We report here the results together with the 95% confidence intervals.

Figure 5.7 shows the values of the user satisfaction in all the experimental conditions. In low season scenario (Figure 5.7a) top popular non-personalized recommendations make users the most satisfied, with more than the 90% of users happy of their choice. Editorial recommendations differers significantly in perceived satisfaction from popular ones, with about the 60% of satisfied users.
In high season scenario users are overall less satisfied than in low season

in all the 3 experimental sub-conditions. Due to the scarcity of hotels in high season, users may impute the unavailability of hotels to a weakness of the catalogue of services. Hence, they can ascribe the phenomenon to the service provider rather than to the contingent situation of booking in a high season period. Non-personalized algorithms are now not able to provide interesting items to users, because the best one are gone, and the percentage of satisfied users almost halves: from 60% to 30% in the editorial case, and from 90% to 45%. In contrast, the hybrid algorithm now makes its users the most satisfied. About the 70% of users which received personalized recommendations are satisfied, and their number does not significantly differ from the low season scenario.

Figure 5.8 shows the histograms for the average price per night of the hotels reserved by the users. Differently from user satisfaction, price is an objective variable. In non-personalized conditions there is *statistically significant negative correlation* between hotel price and user satisfaction. In the low season scenario (Figure 5.8a) users who received popular recommendations are more satisfied and pay less, on average, than users in the editorial scenario, our baseline scenario. About 150€ per night in the editorial scenario, against about 100€ with top popular recommendations. Again, no statistically significant difference emerges when users are recommended through the hybrid algorithm.
In high season the average price per night increases by construction (Section 5.1.2). In this scenario the average price increases by more then 70% in the editorial condition and approximately 50% in the top popular one. Still, when the personalized algorithm is used, no statistically significant difference can be advised in the average price (Figure 5.8b).

Figure 5.9 shows the histograms for the average elapsed time, which corresponds to task execution time. This data confirms that searching for hotels during the low season period takes shorter time that in the high season period. In high season most of the best hotels are unavailable, and the entire decision making process is more complex and requires more time.

A part from this intuitive and predictable behavior, a more interesting fact is the following: the most satisfied users invested more time on the decision process than less satisfied users. This fact is consistent with the findings on the effort measured with the average number of explored hotels (Figure 5.10). Figures 5.9a and 5.10a show that users who received top popular recommendations in low season are the most satisfied, explored the largest number of pages and required the largest time to complete the task. The same happens in high season to users who received personalized recommendations (Figure 5.9b and 5.10b).

Figure 5.11 shows the overall trust over the booked hotel by users, i.e, how much they believe the hotel characteristics matches its description. Interestingly, in both the scenarios the most satisfied used totally trust the system. Trust significantly drops from low to high season for the users recommended with the editorial algorithm.

5.2.3 Discussion over the Study

We have presented before the experimental results of system-centric and user-centric evaluation of the system in the different 6 experimental conditions. We now explain a possible interpretation of this results, also respect to the results obtained in previous studies related to this argument [13, 14, 15]. We have previously discussed of some of them in Section 2.2.3.

System-centric evaluation partially confirms the results of previous works. The performance of non-personalized popularity algorithms in the Top-N Recommendation task is higher than the hybrid personalized algorithm, which is clearly more sophisticated than the TopPop and Editorial algorithm.

When the best items, namely the most popular and widely ranked items, are removed (the high season scenario) the accuracy of non-personalized algorithms decreases. The best algorithm becomes the most accurate and its accuracy does not significantly changes from one scenario to the other. Differently from [15], the effect of removing the short-head is *negligible* on the personalized algorithms.

The robustness of the hybrid recommendation algorithm can be explained considering the partial content-based nature of the algorithm. The strong effect of the short head removal reported in [15] is measured on collaborative algorithms, which tend to exhibit a bias toward popular items. In contrast, content-based algorithms, that do not base their models and recommendations on community ratings, do not exhibit such bias. When best items in the short-head are removed, biases are removed for the collaborative portion of the hybrid algorithm of the hybrid algorithm only, while the content-bases portion is only marginally affected. The overall effect of missing items on accuracy of is marginal. This fact can explain the mismatch with respect to prior studies.

Other significant differences between the low season and high season scenarios emerge from user-centric evaluation. The low season condition is comparable to the situation of *potentially unlimited capacity*, when all the items are always available. This condition characterizes most domains considered by recommender systems research, such as the movie domain.

Thus, it is not surprising that results on user satisfaction in the low season scenario are in line with prior findings in the movie domain [16]. They show that the perceived quality of non-personalized algorithm is comparable to the one of personalized algorithms. One of the possible interpretations of this phenomenon consistent with our results is the following:

> *the opinion of the crowd has a strong persuasion effect, often time higher that individual unexpressed preferences.*

In case of fully availability, there is a large amount of products potentially satisfying the characteristics specified by the user (e.g., stars, services, location). For example, Rome has tens of hotels with more than 3 stars next to the most touristic zones of the city. When the users have to choose over many interesting items, the most important attributes that drive the decision process are the number of ratings and their value. Trivially, most users are biased by *success*, therefore algorithms based on popularity are perceived as the best ones.

Non-personalized algorithms are based on a First Order Persuasion criteria (FOP), i.e., the *product success*. More sophisticated personalized algorithms, are based on a Second Order Persuasion criteria (SOP), i.e., *personalization*. When there is no limitation on item availability, FOP prevails on SEP. Personalized recommendations are not appreciated and less trusted because they do not match the opinion of the crowd. This mismatch can be easily noticed by users by looking at the popularity and ratings of the recommended items.

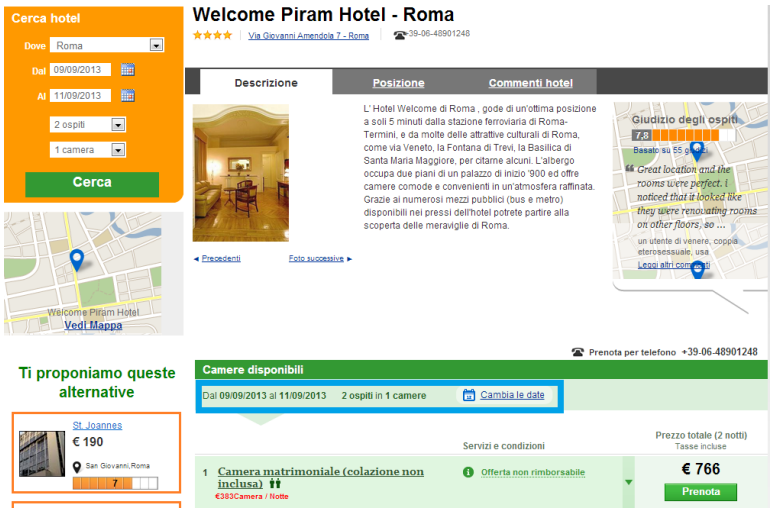
We have already talked about the peculiarities of the hotel domain in Section 2.2.3. We know that novelty does not have a great value in the hotel domain, because users tend to book the same hotel in the past if it offered a satisfactory experience. Hence, when this accommodation is available, recommendations of something new or unexpected might not be taken in account, regardless its adherence with personal tastes. This justifies the better perceived quality of non-personalized algorithms in the low season scenario.

In high season, when the best hotels are not available, personalized recommendations are the most effective. In conditions of scarcity of offer the most popular hotels are not available because the short-head has been removed. Available hotels now have no or few user ratings (Figure 4.4) and they are now indistinguishable from one another, with respect to popularity and average rating that act as persuasion factors when the offer is abundant. Users in these conditions are less biased by popularity and popularity-based algorithms are now perceived as less effective. SOP now takes the place of

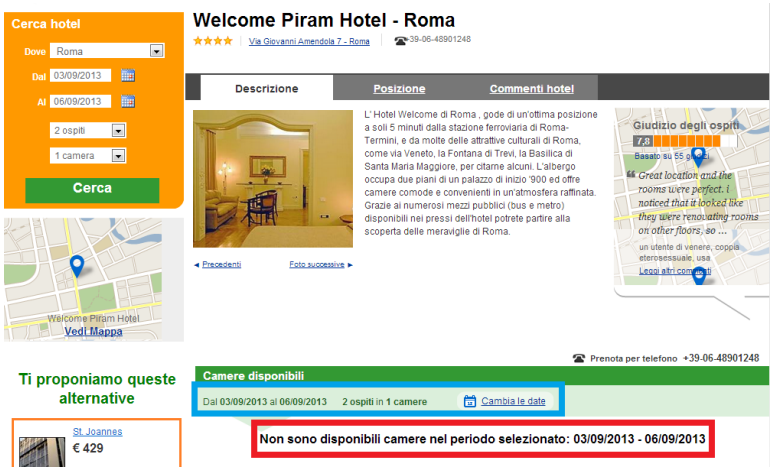
FOP, since it provides an acceptable match between item characteristics and personal needs. Personalized algorithms are unbiased by popularity and help users in finding alternative and novel by yet satisfactory solutions. Hence, in conditions of bounded availability of items, their persuasion strength increases.

In high season trust and effort, measured through variables elapsed time and extension of the hotel search, increase and reach the top level with personalized recommendations, while they decrease with popularity-based recommendations. When the best and most obvious solutions are not available, users are forced to spend more effort in searching for items and exploring information related to the choice process. But this burden is mitigated by benefits of a more satisfying and trustworthy decision process.

Finally, the analysis of the price per night of the booked hotel provides other interesting results. It is intuitive that higher levels of choice satisfaction are related to lower price of the chosen hotel. What is more surprising, but is consistent with the above analysis and the findings on effort, is that the average cost of reserved hotels in the condition of personalized recommendation is not affected by the scarcity of offer, remaining stable in low and high season. By effect of personalized recommendations, users tend to explore more items, become more conscious of alternative offers, and seem to be more able to discover hotels at reasonable prices.



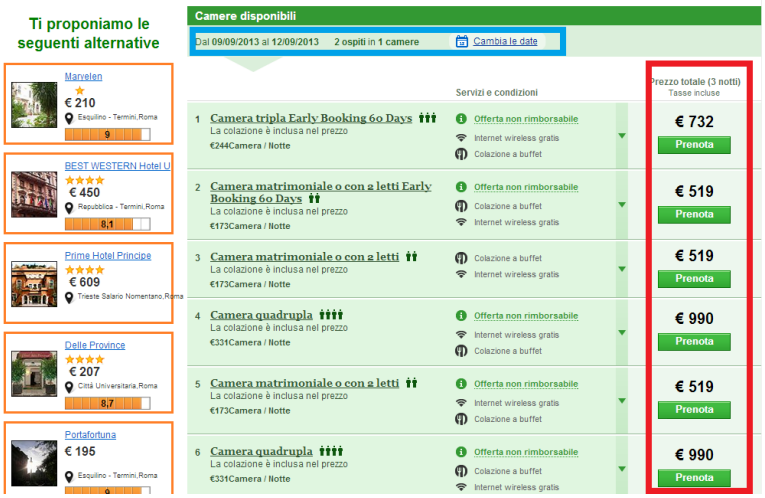
(a) Low season.



(b) High season.

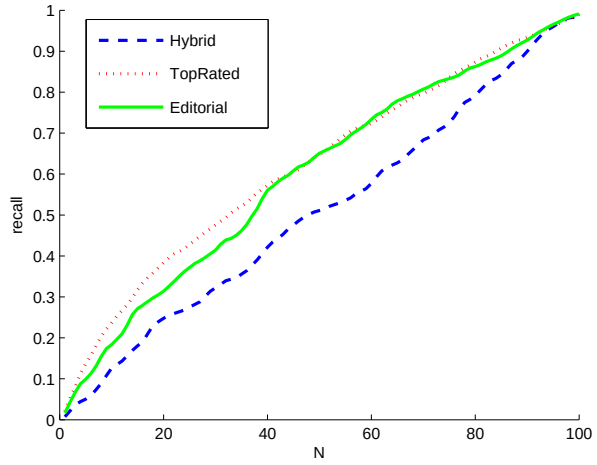
Figure 5.3: Simulated room unavailability (in red*) when the user is exploring the details of an hotel and passes from low to high season and viceversa (in* blue*).*

(a) Low season.

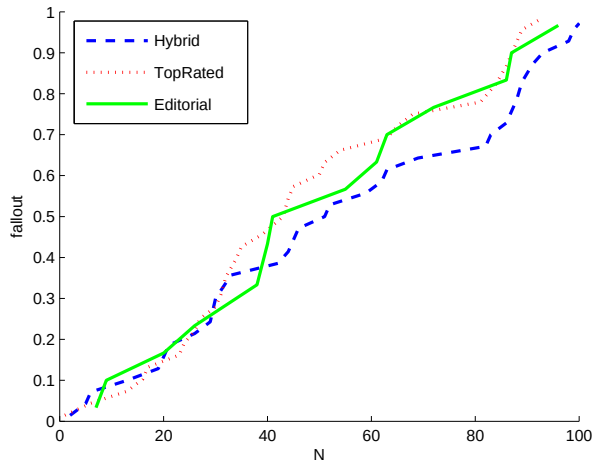


(b) High season.

Figure 5.4: Simulated price (in red*) change when the user is exploring the details of an hotel and passes from low to high season a nd viceversa (in* blue*).*

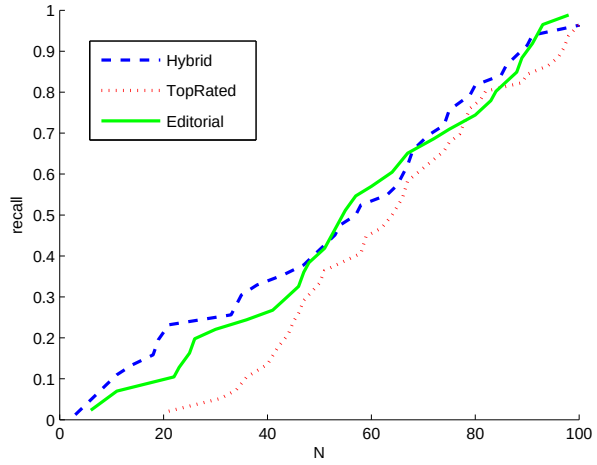


(a) Recall.

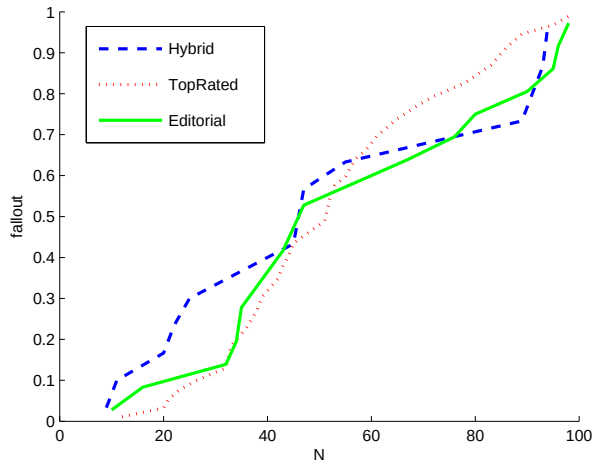


(b) Fallout.

Figure 5.5: Plots of the Top-N Recommendation recall and fallout in case of low season *availability (or fully availability).*

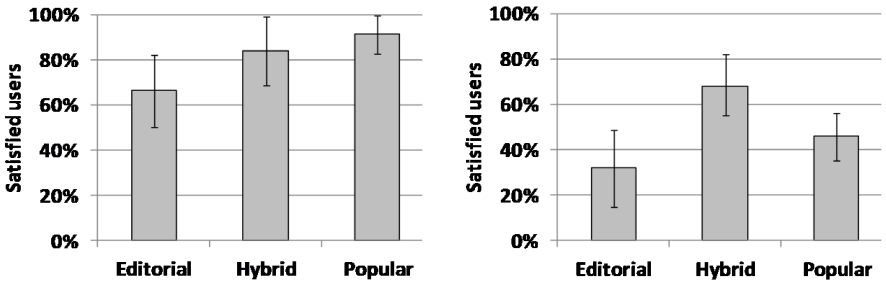


(a) Recall.



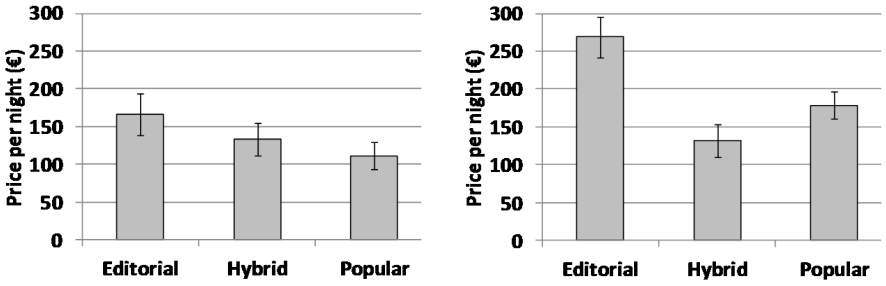
(b) Fallout.

Figure 5.6: Plots of the Top-N Recommendation recall and fallout in case of high season *availability (or limited availability).*



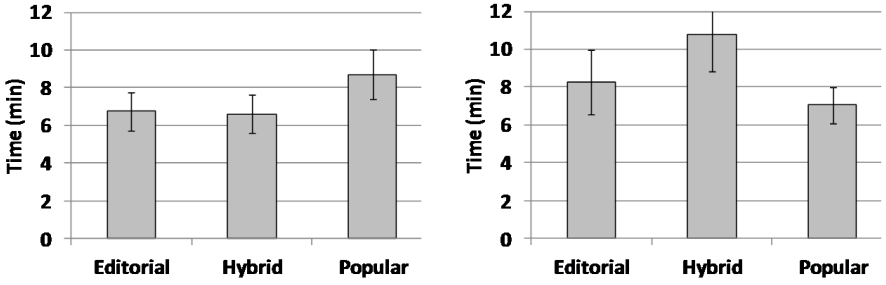
(a) Low season. *(b) High season.*

Figure 5.7: Histograms of the average user satisfaction in the 6 experimental conditions (together with 95% confidence intervals).



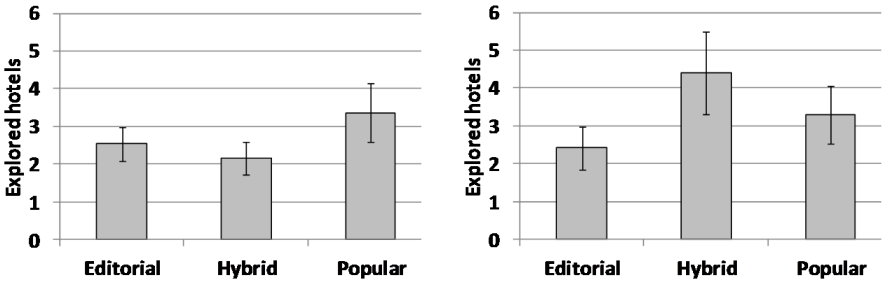
(a) Low season. *(b) High season.*

Figure 5.8: Histograms of the average price per night in the 6 experimental conditions (together with 95% confidence intervals).



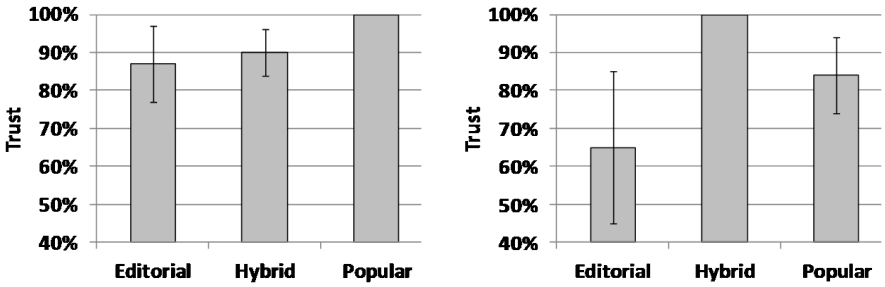
(a) Low season. *(b) High season.*

Figure 5.9: Histograms of the average elapsed time in the 6 experimental conditions (together with 95% confidence intervals).



(a) Low season. *(b) High season.*

Figure 5.10: Histograms of the average number of explored hotels in the 6 experimental conditions (together with 95% confidence intervals).



(a) Low season. *(b) High season.*

Figure 5.11: Histograms of the average percentage of user trusting the booked hotel in the 6 experimental conditions (together with 95% confidence intervals).

Chapter 6

Opinion Summarization in PoliVenus

In e-commerce items often come with many reviews that express opinions over the characteristics of the items. In the e-tourism domain, users often leave a comment on the quality of the services of the hotel and they express their feelings over the overall quality of their stay. Such information can be many times more informative than the mere rating that a user express on a hotel. In fact, peer reviews can have great influence on the decision making process. When an user has to decide the hotel to book, he/she may prefer the hotel that has good reviews over the aspects he/she is more interested on.

However, opinionated reviews are really hard to handle, due to the intrinsic difficulties of natural language. Users can express their opinion using different languages, idioms and, e.g., using irony, which is really hard to model and detect by automated systems. Nonetheless, many useful information can be extracted from free text to model the opinions that are expressed in.

PoliVenus comes with many reviews extracted from Venere.com and Tripadvisor.com. Until now, reviews were only used to extract features for content-based algorithm. No study on the opinions expressed by reviewers was performed.

In this Chapter we present a possible application of sentiment analysis on reviews in PoliVenus. We applied feature-based opinion mining on these reviews. Then we create a model to predict user ratings using opinion scores. In this way, one can assign a numerical rating to users who have written a review without any explicit rating, and consequently to reduce the sparsity of the User-item Rating Matrix.

Language	Reviews
English	73348
Italian	9549
Total	82897

Table 6.1: Subdivision of the reviews from Venere.com according to language.

We detail here the proposed methodology to summarize user reviews in PoliVenus. We present the details of the implemented system and its performances.

6.1 The Proposed Methodology

PoliVenus comes with many reviews extracted from Venere.com and Tripadvisor.com. These reviews were only used to create the model for content-based algorithms using the TF-IDF schema, but no further investigation on the information they contain was performed originally. More precisely, there are available 82897 reviews from Venere.com and 97831 reviews from Tripadvisor.com. Reviews from Venere.com are both in English and Italian (Table 6.1), while only Italian reviews were crawled from Tripdavisor.com at the time of construction of PoliVenus. This constitutes a huge limitation, since almost all the solutions in Natural Language Processing and Opinion Mining consider English language. Consequently, we were able to process only the English fraction of reviews from Venere.com, that is composed of about 70K reviews. Still, this is a sufficient quantity of information to model opinions in hotel reviews, as we will see later.

The extent of our study is to model efficiently the opinions expressed in reviews in order to predict user ratings over the reviewed hotel. Each review is processed to extract the opinion expressed over a set of homogeneous features. Then a model based on such features is trained against the ratings in the User-item Rating Matrix. Our approach can be subdivided in different steps as follows:

1. *Language detection* to distinguish reviews' language and process only English reviews.

2. *Feature-based opinion mining* to create an homogeneous set of features to be used in opinion extraction.

3. *Opinion aggregation* to compute the dominant opinions expressed by each user over each item he/she reviewed.

4. *Model construction* to construct an precise model for summarization purposes.

5. *Rating prediction* to summarize opinionated reviews with a single numerical rating.

6.1.1 Language detection

The first task to be performed is necessarily language detection. At the best of our knowledge, we are able to extract opinion only from English reviews. Therefore it is necessary to detect the language in which reviews are written to discard non-English ones. This is a huge limitation, especially if we consider that about the 60% of the reviews we have are written in Italian. However, English guarantees the generality of our approach, since it is currently the most used language on the Web[1].
There exists various tools that can automatically translate from Italian text to English(e.g., Google Translate[2] and Bing Translator[3]); however, many of these instruments are not for free. Moreover, automatic translation is not an exact process and can act as source of noise in our environment. Therefore, we decided to consider only native-English reviews in our analysis, to keep it the more clean and effective possible.

Several techniques based on statistical analysis of text are available, based on, e.g., language trees [6], language n-grams models [10] or function words [25]. In our work we used the Java library LanguageDetection[4], which uses a Naïve Bayes model with character n-grams to detect 49 languages with more than 99% of accuracy.

6.1.2 Feature-based Sentiment Analysis

Opinionated text has to be processed properly to extract the opinions expressed the reviewer. We have already introduced sentiment analysis, or opinion mining, in Section 2.5. Sentiment analysis is the computational study of opinions, sentiments and emotions expressed in text. In our study we adopted the feature-based approach to sentiment analysis we described in Section 2.5.3. Hotel reviews, such as many other domains as movies or books, tend to express opinions over the same set of aspects and qualities of an hotel. For example, reviewers often express their feelings over the quality

[1]Source: http://www.internetworldstats.com/stats7.htm
[2]https://developers.google.com/translate/?hl=it
[3]http://www.bing.com/translator
[4]https://code.google.com/p/language-detection/

of the services of an hotel (e.g., air conditioning, breakfast, internet connection), or over its location and prices. Hence, hotel reviews are really suitable for frequent feature extraction process. Here we report some reviews taken from Venere.com dataset:

> "A convenient hotel for a night before departure or onward journey. Typical chain hotel lacking the personal touch, but run efficiently up to a good standard. The receptionist was friendly and polite."
>
> "We loved the hotel, and the location was close to the main area of Bologna. However, the street it was on was very busy and industrial without any restaurants or anything really. That being said, once you stepped into the hotel, it was amazing, and we loved it so much that we stayed an extra night! Really quiet, and comfortable."
>
> "Very good hotel on the whole; a bit far from the centre but there is a good bus service within walking distance. Could do a bit better as far as cleanliness in the rooms is concerned. Nothing offputting though."

It can be easily seen that the vocabulary of the three tends to converge to a common set of opinionated features, such as *hotel*, *receptionist*, *location*, *rooms* and *centre*. By extracting this common vocabulary we are able to identify what are the aspects that really interest hotel reviewers. This common set of variables, together with their opinion values, can be used to train a comprehensive model of the opinions expressed in all the reviews.

However, opinions are expressed in different ways, so a robust opinion mining application is needed to perform this kind of operation. Hence, we decided to adopt the approach to feature-based sentiment analysis described in [19]. This approach is exactly what we described in Section 2.5.3, and it is reported to achieve very interesting performances in opinion extraction from reviews over several different products, e.g., cameras and mp3 players. It describes the extension of a prior opinion mining tool, called Opinion Observer [38], with new functionalities in recognizing context-dependent opinions. They used a lexicon-based approach to detect orientations of opinion words in opinionated text associated to a frequent and infrequent features. The detailed pseudo-code of the application is reported in Appendix A.3.

With the explosion of social networks many companies started to become interested on the opinion over their products, and sentiment analysis have became a critical tool in mining the enormous quantity of user-generated

content available on such networks. Research in opinion summarization and mining have taken great advantage from this increasing interest of great companies in this field. On the other hands, many the best algorithms that have been developed are not freely available due to their big economic potential, and are the base of many startups. Opinion Observer and its extensions are no exception. Therefore, we needed to totally re-implement the entire application to perform our analysis. Only the opinion lexicon used by the authors is open-access[5].

Regarding feature-based opinion analysis, there are some observations that can be made by taking in account our domain of interest. In general, the mining task consists of discovering any quintuple $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$ and identifying all the synonyms W_{jk} and feature indicators I_{jk} of feature f_{jk}.
In the hotel domain, the object and the opinion holder can be deduced easily from the review. The object o_j is the reviewed hotel i and the opinion holder h_i is the user u who made the review.
We are interested in time t_l, because a user u can express his/her opinion over hotel i more than one time in different sentences or by writing different reviews.
We will consider only frequent features, since we are interested in building an homogeneous set of features in common with all the reviews. Infrequent features are likely to appear in very few sentences and, although they can bring useful information over the less common aspects of an hotel, they cannot be used to compare a reviews with each other in a consistent and coherent way.

Consequently, the original feature-based opinion mining problem can be reduced to the following:
Given a set of reviews $V = \{v_{uil}\}$ written by user u over feature i at time t_l, we want to discover a set of features $F = \{f_1, f_2, \ldots, f_m\}$ frequent in all the reviews in V. A feature f_k is frequent if it occurs in more than the 1% of reviews in V, that is, if it has a support $sup(f_k) \geq 0.01\|V\|$.
Since a frequent feature f_k may not be opinionated in review v_{uil}, we introduced a new binary parameter p_{kuil}, that we called *feature presence*:

$$p_{kuil} = \begin{cases} +1 & \text{if review } v_{uil} \text{ contains feature} f_k \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

We want to discover any quadruple $(f_k, p_{kuil}, oo_{kuil}, t_l)$; if feature f_k is present in review v_{uil} (i.e., $p_{kuil} = 1$), then its quadruple contains the orien-

[5]http://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html

tation oo_{uikl} of the opinion expressed by user u and hotel i over feature f_k at time t_l. If feature f_k it is not present in review v_{uil} (i.e., $p_{k_uil} = 0$), then its opinion orientation is meaningless and set to 0.

6.1.3 Opinion Aggregation

The result of feature-based opinion mining is a set of orientation values for each feature present in each sentence of a review that we formalized in the previous Section. Orientations are expressed in quadruples $(f_k, p_{kuil}, oo_{kuil}, t_l)$ and are still dependent on time t_l. Orientations should be then aggregated in order to obtain the opinion of the user over the subset of features he/she is interested in regarding a specific hotel.
More formally, we are interested in obtaining all triples (f_k, p_{kui}, oo_{kui}) which express the overall opinion orientation oo_{kui} of user u and hotel i over all the occurring features f_k (i.e., $p_{kui} = 1$), independently from time t_l.

There exist two possible cases in which a user u express its opinion on feature f_k for hotel i, i.e., $\exists l \neq m | p_{kuil} = 1 \wedge p_{kuim} = 1$: the user express its opinion over feature f_k in different sentences into the same review, or in separate reviews. More specifically, these are the special cases we have to handle:

1. A feature f_k can have different orientation in different sentences of the same review.

2. An user can review the same hotel more than one time (e.g., after he/she has returned in the same hotel he booked previously, and he/she wants to leave another comment on it).

In both cases a feature can be associated to more than one orientation score. To correctly handle these cases, we associate to each feature its *dominant orientation*, i.e., the orientation score that occurs more times for a specific feature f_k. Given the set of quadruples $(f_k, p_{kuil}, oo_{kuil}, t_l)$, for each feature f_k we fist compute the sum of its orientations over all time L:

$$\begin{aligned} s_{kui} &= \sum_{l \in L} p_{kuil} \cdot oo_{kuil} \\ &= \sum_{l \in L} oo_{kuil}, \end{aligned} \tag{6.2}$$

since $oo_{kuil} = 0$ when $p_{kuil} = 0$.
Then the dominant orientation $\hat{o}_{kui}$ of feature f_k expressed by user u over

hotel i is computed as follows:

$$\hat{o}_{kui} = \begin{cases} min(s_{kui}, +1) & \text{if } s_{kui} > 0, \\ max(s_{kui}, -1) & \text{if } s_{kui} < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

For example, consider the following two reviews, written by the same user over the same review:

> "**Location[+1]** perfectly central. *Room*[+1] and **hotel[+1]** of a high standard. **Staff[+1]** excellent. However, the **air conditioning[-1]** was not working at all. We'll be coming back."

> "We returned to the same **hotel[+1]** as last year - so obviously we like it! Great **location[+1]** at the end of the Ponte Vechio. Quiet dispite being very central. **Staff[+1]** very helpful in (amoungst other things)providing an early *breakfast*[0] on marathon morning! And now **air conditioning[+1]** was renewed and perfectly working!!!"

We typed in bold font the features in common between the two reviews. Straightforwardly, the dominant orientations of features *hotel*, *location* and *staff* is positive, with score +1. One may expect that opinions of users over the same hotel should be consistent in every review. But this should not be the fact, because the offer of services by hotels may vary over time, and so the opinions of its hosts. That is the case for feature *air conditioning*, which will be assigned to a dominant neutral opinion (score 0). Using dominant opinion we are able to catch the overall feeling of the user over hotel features, from oldest to newest ones.

6.1.4 Model Construction and Rating Prediction

After feature-opinion mining and opinion aggregation we have a set of triples $(f_k, p_{kui}, \hat{o}_{kui})$. For each user u, hotel i and feature f_k, each triple tells us whether f_k is opinionated, p_{kui}, and its dominat opinion orientation, $\hat{o}_{kui}$.

The User-item Rating Matrix $R = \{r_{ui}\}$ contains the ratings expressed by user u on hotel i. The rating $r_{ui} = 0$ if the user u has not rated hotel i. Otherwise, we consider ratings $r_{ui} \in \{1, 2, \ldots, 5\}$. Our discussion can be straightforwardly extended to other rating scales using translations.
We want to model ratings $r_{ui} \in R$ using feature opinion orientations. Hence,

each rating r_{ui} can be expressed as follows:

$$\begin{aligned} r_{ui} &= \sum_{k,u,i} \alpha_{kui} \cdot p_{kui} + \sum_{k,u,i} \beta_{kui} \cdot \hat{o}_{kui} \\ &= \sum_{k,u,i} (\alpha_{kui} \cdot p_{kui} + \beta_{kui} \cdot \hat{o}_{kui}) \end{aligned} \tag{6.4}$$

This equation expresses the rating r_{ui} as the linear combination of two variables, namely the feature presence and feature orientation. Their effects are considered separately and weighted differently through two weighting matrices $\mathbf{A} = [\alpha_{kui}]$ and $\mathbf{B} = [\beta_{kui}]$.

If we group the effects of feature presence and feature orientation, we obtain the following simplified model:

$$r_{ui} = \sum_{k,u,i} \alpha_{kui} \cdot p_{kui} \cdot \hat{o}_{kui} \tag{6.5}$$

where the weighting matrix $\mathbf{A} = [\alpha_{kui}]$ now weights both feature presence and opinion orientation. Moreover, since $\hat{o}_{kui} = 0$ when $p_{kui} = 0$, the (6.5) can be further simplified to the following:

$$r_{ui} = \sum_{k,u,i} \alpha_{kui} \cdot \hat{o}_{kui} \tag{6.6}$$

In this way, ratings are estimated taking in account only the opinion orientation of feature f_k for user u and item k, regardless of feature presence. Effectively, opinion orientation includes the information on feature presence when $\hat{o}_{kui} \neq 0$. In case of neutral opinion $\hat{o}_{kui} = 0$, we lost the information about the presence of the feature f_k and the model cannot distinguish whether the user expressed a neutral opinion on the feature, or if he/she did not considered it at all. However, we considered this loss of information less important than the opportunity of managing a more simplified model.

The weighting matrix $\mathbf{A}$ is a tridimensional matrix of size $\|F\| \times \|U\| \times \|I\|$, where F is the set of features, U is the set of users and I is the set of items. This weighting matrix is the real source of complexity in the model, since it can be of very huge dimensions in real applications and very hard to compute. To reduce its complexity, we introduce the following two assumptions:

1. The importance of a specific feature f_k is *independent* on user u, i.e., the weights of feature f_k are not related with user u.

2. The importance of a specific feature f_k is *independent* on item i, i.e., the weights of feature f_k are not related with item i.

With these assumptions the model can be reduced to:

$$r_{ui} = \sum_k \alpha_k \cdot \hat{o}_{kui} \tag{6.7}$$

where $\boldsymbol{\alpha} = [\alpha_k]$ is contains the weights of every frequent feature $f_k \in F$. The (6.7) can be transcribed as follows:

$$\mathbf{R} = \boldsymbol{\alpha}^T \cdot \mathbf{O} \tag{6.8}$$

where $\mathbf{R}$ is the URM of size $\|U\| \times \|I\|$, $\boldsymbol{\alpha}$ is column vector of length $\|F\|$, and $\mathbf{O}$ is the opinion orientation matrix of size $\|F\| \times \|U\| \times \|I\|$.
The vector $\boldsymbol{\alpha}$ can be estimated by solving the following *least squares* problem:

$$\underset{\bar{\alpha}_k}{\operatorname{argmin}}(r_{ui} - \sum_k \bar{\alpha}_k \cdot \hat{o}_{kui})^2 \tag{6.9}$$

Once all weights $\bar{\alpha}_k$ that minimize the squared error have been computed, the *predicted rating* $\hat{r}_{ui}$ is computed as:

$$\hat{r}_{ui} = \sum_k \bar{\alpha}_k \cdot \hat{o}_{kui} \tag{6.10}$$

6.2 Experimental Results

The first phase of the work, and the largest time consuming, regarded the total re-implementation of the feature-based opinion mining application described in [19]. We needed to totally recode a Java application that implements all the characteristics of the application, a part from the opinionated lexicon that was available on the author's website.
We used several open source libraries to this purpose:

- Apache OpenNLP 1.5[6] and Apache Lucene 4.0.0[7] libraries for natural language processing operations like tokenization, stop-words filtering and stemming;
- JWNL 1.3[8] to access to WordNet 2.1 Dictionary[9];
- Simmetrics Library[10] that provides several string similarity metrics to be used to implement approximate string matching;

[6] http://opennlp.apache.org/
[7] http://lucene.apache.org/core/
[8] http://jwordnet.sourceforge.net/
[9] http://wordnet.princeton.edu
[10] http://sourceforge.net/projects/simmetrics/

- We extended the Apriori-Java Library[11] to compute frequent features in review sentences.

We tested our implementation on some of the test datasets used by the authors to confirm the validity of our implementation. We obtained very valuable results, although not equal due to the intrinsic difficulties in replicating exactly another technology. Moreover, each possible context requires an extensive study in order to identify possible context-depended opinion words, feature indicators and synonyms. Nonetheless, we were not interested into obtaining a perfect match between the original and our application, but more in obtaining a correctly tuned instrument to be used in the subsequent elaborations.

6.2.1 Training the Instrument

Feature-based opinion mining tool is a unsupervised learning instrument that needed to be properly trained in order to process correctly all the review in the dataset. We trained the feature-based opinion mining instrument over 10K randomly selected reviews (i.e., the *training set*) taken from the 73348 English reviews of Venere.com. The training process is necessary to extract frequent features, to identify synonyms and antonyms of opinion words, and to assign the orientation to context-dependent opinion words. This data will be subsequently used in the actual elaboration of all the reviews, as it can be seen from the pseudo-code of the application we reported in Appendix A.3.

Each sentence was extracted and considered as a transaction. We set the minimum support threshold to 1%, i.e., all the words and words and word phrases that occurs more at least in the 1% of the extracted sentences are considered as frequent features.
We successfully extracted 65 frequent features from the training set. They are completely reported in Table A.1 together with their support information. It can be seen that this process extracts several words and word phrases that are really meaningful for the tourism domain. For example, the most frequent words are *hotel*, *room*, *breakfast* and *staff*, which clearly are broadly used by reviewers and are very valuable features to describe hotels. Furthermore, also some interesting word phrases are automatically extracted, such as *train station* and *room clean*.
Some sample opinion words related to feature *location* is reported in Table 6.2. Opinion words are associated to each feature together to their

[11] code.google.com/p/apriori-java/

cumulative orientation value, i.e., the sum of the opinions collected over all sentences, and their respective dominant opinion. It is necessary to compute dominant opinion in handling context-dependent opinion words, for which opinion cannot be defined a-priori. It can be seen that the opinion words are clearly related with the feature *location*.

Opinion Words	**Cum. Orientation**	**Dom. Orientation**
limited	-1	-1
satisfied	13	+1
benefit	2	+1
thrilled	-1	-1
nice	54	+1
fortunately	1	+1
optimal	2	+1
strange	-1	-1
free	3	+1
excellent	10	+1
appealing	0	0
right	20	+1
enjoyable	1	+1

Table 6.2: Sample opinion words extracted for feature location*, together with their cumulative and dominant orientation.*

Once the opinion mining tool has been properly trained, we processed all the reviews in dataset to extract all the triples *(feature, feature presence, opinion orientation)*. For each sentence, all the features that occur are first extracted, and then their opinion orientation is computed. After that, for each review is computed the dominant orientation of all the features that have been previously mined. Finally, opinions are from reviews on the same hotel from the same user are aggregated.
In the end, we obtained the matrix **O** of opinion orientations we described in the previous section. The opinion aggregation step reduces the number of opinionated records from the initial 73348 to 43737 records, regarding the opinion expressed by 43154 users over 2297 hotels.
Table 6.4 reports some statistics over the opinion mining process. We can see that almost every user have expressed *one opinion*. This fact justifies the first assumption we made to simplify the model described in the previous Section, i.e., considering features as independent from users. In fact, if users tends to express only one opinion, there is no need to model the weight of features per user.
Hotels instead received in average 20 opinions each. However, the median

number of opinions per hotel is only 8. The box plot in Figure 6.1 shows that hotels with really many opinion records are outliers in the distribution, therefore also our second assumption, i.e., the independence between feature weights and hotels, still holds. This because it would be really difficult to create a statistically significant model of the importance of each feature per hotel with such poor base of opinion records, with consequently poor performances in rating prediction.

We provide here only an excerpt of the results we can obtain after the opinion mining elaboration. For example, consider the following review:

> "The *place* was *good* and the *breakfast* was excellent.
> My only complaint is that we had requested a non-smoking *room*
> and the room we got reeked of smoke so bad we had to leave the
> *window* open-in December."

The features that have been identified, together with their feature presence and opinion orientation values are the following:

Feature	**Presence**	**Orientation**
place	1	+1
good	1	+1
breakfast	1	+1
room	1	-1
window	1	-1

Table 6.3: Example of feature-based opinion mining.

It shows that features and opinions are well detected. In fact, the users express a favourably over the place and breakfast, and complains over the room. Moreover, it can be seen that feature presence is a redundant field if no neutral opinion is expressed by the user.

6.2.2 Model Training and Evaluation

After the opinion extraction processes, we have a sufficient amount of data to train the model we described in Section 6.1.4. We used Matlab R2009b for matrix processing, and its Statistics Toolbox to train the model and check the quality of predictions. Matlab provides several instruments to solve least squares problems and, in general, to train and test linear models.

We started from the matrix $\mathbf{O}$, a tridimensional sparse matrix of size $\|F\| \times \|U\| \times \|I\|$, where $\|F\| = 65$, $\|U\| = 43154$ and $\|U\| = 2297$. $\mathbf{O}$ contains $\|U\| \cdot \|I\|$ opinion orientation records computed in the previous step.

Processed reviews	73348
Aggregated opinions	43737
Users	43154
Hotels	2297
Max Opinions/User	10
Min Opinions/User	1
Avg. Opinions/User	1.098
Median Opinions/Users	1
Max Opinions/Hotel	563
Min Opinions/Hotel	1
Avg. Opinions/Hotel	20.619
Median Opinions/Hotel	8

Table 6.4: Statistics over the opinion mining processing (Opinions refers to the aggregated opinion record associated to a pair (user,hotel)).

Together we extracted the rating matrix $\mathbf{R}$ from the URM of PoliVenus, by picking the rating corresponding to each pair *(user,item)* in $\mathbf{O}$. $\mathbf{R}$ is also a sparse matrix with values in $[1, 5]$. The missing ratings are treated as zeros. We divided both matrices into two sets, namely the *training* and the *test* set. The training set comprises the 80% of randomly selected opinion records of $\mathbf{O}$ and their respective ratings in $\mathbf{R}$. The remaining 20% is used as test set.

We slightly modified the general model in (Equation 6.8) by adding a constant term α_0 to include constant variations in user ratings. The result is the following:

$$\begin{aligned} \mathbf{R} &= \alpha_0 + \boldsymbol{\alpha}^T \cdot \mathbf{O} \\ &= \hat{\boldsymbol{\alpha}}^T \cdot [1|\mathbf{O}] \end{aligned} \tag{6.11}$$

We computed the weighting vector $\hat{\boldsymbol{\alpha}}$ by solving the least squares problem in (6.9) over the training set. We then evaluate the accuracy of predictions using the (6.10). To assess the quality of predictions we computed the Root Mean Squared Error (RMSE) with 10-fold cross-validation over the test set. We obtained a RMSE of 0.6739.
The resulting weight vector, together with the 95% confidence intervals, is reported in table A.2. We have obtained appreciable performances in prediction, especially if we consider how noisy and hard to manage is the user-generated content we started from.

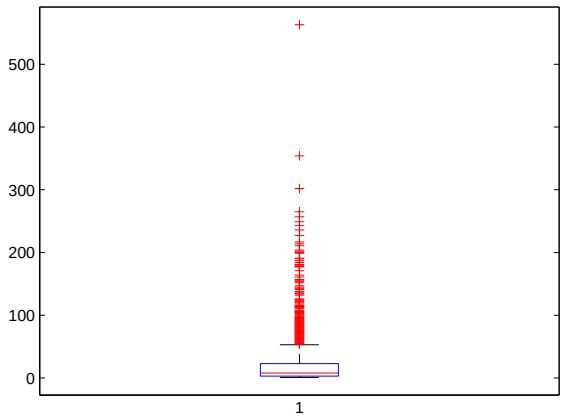


Figure 6.1: Boxplot of the distributions of opinion records per hotel.

Chapter 7

Conclusions and Future Work

In the previous chapters we have explained the characteristics of a RSs in suitable for application in the e-tourism domain. We have introduced an effective implicit elicitation method. We have accurately modeled the phenomenon of constrained availability of items and the variation of items availability across time. We have exhaustively studied the impact of bounded availability of hotels on the quality of recommendations. We performed both offline system-centric analysis and online user-centric analysis of the quality of recommendations through an accurate user experiment.
It results that personalized recommendations combined with implicit elicitation are able to provide very good recommendations in every experimental condition. Especially, they can mitigate the potentially negative effects of circumstantial scarcity of resources and to increase customers' trust.

This research can be useful to design recommendation algorithms optimized for "the best are gone" condition. This condition is of really practical interest, since there are many domains affected by such type of item consumption. The absence of the most interesting items can be seen by users as a weakness of the catalogue of offers. Users may ascribe this phenomenon to the provider, reduce their trust on it, and eventually leave the system in favour of a competitor. Providing personalized recommendations when many of the most interesting items are not available can really make the difference, keeping user satisfaction high also when the contingent situation makes the decision making process harder.

Moreover, we proposed a novel approach in user review summarization. We have used the English reviews available in PoliVenus to extract the opinions that users expressed on the most interesting features of every ho-

tel. We then computed a linear model to weight features according to the numerical rating given reviewers on hotels. We were able to achieve good performances in predicting the numerical rating starting from the opinion orientation records extracted from textual reviews.
This type of opinion summarization associates the qualitative set of opinions expressed in reviews to a single numerical rating that express the overall feeling of the user on the hotel. This potentially allows to enrich User-item Rating Matrices with new ratings generated from users' reviews, leading to more reliable results in collaborative recommendations.

There exist several more potential developments in the integration between RSs and user textual reviews. In the hotel domain users tends to review the same hotel more times, hence some temporal decay schema can be introduce to give greater importance to newer reviews than to older ones in computing dominant opinions, since the most recent reviews may reflect better the actual feelings of the user on hotel qualities.
Feature-based opinion mining provides detailed information on the opinion that each user have on a selected subset of interesting features. By opinion summarization we lost this detailed information in favour of a more compact numerical rating. Consequently, it would be advisable to use feature opinions in an mixed collaborative/content-based algorithm, which integrates hotel features with the information over the opinions expressed by users on these features. Such kind of opinion-based recommendation algorithm will be aware of user interests and tastes, and can recommend him/her items similar to what he liked in the past and on which other users expressed favourable opinions.
Finally, it is well known that numerical user ratings are often biased by the systematic tendency for some group of users to rate higher then others. Each user has a different perception of the value subsumed by every numerical value, and this has great influence on the ratings they give to items. But in textual reviews users can freely express their opinion on items without being constrained by a rigid numerical scale. Therefore, the opinions contained in reviews can potentially be more reliable than the mere numerical rating. Consequently, by adopting a different opinion summarization schema, we are able to create more reliable URMs and, thus, to provide more effective recommendations.

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] Chris Anderson. *The long tail: Why the future of business is selling less of more*. Hyperion Books, 2008.

[4] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

[5] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender Systems Handbook*, pages 299–331. Springer, 2011.

[6] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. Language trees and zipping. *Physical Review Letters*, 88(4):048702, 2002.

[7] Michael W Berry. Large-scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

[8] Robin Burke. Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce*, pages 69–72, 1999.

[9] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[10] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[11] Eugene Charniak. Statistical techniques for natural language parsing. *AI magazine*, 18(4):33, 1997.

[12] Lorcan Coyle and Pádraig Cunningham. Improving recommendation ranking by learning personal feature weights. In *Advances in Case-Based Reasoning*, pages 560–572. Springer, 2004.

[13] Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Vittorio Papadopoulos, and Roberto Turrin. Looking for good recommendations: A comparative evaluation of recommender systems. In *Human-Computer Interaction–INTERACT 2011*, pages 152–168. Springer, 2011.

[14] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. User-centric vs. system-centric evaluation of recommender systems. In *Human-Computer Interaction–INTERACT 2013*, pages 334–351. Springer, 2013.

[15] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[16] Paolo Cremonesi and Roberto Turrin. Time-evolution of iptv recommender systems. In *Proceedings of the 8th international interactive conference on Interactive TV&Video*, pages 105–114. ACM, 2010.

[17] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

[18] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[19] Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the international conference on Web search and web data mining*, pages 231–240. ACM, 2008.

[20] Antonio Donatacci. Exploring recommender systems for decision-making in e-tourism. Master's thesis, Politecnico di Milano, 2012.

[21] Sara Drenner, Shilad Sen, and Loren Terveen. Crafting the initial user experience to achieve community goals. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 187–194. ACM, 2008.

[22] Andy P. Field. *Analysis of Variance (ANOVA)*, pages 33–36. SAGE Publications, Inc., 0 edition, 2007.

[23] George W Furnas, Scott Deerwester, Susan T Dumais, Thomas K Landauer, Richard A Harshman, Lynn A Streeter, and Karen E Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 465–480. ACM, 1988.

[24] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1805–1808. ACM, 2010.

[25] Gregory Grefenstette. Comparing two language identification schemes. In *Proceedings of the 3rd International Conference on Statistical Analysis of Textual Data (JADT 95)*, pages 263–268, 1995.

[26] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[27] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760, 2004.

[28] Dietmar Jannach, Markus Zanker, Markus Jessenitschnig, and Oskar Seidler. Developing a conversational travel advisor with advisor suite. *Information and Communication Technologies in Tourism 2007*, pages 43–52, 2007.

[29] Nicholas Jones and Pearl Pu. User technology adoption issues in recommender systems. *Proceedings of NAEC, ATSMA*, pages 379–39, 2007.

[30] H Kim, Kavita Ganesan, PARIKSHIT Sondhi, and CHENGXIANG Zhai. Comprehensive review of opinion summarization. *Illinois Environment for Access to Learning and Scholarship, Tech. Rep*, 2011.

[31] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann, 1995.

[32] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[33] Antonis Koukourikos, Giannis Stoitsis, and Pythagoras Karampiperis. Sentiment analysis: A tool for rating attribution to content in recommender systems. In *Proceedings of the 2nd Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2012). Manouselis, N., Drachsler, H., Verbert, K., and Santos, OC (Eds.). Published by CEUR Workshop Proceedings*, volume 896, pages 61–70, 2012.

[34] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 115–122. ACM, 2012.

[35] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[36] Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer, 2007.

[37] Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:568, 2010.

[38] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

[39] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[40] Pasquale Lops, Marco Degemmis, and Giovanni Semeraro. Improving social filtering techniques through wordnet-based user profiles. In *User Modeling 2007*, pages 268–277. Springer, 2007.

[41] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM, 2009.

[42] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[43] Benjamin M Marlin and Richard S Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12. ACM, 2009.

[44] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM, 2007.

[45] Stuart E Middleton, Harith Alani, and David C De Roure. Exploiting synergy between ontologies and recommender systems. *arXiv preprint cs/0204012*, 2002.

[46] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266. ACM, 2003.

[47] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[48] Gonzalo Navarro, Ricardo A. Baeza-Yates, Erkki Sutinen, and Jorma Tarhio. Indexing methods for approximate string matching. *IEEE Data Eng. Bull.*, 24(4):19–27, 2001.

[49] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[50] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[51] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[52] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[53] Stephen R Porter and Michael E Whitcomb. The impact of lottery incentives on student survey response rates. *Research in Higher Education*, 44(4):389–407, 2003.

[54] Bruno Pradel, Nicolas Usunier, and Patrick Gallinari. Ranking with non-random missing ratings: influence of popularity and positivity on evaluation metrics. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 147–154. ACM, 2012.

[55] 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*, pages 157–164. ACM, 2011.

[56] Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

[57] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 10(2):90–100, 2008.

[58] Francesco Ricci and Fabio Del Missier. Supporting travel decision making through personalized recommendation. In *Designing personalized user experiences in eCommerce*, pages 231–251. Springer, 2004.

[59] Francesco Ricci, Daniel R. Fesenmaier, Nader Mirzadeh, Hildegard Rumetshofer, Erwin Schaumlechner, Adriano Venturini, Karl W. Wöber, and Andreas H. Zins. Dietorecs: a case-based travel advisory system. *Destination Recommendation Systems: Behavioural Foundations and Applications*, pages 227–239, 2006.

[60] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112. Association for Computational Linguistics, 2003.

[61] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[62] Sunita Sarawagi. Information extraction. *Foundations and trends in databases*, 1(3):261–377, 2008.

[63] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[64] Janyce M Wiebe, Rebecca F Bruce, and Thomas P O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics, 1999.

Appendix A

Appendix

A.1 Implicit Elicitation in Polivenus

We report here the detailed objects used in PoliVenus for implicit elicitation we described in detail in Section 3. Each figure shows all the objects available in the user interface. Active objects are highlighted, and generate signals that are used to create the user profile.

Figure A.1: Objects in the homepage.

Figure A.2: Objects in hotel list page.

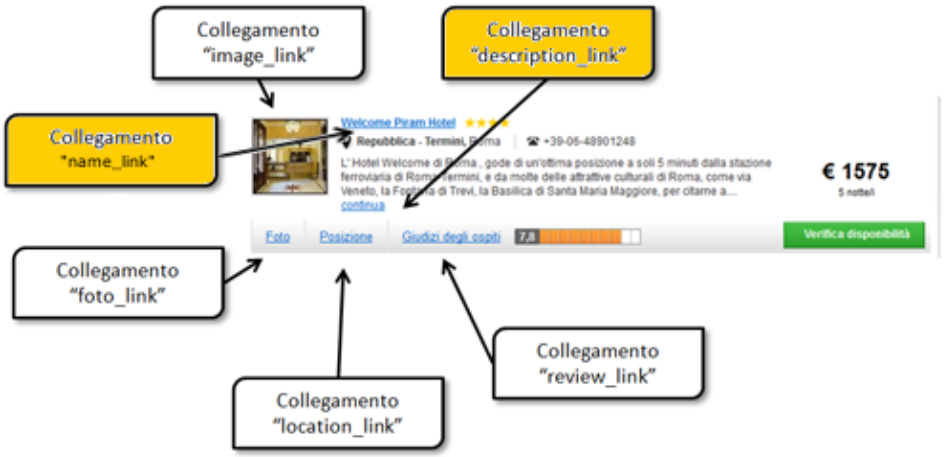


Figure A.3: Objects in hotel list page.

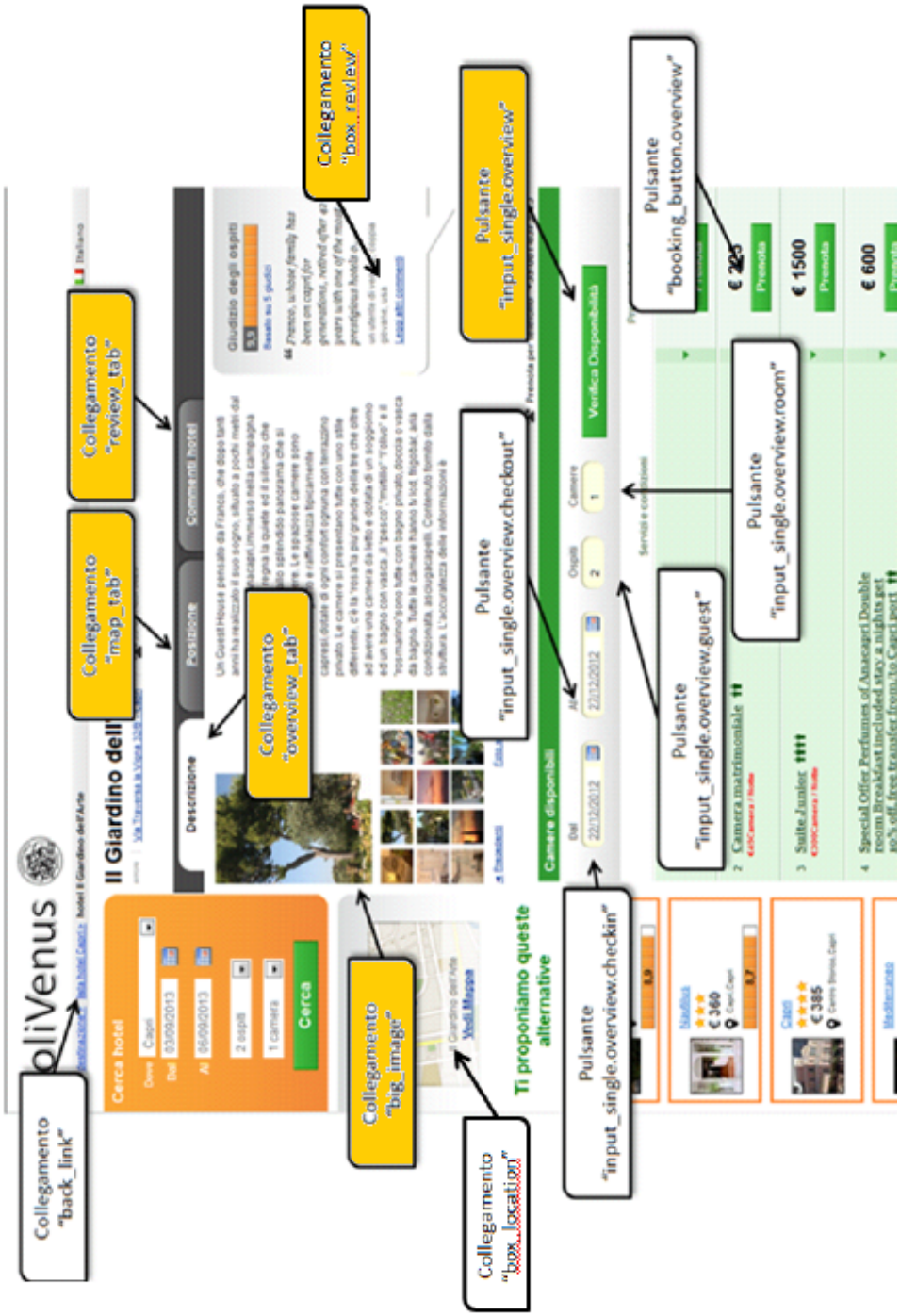


Figure A.4: Objects in hotel detail page (they are repeated in map and review pages).

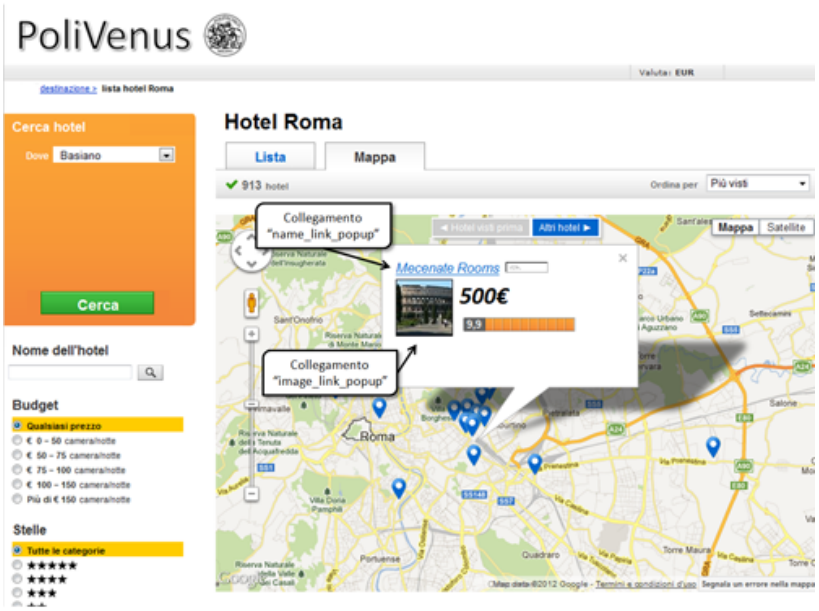


Figure A.5: Objects in the hotel location map.

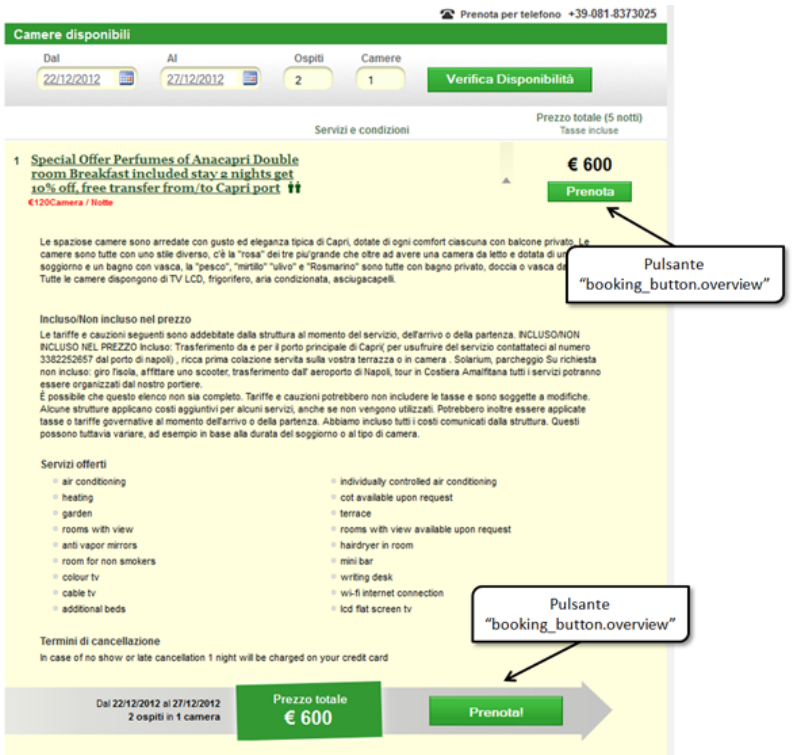


Figure A.6: Objects in the reservation detail page.

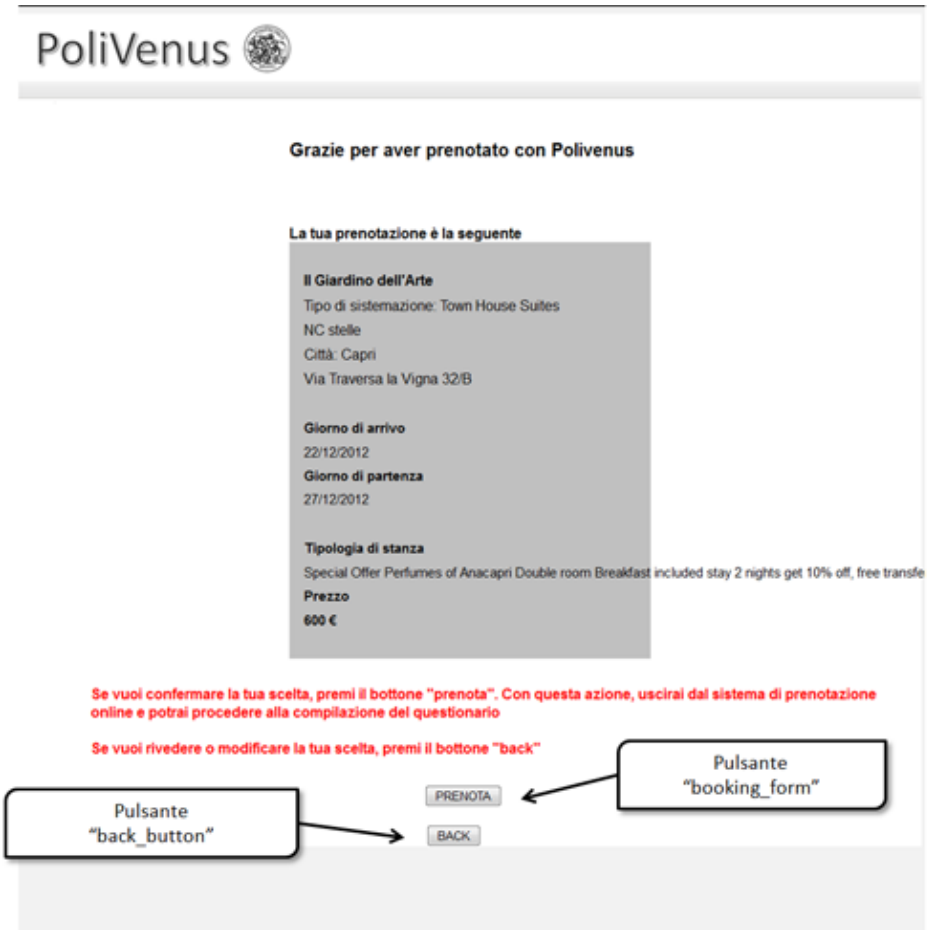


Figure A.7: Objects in the booking confirmation page.

A.2 Final Survey

Here we report the final survey that the participants of the user experiment has to complete at the end of the task. Each question is associated with the possible responses (parenthesis) and sub-questions (square brackets).

1. Is this the first time you stay in Rome? (Yes/No)
2. Did you previously stay in the hotel you choose? (Yes/No)
3. Is there any other hotel, different from the one you choose, that you would like to book but it was not available? (Yes/No)
4. Are you satisfied with your final choice? (No/Sufficient/Yes)
5. How much time did you approximately spend to make your reservation? (5/10/15/20/30 or more minutes)
6. Do you feel that such time was: (short/normal/long)
7. Do you feel that this activity was: (simple/correct/hard)
8. Will the description of the chosen hotel match its real characteristics? (Yes/No)
9. Do you think that this web site offers a satisfying hotel booking service? (No/Sufficient/Yes)
10. How many times do you use online hotel booking services? (Never/1 or 2/3 or more times per year)
11. Do you usually use Venere.com? (Never/1 or 2/3 or more times per year)
12. What are your priorities whey travel for vacation? [Price/Serices/Position/Adequate to my family] (Modest/Indifferent/Very important)
13. How old are you?
14. Gender
15. Nationality
16. Educational qualification
17. Profession

18. Where are you at the moment? (At Home/University/Work/Holiday/Other)

19. Do you have any commend you would like to express over PoliVenus? (Free text)

A.3 Opinion Observer Pseudo-Code

We report here the entire pseudo-code of the feature-based opinion mining tool we have re-implemented. The detailed explanation of the code is available in [19].

Algorithm A.1 OpinionOrientation()

```
1: foreach Sentence s_i that contains a set of features do
2:     F ← features in s_i
3:     foreach Feature f_i ∈ F do
4:         orientation ← 0
5:         if Feature f_i is in "but" clause then
6:             orientation ← apply the "but" clause rule
7:         else
8:             Remove "but" clause from s_i if it exists
9:             foreach Unmarked opinion word ow ∈ s_i do
10:                /* ow can bee a TOO word or Negation word as well */
11:                orientation += WordOrientation(ow,f_i,s_i)
12:        if orientation > 0 then
13:            f_i's orientation in s_i = 1
14:        else if orientation < 0 then
15:            f_i's orientation in s_i = −1
16:        else
17:            f_i's orientation in s_i = 0
18:        if f_i is an adjective then
19:            (f_i).orientation += f_i's orientation in s_i
20:        else
21:            o_ij ← nearest opinion word to f_i in s_i
22:            (f_i, o_ij).orientation+=f_i's orientation in s_i
23: /* Context-dependent opinion words handling */
24: foreach f_i with orientaiton = 0 in sentence s_i do
25:    if f_i is an adjective then
26:        f_i's orientation in s_i = (f_i).orientation
27:    else
28:        /* Synonym and antonym rule should be applied too */
29:        o_ij ← nearest opinion word to f_i in s_i
30:        if Exists (f_i, o_ij) then
31:            f_i's orientation in s_i = (f_i, o_ij).orientation
32:    if f_i's orientation in s_i = 0 then
33:        f_i's orientation in s_i = apply inter-sentence conjunction rule
```

Algorithm A.2 WordOrientation$(word, feature, sentence)$

```
1: if word is a Negation word then
2:     orientation = apply Negation Rules
3:     mark words in sentence used by Negation rules
4: else if word is a TOO word then
5:     orientation = apply TOO Rules
6:     mark words in sentence used by TOO rules
7: else
8:     orientation = orientation of word in opinionWord_list
```

A.4 Frequent Features

We report here the frequent features used in feature-based opinion mining over reviews in PoliVenus. Features marked with asterisk (*) are subsequently removed using sequential backward feature selection.

Feature	Support
hotel	9414
room	6696
breakfast	3407
staff	3276
locat	3117
good*	2467
place	1738
stay	1626
great	1624
station	1566
night	1410
price	1375
time	1195
walk	1150
friend*	1139
restaur	1101
area*	1087
bathroom*	1050
servic	1041
hotel room	1013
train	990
street	928
clean*	891
small*	851
hotel locat	842
littl	842
citi*	830
room breakfast	793
bus*	786
help	760
hotel staff	746
train station*	729
bed	692

Feature	Support
view	683
valu	682
quiet	642
problem*	622
peopl	620
distanc	585
nois	566
floor*	556
metro	531
shower*	530
hotel station	528
room clean	522
room bathroom	522
room staff	520
front*	501
site	495
hotel breakfast	484
money	468
staff help	467
way	460
staff friend	459
b&b*	457
trip	455
english	454
airport	446
window	436
thing	430
recept	428
desk	424
hotel price*	422
owner	421
central	421

Table A.1: Frequent features extracted from reviews in PoliVenus. Features marked with asterisk (*) are subsequently removed with sequential backward feature selection.

A.5 Linear Model Weights

We report here the complete list of feature weights used in our opinion summarization in PoliVenus. Each weight is presented together with its 95% confidence interval.

Feature	Weight	Lower Limit	Upper Limit
α_0	4.3796	4.3693	4.3899
hotel	0.1341	0.1222	0.1461
room	0.0596	0.0231	0.0961
breakfast	0.0735	0.0605	0.0866
staff	0.1421	0.1151	0.1691
locat	0.0520	0.0374	0.0665
place	0.0385	0.0082	0.0687
stay	0.0283	0.0075	0.0491
great	0.0580	0.0266	0.0895
station	0.1321	0.1098	0.1544
night	-0.0208	-0.0423	0.0006
price	0.1562	0.1421	0.1704
time	0.0629	0.0370	0.0888
walk	0.0343	0.0117	0.0568
restaur	0.0720	0.0359	0.1080
servic	-0.0172	-0.0426	0.0081
hotel room	0.0940	0.0422	0.1457
train	0.0162	-0.0249	0.0573
street	0.0470	0.0228	0.0712
hotel station	0.1484	0.0798	0.2171
littl	0.0306	0.0081	0.0532
room breakfast	0.0588	0.0150	0.1025
help	0.0667	0.0365	0.0968
hotel staff	0.1306	0.0891	0.1722
hotel locat	0.0215	-0.0098	0.0527
bed	0.0393	0.0063	0.0723
view	0.0616	0.0407	0.0825
valu	0.0353	-0.0000	0.0707
quiet	0.0605	0.0274	0.0936
peopl	0.0957	0.0758	0.1156
distanc	-0.0464	-0.0707	-0.0222
nois	-0.0677	-0.1034	-0.0319
metro	0.0205	-0.0054	0.0464

Table A.2: Resulting weight per feature together with their 95% confidence intervals.

Feature	Weight	Lower Limit	Upper Limit
room clean	0.0383	0.0104	0.0662
room bathroom	0.0461	0.0108	0.0814
room staff	0.1999	0.1091	0.2908
site	0.1261	0.1071	0.1451
hotel breakfast	0.1092	0.0189	0.1994
money	0.0527	0.0234	0.0821
staff help	0.1518	0.1225	0.1810
way	0.1211	0.0818	0.1604
staff friend	0.0984	0.0723	0.1245
trip	0.0847	0.0702	0.0992
english	0.0209	-0.0152	0.0569
airport	-0.2157	-0.2496	-0.1818
window	0.0503	0.0272	0.0733
thing	0.0356	0.0024	0.0688
recept	0.0411	0.0073	0.0749
desk	0.0330	-0.0306	0.0965
owner	0.0643	0.0297	0.0988
central	0.1492	0.1061	0.1924

Table A.3: Resulting weight per feature together with their 95% confidence intervals.