Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Benchmarking on web technologies and a recommender system development for E-commerce

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Anna Carolina de Oliveira Paegle

DIRECTOR: Roc Meseguer

DATE: December 21th 2012

Title: Benchmarking on web technologies and a recommender system development for E-commerce

Author: Anna Carolina de Oliveira Paegle

Director: Roc Meseguer

Date: December 21th 2012

Overview

Internet has opened a window enabling retailers to sell to anyone, anywhere and at any time. E-commerce has completely changed the way of doing business and in this context appeared the "**daily deals**" or "**group buying**" web pages: a business model which attracts millions of customers through the online sale of experiences – like a dinner or a trip – or products with a high percentage of discount, possible due to the great number of buyers.

Motivated by this context of growth, in this project a benchmarking on web functionalities is done for the main group buying web pages and some general electronic marketplaces.

As a result of this benchmarking a web functionality is chosen to be analyzed and developed for an e-commerce deals web: recommender systems. This leads to a second part of this project, where the main techniques to implement a recommendation system are studied and then used to generate a proof of concept for a recommender engine to work in a group buying web environment.

The goal of using a recommendation system in this environment is to make the right information arrive to the right costumers. Recommender systems are valuable both for users and businesses. From a consumer perspective, they may help the users to manage the information overload of the e-commerce world. From a corporate point of view, they may contribute to the cross-sell and upsell of products.

In this project, an item-based collaborative filtering approach is used to generate the recommender engine for the group buying web environment. After the model is designed, a proof of concept focused in the recommender core is implemented. At the end, some evaluation techniques for the recommender system are described.

INDEX

CHAPTER 1. INTRODUCTION 1

1.1. Background and motivation 1
1.1.1 E-commerce and Daily Deals 1
1.1.2 Project conception 1

1.2. Project scope 2

1.3. Environmental impact 3

1.4. Report organization 3

CHAPTER 2. BENCHMARKING – WEB FUNCTIONALITIES 4

2.1 Starting point 4
2.1.1 Pure daily deals 5
2.1.2 Reselling daily deals 6
2.1.3 Travel and leisure aggregators 7
2.1.4 Ticketing platforms 8
2.1.5 General marketplaces 9

2.2 Analysis conclusions 11

2.3 Web functionality selection 12

CHAPTER 3. RECOMMENDER SYSTEMS 13

3.1 Introduction 13

3.2 Building a recommendation engine 13
3.2.1 Users and items 14
3.2.2 Recommendation engine basics 14
3.2.3 Item-based and user-based analysis 15
3.2.4 Content-based and collaborative filtering techniques 16

3.3 Recommender systems and e-commerce 19
3.3.1 Example of recommendation systems in e-commerce 21

CHAPTER 4. MODEL DESIGN 23

4.1 Introduction 23

4.2 Analysis of the system environment 23
4.2.1 Requirements 24
4.2.2 Scope 24

4.3 Conceptual model 25
4.3.1 Structure of the recommendation system 25
4.3.2 Data gathering 26
4.3.3 Recommendation engine 27
4.3.3.1 Item- or used-based analysis 27
4.3.3.2 Collaborative or content-based approach 27
4.3.3.3 Recommendation engine overview 28

4.3.3.4 Computing similarity 29
4.3.3.5 Applying the model 30
4.3.3.6 New items problem 31

4.4 Final overview 32

CHAPTER 5. PROOF OF CONCEPT 34

5.1 Introduction 34

5.2 Generating the proof of concept 34
5.2.1 Weka 34
5.2.2 Data set 35
5.2.3 Generating user-item matrix 35
5.2.4 Computing item-to-item similarity matrix 36
5.2.5 Making recommendations 39

CHAPTER 6. EVALUATION 41

6.1 Introduction about the recommender system evaluation 41

6.2 Classification accuracy metrics 41
6.2.1 Precision and recall 42
6.2.2 The Kappa Statistic 43
6.2.3 ROC curves and ROC area underneath the curve 44

6.3 Testing results 45

6.4 Refining the recommender engine using feedback 47

CHAPTER 7. CONCLUSIONS 49

BIBLIOGRAPHY 51

IMAGES INDEX

Fig. 3.1 Users and items in a recommendation system 14
Fig. 3.2 Recommendation Engine 15
Fig. 3.3 User-item matrix 17
Fig. 3.4 User-based collaborative filter 18
Fig. 3.5 Item-based collaborative filter 18
Fig. 4.1 System Structure 25
Fig. 4.4 Illustration of the recommender system in the web 32
Fig. 5.1 User-item matrix for 5 items and 10 users 36
Fig. 5.2 Item-user matrix for 5 items and 10 users 36
Fig. 5.3 Normalized item-user matrix for 5 items and 10 users 37
Fig. 5.4 Item-to-item similarity matrix for 5 items and 10 users 37
Fig. 5.5 Fragment of the item-to-item similarity matrix to the first 50 items and 6000 users 38
Fig. 5.6 Input vector *U* – active user interaction data 39
Fig. 5.7 Input and Output vectors 40
Fig. 6.1 ROC curves for different systems 45

TABLES INDEX

Table 2.1 Web Functionalities of the main Daily Deals' web pages 5
Table 2.2 Web Functionalities of reselling daily deals sites 6
Table 2.3 Web functionalities – Atrapalo.com 7
Table 2.4 Web functionalities – Eventbrite 8
Table 2.5 Web functionalities of the main general marketplaces 9
Table 6.1 Characterization of kappa values 44
Table 6.2 Basic evaluation measures 46
Table 6.3 Fragment of the table with the detailed accuracy by class 46

CHAPTER 1. INTRODUCTION

1.1. Background and motivation

1.1.1 E-commerce and Daily Deals

Electronic commerce or e-commerce is the buying and selling of products and services over electronic systems such as Internet [1]. The possibility of digitalizing products (music, books, films, photography, coupons…) and distribute them online has turned entire industries upside-down, putting in danger many well established corporations and their business models. For physical and non-digitalizable goods, Internet has opened a window enabling retailers to sell to anyone, anywhere and at any time. But other competitors have taken advantage of this new medium and built digital empires such as Amazon, eBay, Google, Facebook, Apple, etc.

The advantages of selling online are many for buyers and sellers: open 24/7, geographical reach, cheaper prices and costs, wider variety, etc. These reasons, together with the increasing penetration of high-speed and mobile Internet, safer payment methods and the adoption of new consumer habits have made e-commerce boom in the last years through the entire world.

In the context of e-commerce, "**daily deals**" or "**group buying**" web pages are a business model which attracts millions of customers through the online sale of experiences – like a dinner or a trip – or products with a high percentage of discount, possible due to the great number of buyers. This phenomenon is certainly related to an increased sense of comfort by people to purchase online and to the economic instability – people are looking for the lowest prices. Groupon is the main responsible for the international expansion of this kind of web pages and these days a lot of "group buying" organizations are trying to emulate its model.

1.1.2 Project conception

The conception of this project came up from a new business idea based in the daily deals concept. I was involved in a group which wants to develop a new web page called "**Recupón**": a site where users can re-sell their unused coupons and a platform where business owners are able themselves to create and sell their deals, setting the conditions as they want to. This project was developed to support the implementation of the here discussed business idea.

Recupón has two business lines:

A) A web page where users may sell coupons they have purchased but finally they won't be able to use them. The idea here is to build a site to re-sell coupons – a marketplace of client-to-client selling.

B) A platform where the small businesses may build and sell their own coupons, removing the mediator's role of the group buying web pages and incorporating the business-to-client selling in the same Recupón marketplace.

In the case A, some sources state that 40% of the vouchers purchased don`t get redeemed [2]: Recupón would be the user's chance to not lose all his money. Besides, it would increase the number of redeemed coupons, which means fewer revenues to the group buying sites (those ones keep the total amount of the no-redeemed tickets) but more clients to the small businesses. This way a user may want to sign up to buy coupons (as in any daily deals site) or also to sell them. 'DealsGoRound' and 'CoupRecoup' in the USA and 'Regrupe' in Brazil are already implementing this business model. In Spain, 'Cup-off' is the first and only web to re-sell coupons. It appeared at the end of 2011 and it's still in its beta version.

The case B comes to respond to the small businesses need to create their own coupons and to sell them directly, without the group buying sites as abusive mediators. It will be very easy for the businesses to configure their coupons – models and guides teaching how to offer attractive tickets will be provided. Besides, selling in Recupón they will cash the total amount of the sold vouchers (except the commission, which includes IVA + management fee) including those vouchers that will not be redeemed.

1.2. Project scope

Motivated by the context presented in the previously sections, the purpose of this project is to carry out the following steps:

1. Perform a benchmarking analysis comparing the web functionalities of the main web pages related to the daily deals concept. Not only group buying or coupons re-selling web pages will be analyzed, but also another e-commerce successful platforms with many functionalities that could be useful but are still not used yet in those kind of pages.

2. After the benchmarking is done, one functionality will be chosen among the entire set of web functionalities analyzed – the idea is to choose a functionality that may possibly add value to the selling and re-selling deals web presented in this project background. For example, it may be interesting to try a functionality that is not popular yet in this kind of webs.

3. Study the chosen functionality: learn the main concepts that are necessary to implement it the following proof of concept.

4. Proof of concept: implement the chosen web functionality using the available technologies. At this point, the selection of the most suitable technologies is also very important.

5. Present a set of evaluation techniques that may be used to proof the implemented functionality.

1.3. Environmental impact

This project is directed to the web technologies sphere. Its usage is in the internet; therefore the environmental impact is the minimum. In order to measure this impact, energy consumption and maybe other materials (like paper) consumption should be analyzed and that isn't in the scope of this project.

1.4. Report organization

This project report is organized into seven chapters. The second chapter (first one after this introduction) contains a benchmarking on web functionalities of current daily deals webs, general e-commerce marketplaces and other similar webs. At the end of this chapter a web functionality will be chosen to be developed in this project.

In chapter 3 are presented the main theoretical concepts necessary to understand and develop a proof of concept of the web functionality chosen in chapter 2.

Chapter 4 contains the design of the model developed in order to implement the proof of concept, which is presented in chapter 5. Finally, in chapter 6 some evaluation techniques that may be applied in the proof of concept are presented, followed by the last chapter which contains the project final conclusions.

CHAPTER 2. BENCHMARKING – WEB FUNCTIONALITIES

2.1 Starting point

There are many different e-commerce platforms: for every kind of product or service, for every type of customer in whatever situation he or she is in. But the phenomenon regarding 'Daily deals' websites must be analyzed in its context, considering it as a part of a wider e-commerce scope. This sub-segment should also include platforms that sell immaterial goods and services, making them relevant to the daily deals business. Due to the kind of business they manage, they could be grouped in the following way:

- **Pure daily deals (or group buying):** Their approach is simple: every day you have at your disposal one or several deals, featuring a big discount on a product or service such as travelling, beauty treatments, electronic devices, leisure activities, etc. The idea is to focus the attention in that few deals, inducing to the purchase with clear click-to-actions.

- **Reselling daily deals webs:** These are second markets for unused coupons coming from pure daily deal webs. As they offer the opportunity to sell, they add many functionalities which make them slightly more complicated than the previous ones.

- **Travel and leisure aggregators**: Daily deals is one of their business lines, so they also have wider and more complex structures to make their complete offering accessible to the user.

- **Ticketing platforms**: Main tool used for individuals who want to publish and sell tickets for their own events. In this case the level of sophistication varies depending on the needs of customization of the user.

- **General marketplaces:** Current Internet titans such as eBay and Amazon. Their core business is to serve as marketplaces connecting individuals or business wanting to sell and buy. They offer different kinds of marketplaces and a lot of different functionalities to accommodate the offering to the buyer's and seller's need.

The benchmarking performed in this chapter is divided in the previously described five groups. In the following sections, the web functionalities of these categories' main players will be analyzed and compared.

Diverse categories could be added, but these are the ones competing directly in the e-commerce sector subject of this analysis. Nevertheless it must be recalled that the velocity of change in internet-related industries is overwhelming: new

players rise and industries are turned around in a matter of days, so the previous list could be completely different in a short period of time.

2.1.1 Pure daily deals

The sites analyzed here are Groupon, LetsBonus, Groupalia and Offerum. These constitute the main four players in the Spanish market. Groupon can be considered the pioneer, based in the USA. The other 3 were founded in Spain following Groupon's scheme. LetsBonus was bought by LivingSocial (US-based, the one truly worldwide competitor for Groupon), while Groupalia started an aggressive international expansion primarily in Latin America and Offerum is now finishing its expansion in Spain and some other European countries.

In the Table 2.1 one may see the most relevant web functionalities of today's main group buying platforms.

Table 2.1 Web Functionalities of the main Daily Deals' web pages

Web Functionalities	GROUPON	Groupalia	LetsBonus	offerum
User Profile	Basic registration info	Interesting deals, hobbies, education level, single?, job, money, children	Basic registration info	Basic registration info
Further subscription options	Yes: Twitter, Facebook, RSS	Yes: Twitter, Facebook, Flickr, Youtube, Foursquare	Yes: Twitter, Facebook, RSS	Yes: Facebook, Twitter and blog
Connect account to social networks or e-mail account	No	Yes: Facebook Connect	No	Yes: Facebook Connect
Recommend a deal with Social Netw. and e-mail	Yes: Facebook, Twitter and e-mail	Yes: Facebook, Twitter, e-mail, Blogger and AddThis	Yes: Facebook, Twitter and e-mail	Yes: Facebook, Twitter and Google+
Use E-mail or Social Netw. contacts to send invitations	No	No	No	Yes: Gmail, MSN or Yahoo
Newsletter with targeted offers	No	No	No	No

Recommendation	No	No	No	No
User/business reputation	No	No	No	No
User rankings	No	No	No	No
Member get member	Recommend a deal and get 6€	With the first friend's purchase, get 5€	With the first friend's purchase, get 5€	Friends' purchases: 10€ each
Mobile Apps	Yes	Yes	Yes	Yes
Integration with complementary platforms	No	No	Yes: Restalo.es	No
Online support chat	No	No	No	No
Search deals in map	No	No	No	No
Comment and rating store/deal	No	No	No	No
External Publicity	No	No	No	No

2.1.2 Reselling daily deals

Regrupe and DealsGoRound will be analyzed in this subsection. They are second markets for the 'leftovers' of the purely daily deals webs well established in Brazil and USA, respectively. Their web functionalities are stated in the following table.

Table 2.2 Web Functionalities of reselling daily deals sites

Web Functionalities		
User Profile	Basic info	Name, e-mail, Paypal account
Further subscription options	Facebook, Twitter, E-mail	Blog, Twitter, Facebook
Connect account to social networks or e-mail account	No	Yes: Facebook Connect
Recommend a deal with Social Netw. and e-mail	Facebook, Twitter	Facebook, Twitter, AddThis
Use E-mail or Social Netw. Contacts to send invitations	No	No
Newsletter with targeted offers	No	No

Recommendation	No	No
User/business reputation	No	No
User rankings	No	No
Member get member	No	No
Mobile Apps	No	Yes
Integration with complementary platforms	No	No
Online support chat	No	No
Search deals in map	Yes	No
Comment and rating store/deal	No	No
Interchange of messages (Seller/Buyer)	No	No
Link your account	No	Yes – link with your provider's account (Groupon, LivingSocial, etc.)
External Publicity	No	No
Use of API	Yes	Yes

2.1.3 Travel and leisure aggregators

In this group the analyzed site is Atrapalo, which saw its business seriously damaged when daily deals appeared. They launched a similar product called 'Hibiscus' to complement their offering for their commercial partners and also for a better exploitation of their existing customer database. In the Table 2.3 its web functionalities are analyzed.

Table 2.3 Web functionalities – Atrapalo.com

Web Functionalities	ATRAPALO.COM
User Profile	Photo, nickname, ID, Interests
Further subscription options	Yes: Twitter, Blog and Facebook
Connect account to social networks or e-mail account	No

Recommend a deal with Social Netw. and e-mail	Yes: Facebook, Tuenti, Google+ and Twitter
Use E-mail or Social Netw. Contacts to send invitations	No
Newsletter with targeted offers	No
Recommendation	No
User/business reputation	Business reputation: Users opinions (Punctuation - several categories)
User rankings	No
Member get member	No
Mobile Apps	No
Integration with complementary platforms	No
Online support chat	No
Search offers in map	No
Comment and rating store/deal	Users opinions (Punctuation - several categories)
External Publicity	No

2.1.4 Ticketing platforms

The web functionalities of Eventbrite are here analyzed (Table 2.4). This platform allows any individual to publish an event and sell their tickets.

Table 2.4 Web functionalities – Eventbrite

Web Functionalities	Eventbrite
User Profile	Organizer name, logo, about, settings (web site, number of events, categories, locations), communicate with attendees (add my Facebook page feed, twitter username), customized profile colors

Further subscription options	Facebook, Twitter, RSS
Connect account to social networks or e-mail account	No
Recommend an event with Social Netw. and e-mail	Yes: Facebook, Twitter, Linkedin, E-mail
Use E-mail or Social Netw. contacts to send invitations	No
Newsletter with targeted offers	No
Recommendation	Yes (System: top events, personalize your recommendations)
User/business reputation	No
User rankings	No
Member get member	No
Mobile Apps	Yes
Integration with complementary platforms	No
Online support chat	No
Search deals in map	No
Comment and rating store/deal	No
Interchange of messages (Seller/Buyer)	Yes
Follow the stores	Subscribe to a host
External Publicity	No
Use of API	

2.1.5 General marketplaces

The two e-commerce giants eBay and Amazon have their web functionalities enumerated here (Table 2.5). In these pages everyone can sell everything. Its foundation and success in the early years of Internet were a real breakthrough that certainly explains the birth of new business models such as daily deals websites. Currently, the daily deals business is just a tiny part of their gigantic enterprise.

Table 2.5 Web functionalities of the main general marketplaces

Web Functionalities	ebay	amazon.com
User Profile	Basic info, Addresses, Communication ways, Vote, Subscriptions	Name, city, photo, personal description, interests, tags, events, favorite items, wish lists
Further subscription options	Yes: Facebook, Twitter	No

Connect account to social networks or e-mail account	No	No
Recommend a product with Social Netw. and e-mail	No	Yes: Email, Facebook, Twitter
Use E-mail or Social Netw. contacts to send invitations	No	No
Newsletter with targeted offers	Yes (based on user's history)	No
Recommendation	Based on user history: similar items, from the same seller. System recommendation: People who viewed this also viewed…)	Between users. System recommendation: Frequently bought together, users who bought this also bought …, users who viewed this also viewed…, related items, similar items, random recommendations, amazon betterizer (like items), improve your recommendations
User/business reputation	Between users and system reputation	Between users and system reputation
User rankings	Yes	No
Member get member	No	No
Mobile Apps	Yes	Yes
Integration with complementary platforms	No	No
Online support chat	No	Yes
Comment and rating store/product	No	Rating items/Users reviews
Interchange of messages (Seller/Buyer)	Yes	Yes
Customer discussions	In the community	Yes – forums
Community	Yes (Announcements, Answers Center, Discussion Forums, Preview new features, Groups, eBay Top Shared)	Yes
External Publicity	No	Yes (sponsored links: related to your viewed items)

Use of API	No	No

2.2 Analysis conclusions

The main conclusions of the analysis previously performed are:

- Nearly all platforms analyzed include widgets or functionalities related to social media (Facebook, Twitter, etc.). Besides the e-mail option, the users may use social networks as Facebook and Twitter to subscribe to the platforms, to send invitations to their contacts and friends or to recommend a deal or product. Also, the platforms may obtain user's data directly from the social networks. The use of this type of communication is very popular and its rise in recent years consolidated them as an important channel to consider. Social media helps to engage prospective clients, facilitating its diffusion among the target individuals, as well as offering a more efficient way of performing customer service, and at a cost-worthy price for the organization.

- Emailing is a key driver of traffic and sales generation for daily deals firms. But all of them, except eBay, do not segment the content of the message. Without the proper targetization of the newsletter (ideally based on the historic record of the user), the customer loses interest in deals as these aren't relevant for them, eventually unsubscribing or sending the emails to spam and losing future sales, shortening the lifetime value of a customer. The targetization of the newsletter could also be simply done using the data into the user profile. In the analysis can be observed that the majority of webs only keep the basic data of the users. A profile with relevant users' data could be very useful to make sure that the right offers are arriving at the right person's inbox.

- From all the analyzed sites, the only ones that do recommendations in the web are eBay, Amazon and Eventbrite. There are different ways of recommending: randomly based on top sellers, such as Eventbrite, or based on the user's history – Amazon does it this way and outstandingly well. Amazon keeps record of your history and uses it to perform efficient cross-selling and up-selling. Recommendations could have a clear positive impact on the business as it increases the purchase amount, however only a handful of the players in the market do recommendations.

- Another interesting functionality that's not yet very popular among the analyzed sites is the reputation of users or businesses (only Atrapalo, eBay and Amazon use it). The users/businesses reputation can be given by the businesses/users or automatically by the platform. In the case of Atrapalo, for example, the users give a score to the businesses according to several categories. It could be especially useful for users to know if a business is reliable before making the decision of buying services or products from it.

- Mobile Apps are available for almost all the web sites analyzed. The increasing popularity of smartphones and tablets, together with technological advance (in geolocalization, Near Field Communication payment methods, among others) and new consumer habits, makes it necessary to develop a mobile application in an e-commerce platform. A service without the adequate mobile support and device integration is in clear disadvantage against other players, being left behind in technological advance and customer expectations.

- The two giant marketplaces eBay and Amazon have two very interesting functionalities: customer discussions forums and users community. When a platform reaches a great number of users like these both, having a community and allowing customers discussions could be a very good way of performing customer services, make publicity, obtain users' feedback among others.

- Everybody knows that another way of making money in internet is to provide space in your site to external publicity. In this analysis could be seen that only Amazon has external publicity in its web. The main reason why none of these webs are giving space for external publicity is the risk it means: sometimes it's possible you are publicizing your own competitors. A likely solution to this is to build some filters with which it's possible to avoid non-desired publicity.

2.3 Web functionality selection

After performing the benchmarking, I decided to choose recommendation as the web functionality to be studied and developed as a proof of concept in this project. The idea is to implement a recommendation system that works in the selling and re-selling deals context in two ways: making recommendations to costumers trough mailing and directly in the web.

The reasons of this choice are clear: recommender systems are a differentiator into e-commerce context as just a handful of players are using them. These systems try to ensure that the right information will arrive to the right person, which can make publicizing your own products through e-mailing or directly in the web much more efficient. As have seen in the benchmarking, although most analyzed webs don't use recommender systems yet, e-commerce successful giants like eBay and Amazon are already using them to help their customers find products to purchase.

As a consequence of this choice, the following chapters of this project will be focused in recommender systems and how to design a recommendation engine to be used in the context of a selling and re-selling web site as explained in the first chapter. The next chapter will present the main concepts and techniques to build a recommendation system and generate the proof of concept proposed.

CHAPTER 3. RECOMMENDER SYSTEMS

3.1 Introduction

As stated in last chapter's final section, a recommendation system to be used in a selling and re-selling deals web will be designed and its proof of concept will be generated. The goal of this chapter is to introduce recommender systems and describe the main techniques to build such systems, which will be necessary to design the proposed system.

Recommender systems are tools that help users find and evaluate items of interest. Through the association of user consumption and browsing patterns (which allow to gain insights on the user's preferences and inclinations) cross-referenced with the behavior of similar profiled consumers, recommender systems provide suggestions to support their users in decision-making processes, such as what items to buy or what music to listen. Recommender systems are valuable both for users and businesses:

- From a consumer perspective, in a world with an overwhelming number of choices, recommender systems allow users to manage the information overload and have become powerful tools in fields from electronic commerce to digital libraries and knowledge management.

- From a corporate point of view: recommender systems allow the cross-sell and upsell of products, complementing the user's choice and increasing the purchase ticket.

There have been several techniques developed for recommendation generation. In the following sections of this chapter the basic approaches to build a recommendation engine are presented. In the last section, an explanation of the current use of recommender systems in e-commerce can be found.

3.2 Building a recommendation engine

In this section, a brief introduction to various concepts related to building a recommendation engine is presented. The content of this section is drawn from the sources described in references [3] to [8].

3.2.1 Users and items

In a classical model of recommendation system, there are items and users. An item is any entity of interest in the application: may be articles, photos, movies, blog entries, daily deals, products, etc.

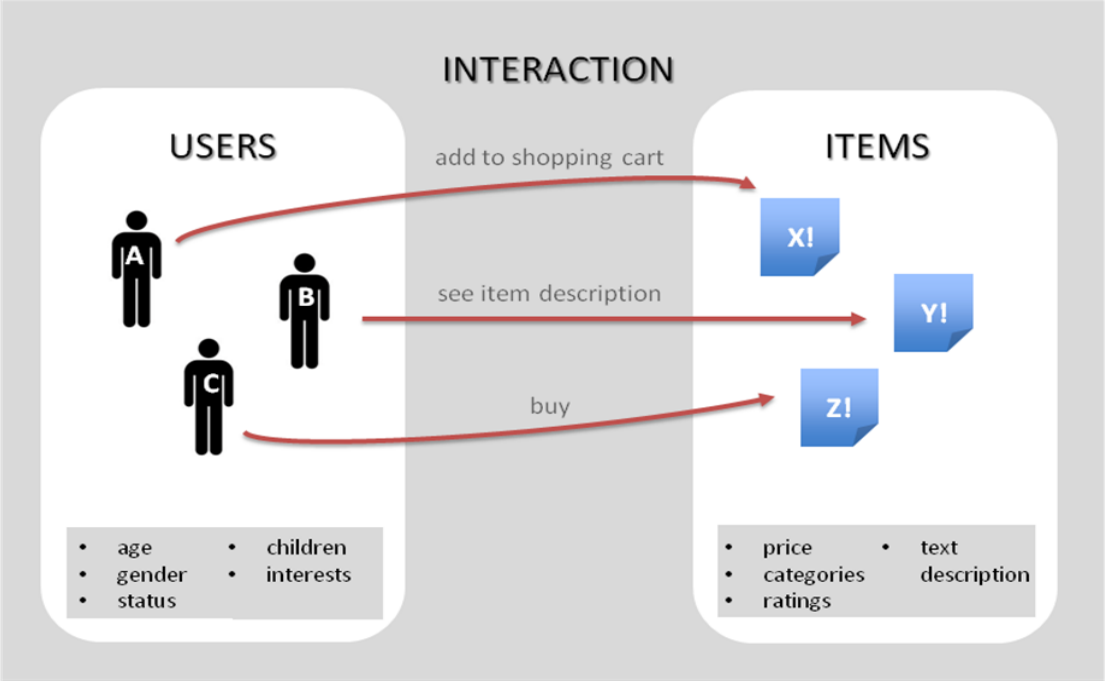

Fig. 3.1 Users and items in a recommendation system

Each item has associated metadata (or content), such as text description, price, ratings, popularity raking, or anything that provides a higher level of information about the item and can be used to correlate items together. Metadata can be understood as a set of attributes that help qualify an item.

Users have also associated metadata (or content), which consists of profile-based data such as age, gender, demographic information, etc.

On top of users and items, there are the interactions (or transactions) between them, such as user “A” acted on item “X” (where acted can be purchased, viewed, saved, etc.).

3.2.2 Recommendation engine basics

A recommendation engine usually uses three inputs to generate a recommendation:

- The user’s profile: information such as age, gender, geographical location, etc.

- Item's information: content associated with the items like price, category, etc.
- The interactions of the users: ratings, bookmarking, tagging, browsing content, emailing, etc.

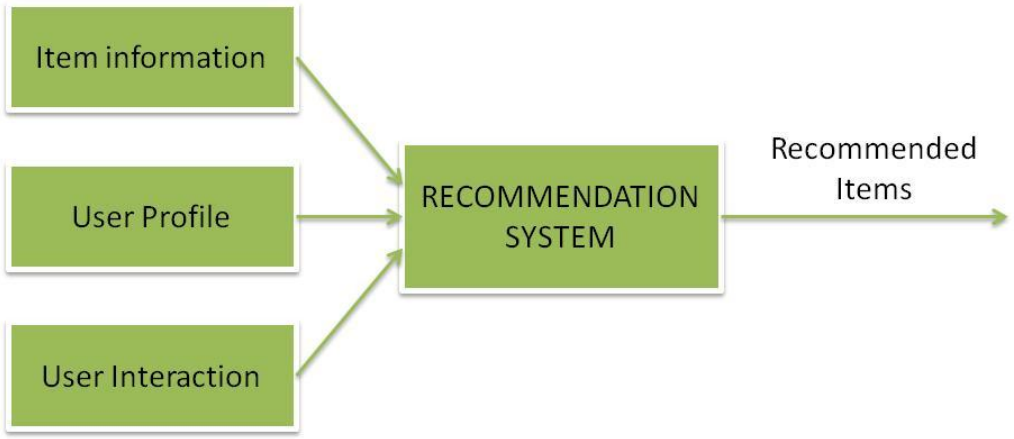

Fig. 3.2 Recommendation Engine

A simple "Top Item List", where the items that have been viewed, bought or tagged the most in a period are presented to the user, is a quite easy form of making recommendations. Promoting the top products could be useful, although these recommendations are not personalized.

Recommendation engines may help building features in the application like "items related to this item lists" or "users who acted on this item also took action on these other items", where the action could be purchasing, viewing, emailing, etc., can be added to the application. These kinds of recommendations are based in the user's and items' available information.

Recommend a few items selected at random may also be a good strategy. Besides the items that are similar to the ones the user has rated in the past, recommending new different ones gives more diversity to the recommendation set provide to the user and could help to find new recommendation spots that may be explored later.

In the following section two approaches to building a recommendation engine will be explained.

3.2.3 Item-based and user-based analysis

There are two main approaches to building recommendation systems: item-based or user-based analysis.

In item-based analysis the system searches for items related to a particular item. When a user likes an item, other items similar to liked one are recommended. In this case, the recommendation engine must find similar items. In user based analysis, users similar to the user are determined. If a user 'X' likes an item, then, the same item can be recommended to other users who are similar to user 'X'. In this case, the recommendation engine must find similar users using profile-based information (such as age, gender, etc.) or analyzing the users' actions.

Depending on the application, one approach can be better than other. In order to find which approach is most suitable for the application some things must be considered:

- If the item list doesn't change much, an item-to-item correlation table using item-based analysis can be used in the recommendation engine.
- If the item list changes frequently, find related users for recommendations could be useful.
- The dimensionality of the item and user space could determine which approach is easier to implement. If there are millions of users and an order of magnitude fewer items, item-based analysis may be the easier choice.
- When the number of users is small, a good option is to begin using an item-based analysis. Furthermore, there are no reasons why these two approaches can't be combined.
- There are empirically demonstrations that item-based algorithms are computationally faster to implement than user-based algorithms, providing comparable or even better results [16].

When building a recommendation engine with either item- or user-based analysis, it's necessary to compute a similarity metric. In the following section the two main approaches for computing similarity are explained.

3.2.4 Content-based and collaborative filtering techniques

Regardless of whether the recommendation system is item- or user-based, there are two main approaches for computing similarity: content- or collaborative-based analysis.

Content-based analysis is based on information about the item or user itself. In this approach, the metadata is used to categorize users and items and then match them at the category level. For example, when recommending jobs to candidates, a text search can be done to match the user's resume with the job descriptions. Similarity is measured according to the item's metadata and

various distance functions can be used: the goal here is to find the k-nearest neighbors of the item the user likes.

When building a content-based recommendation engine, one needs to implement strategies for:

- representing the items;
- creating a user profile that describes the types of items the user likes/dislikes;
- comparing the user profile to some reference characteristics to predict whether the user is interested in an unseen item or not.

Collaborative-based analysis uses the information provided by the interactions of users to predict items of interest for a user. That means in this approach, the interactions between users and items are used to perform the recommendation. A “user-item” matrix can represent the interaction data:

	Item 1	**Item 2**	**Item 3**	**Item 4**
User A				
User B				
User C				

Fig. 3.3 User-item matrix

Each cell in the matrix represents the interaction between user and item. For example, in a system where users rate items, a cell contains the rating that the user gave to the item (in this case the cell is a numeric value). In this example, the system will find patterns in the way items have been rated by the users to find additional items of interest for a user.

In a user-based collaborative filter, three steps are followed:

1- Find a group of users that are similar to user “A” based on users interactions

2- Find all the items liked by this group of users that hasn’t been viewed (or purchased, or saved, etc.) by user “A”

3- Rank these items and make recommendations to user “A”

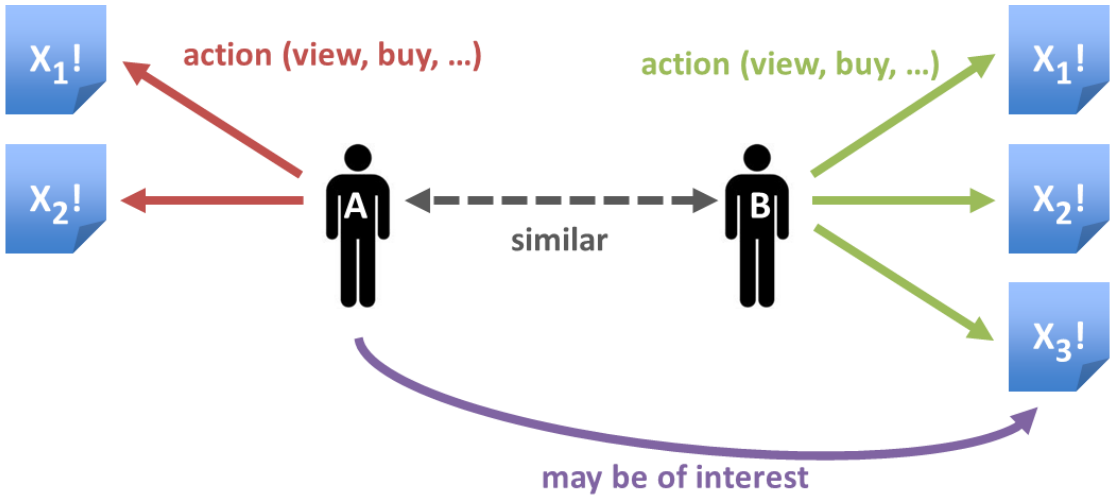

Fig. 3.4 User-based collaborative filter

In item-based collaborative filter, those three steps would be:

1- From interaction data, find the set of items user “A” likes

2- Find a group of items that are similar to those ones in the set of items liked by user “A”

3- Rank these items and recommend to user “A”

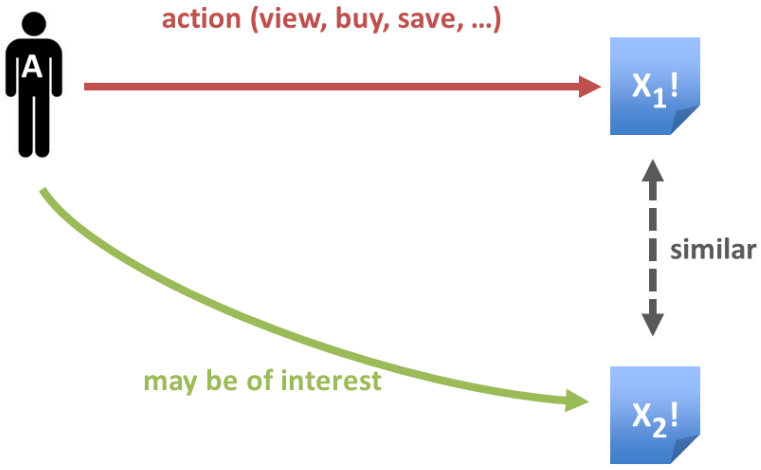

Fig. 3.5 Item-based collaborative filter

Both content- and collaborative-based approaches have their own advantages and drawbacks, as showed next:

- As collaborative-based techniques don't use any information about the content itself, they treat an item as a black-box. Therefore, the same infrastructure is applicable across domains and languages. The same infrastructure that works in English will also works in Chinese, which is not true for content-based analysis. Also, for content such as images or music, that might not have any text associated with them, content-based analysis may not be an option, unless users are allowed to tag the items.

- Using content-based analysis, the recommendation system can't predict the quality of an item (how popular is an item or how a user will like this item). On other hand, with collaborative approach items that are popular with a certain segment of the user population will appear often in their interaction history—viewed often, purchased often, and so forth. The frequency of occurrence provided by users is an indicative of the quality of the item to the appropriate segment of the user population.

- The results from content-based analysis don't change much over the time since the information associated with the item may not change much. Collaborative-based approaches rely on user interaction: over the time user interactions on that item are more likely to change.

- User-based collaborative systems use the information provided by a user to find other similar users and then, recommend items based on the similar users' ratings an items. If there isn't an adequate amount of data, the performance of these systems in their prediction may be not satisfactory. When a user is new in the system and have only a few interactions in his history, there may not be enough information to find similar users using the user's interaction history for a collaborative approach. Typically, to overcome this, user-profile information, such as age, gender, etc., is also used to find similar users.

- When new items are added to the system, unless they have been rated by a substantial number or users, collaborative-based systems won't recommend them. A hybrid approach can also be used, combining content-based and collaborative analysis. This combination could be obtained implementing the two methods separately and then combining the results.

3.3 Recommender systems and e-commerce

From the previous sections it could be understood that a recommender system learns from a customer and recommends products that he will find most valuable from among the available products. It's not difficult to see that they can be very powerful tools for electronic commerce platforms. Many of the largest commerce websites are already using recommender systems to help their

customers find products to purchase. The content of this section is based on reference [7].

Companies are shifting from the old world of mass production with standardized products and homogeneous markets to new world where variety and customization are the rule. Building one product is not enough: companies need to develop multiple products that meet the multiple needs of multiple customers. E-commerce has allowed companies to provide customers with more options. However, by offering a larger product assortment, the amount of information that customers must process before they are able to select which items meet their needs has also increased. Recommender systems are a solution to this information overload.

E-commerce sites may suggest products to their customers based on the top overall sellers, the demographics of the customer, or based on an analysis of the past buying behavior of the customer as a prediction for future buying behavior. Using these techniques the site increases its personalization and adapts itself to each customer, eventually increasing conversion. Recommender systems automate personalization on the Web, enabling individual personalization for each customer. Jeff Bezos, CEO of Amazon.com™ , once said “If I have 2 million customers on the Web, I should have 2 million stores on the Web.”

Recommender systems improve E-commerce sales in three different ways:

- Converting browsers into buyers: people often look over the site without ever purchasing anything. Recommender systems help customers find products they wish to purchase.

- Cross-sell: Recommender systems enhance cross-sell by suggesting additional products for the customer. If the recommendations are good, the average order size may increase. For instance, a site might recommend additional products in the checkout process, based on those products already in the shopping cart.

- Loyalty: Recommender systems enhance customer loyalty through the creation of a value-added relationship between the site and the customer. The websites invest in learning about their users with the use of recommender systems, and present custom interfaces matching to customer needs. Customers may repay these sites by returning to the ones that best match their preferences. The more a customer uses the recommendation system – teaching it what they want – the more loyal they are to the site.

3.3.1 Example of recommendation systems in e-commerce

In this section, a good example of e-commerce business that utilizes recommender systems technology in its website is presented: Amazon.com. Amazon is probably the canonical example of recommendations technology on the web. They report higher click-through and email advertising conversion rates using their recommendation engine, compared to untargeted content such as banner advertisement and top-seller lists.

Amazon makes recommendations based on information from multiple contexts, including:

- Short term information: based on item browsing history and on recent search terms.
- Items available in the shopping cart.
- Purchasing history: based on past purchases.
- The system also uses the information available about the user to send out targeted emails with personalized recommendations.

Amazon uses an item-based approach, where items similar to an item are used for recommendations. They use an item-to-item collaborative filtering algorithm that scales independent of the number of users and develops the expensive item-to-item similarity table offline.

The main recommendations features at Amazon are:

- Customers Who Bought This Item Also Bought: This feature can be found in the products information pages and it recommends a list of products frequently purchased by customers who bought the selected product. In the case the products are books or music CDs there is an additional list of recommendations for this feature: Amazon also recommends book or music authors whose books or CDs are frequently purchased by customers who bought works by the author of the selected product.
- Frequently Bought Together: This feature recommends products frequently purchased together with the selected product by other customers.
- What Other Items Do Costumers Buy After Viewing This Item: It recommends products that other costumers frequently purchased after buying the selected product.
- Your Recent History: At the page's bottom, Amazon presents customer's recently viewed items and keeps showing a list of products frequently purchased by people who bought the selected product. This history can

be easily edited by the user so the recommendations can be updated as the customer wishes.

- Customer Comments: The Customer Comments feature allows customers to receive text recommendations based on the opinions of other customers. Located on the information page for each product is a list of 1-5 stars ratings and written comments provided by customers who have purchased the product in question and submitted a review.

- Recommended For You: This feature is only for subscribed users. In the customer's page, Amazon recommends a list of products based on the user's purchased products or in the case the user has not bought anything yet, these recommendations could be based on other facts like the top-selling items.

- Amazon Betterizer: In order to obtain more user's information and provide more personalized product recommendations, this feature presents to the customer several lists of products grouped by categories (books, movies, etc.) and asks to the user to click on "Like" when some product is of his interest.

- Improve Your Recommendation: When a recommendation is made in the user's personal page, he has the option of telling Amazon that he is not interested in this item or he already owns it. The user can also rate the recommended item. At the end of the recommendation description there is an explanation of why this item was recommended to the user. For example, if a customer "likes" a book "X" of given author in Amazon Betterizer, other books of this author will be recommended to him. At the end of the recommendation will appear: "Recommended because you liked book "X". There is also an option to fix this and stop receiving recommendations based on this "like" action.

Based on the concepts presented in this chapter, the most suitable studied approaches will be selected and used to generate a recommender system (like the Amazon example just described), with the appropriated characteristics for a selling and re-selling deals environment. All the theory presented here will be the base to develop the mentioned recommended system in the following chapters.

CHAPTER 4. MODEL DESIGN

4.1 Introduction

At the end of Chapter 2, after a benchmarking of the web functionalities of the main daily deals/e-commerce webs, it was proposed to design a recommender system to be used for a selling and re-selling deals web site and to generate a proof of concept of this design.

After study the main concepts and techniques described Chapter 3, all this knowledge will be used to generate a model of what a recommender system for the mentioned type of web may look like. It's important to make clear that this model is only a simplified representation of a system that doesn't exist yet and to be designed eventually. The process of generating this model will be explained in the following sections.

The first step in the model design is the analysis of the recommender environment, defining the main requirements and the scope of the system. The analysis is followed by the conceptual design, containing the system's methodological framework.

4.2 Analysis of the system environment

The recommender system here developed is intended to work in a daily deals' web site, where people can re-sell not used deals and businesses owners themselves can post their deals online. In this kind of web, the amount of deals available can be really large and the recommender system may help the users to deal with the information overload and this way, maybe help to improve the web conversion rates.

As seen in the benchmarking performed in Chapter 2, this kind of web functionality is not very popular yet among daily deals webs, although it is really well developed for general electronic marketplaces like eBay and Amazon. Considering the target web will work not only for single deals that users are trying to re-sell but also as a platform where businesses owners can directly post their offers, the variety of services and products available to sell can represent a marketing challenge.

The generated recommendations may be employed both to recommend deals in the web site (may improve click-through conversion rate) as to recommend through emailing (may improve emailing advertising conversion rate).

4.2.1 Requirements

The main challenges (these could be also defined as the system requirements) to develop the recommender engine are:

- The recommender must work using the available information about users and items available in the web (users' history and items' metadata will be available).

- Responsiveness to new information:

    - The system shall respond quickly to new information from the costumers;

    - In this kind of web new items are added frequently and it's important to make sure that the recommender engine will also promote these new items.

- The system will deal with a large number of items and users, therefore the solution must be scalable.

- Information about new costumers is limited. This must be taken into account when recommending using user's history.

All these points must be considered when developing the recommender system.

4.2.2 Scope

As already said in the beginning of this chapter, the goal of this part of the project is to design a model for recommendation system for a selling and re-selling deals web site and to generate a proof of concept of this design. In this project will be emphasized the main characteristics of the recommendation system itself.

The following aspects of what would be the real world solution are left out of the model here presented:

- Programming language, platform, network features and all others implementation aspects.

- Log files and other the data files creation. In this project it's assumed that the database with information about items and users is already available.

4.3 Conceptual model

Now that the requirements and scope of the model are clearly defined, the next step is the construction of a conceptual model for the recommender engine, which is presented in this section. The conceptual design consists of a methodological structure for the recommender system and an explanation of its main components.

The foundations for the design choices made in the conceptual design are the requirements and scope specified in the previous section.

4.3.1 Structure of the recommendation system

The system structure can be seen in the next figure. As explained in Chapter 3, a recommender system has as inputs the information about items and users and the interaction between them.

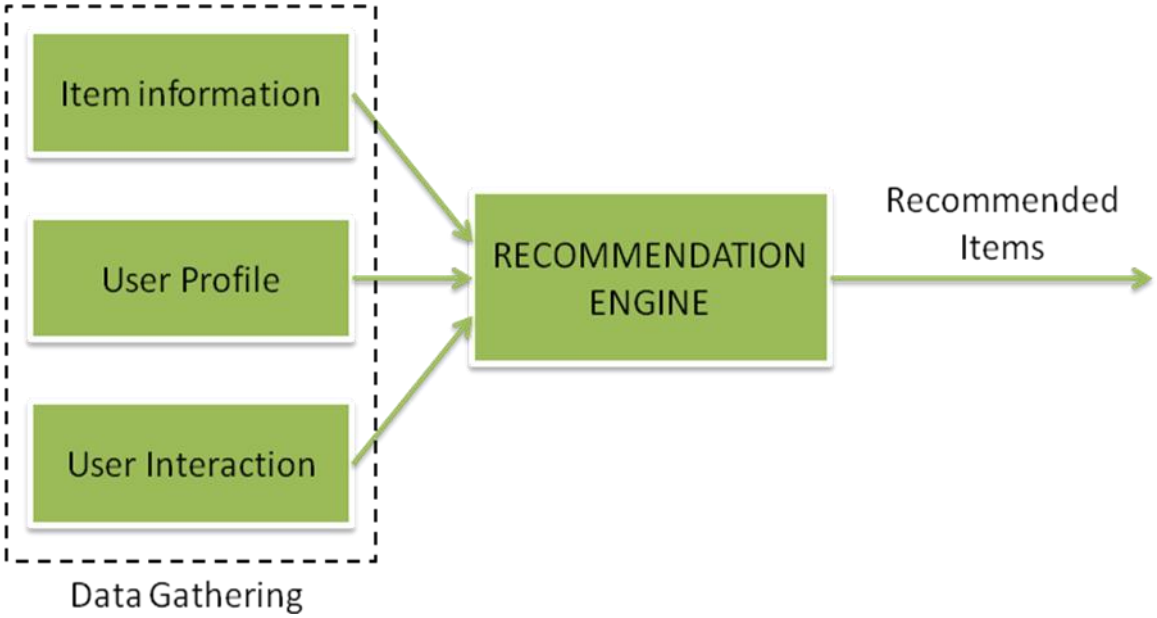

Fig. 4.1 System Structure

The structure presented can be separated into two modules: data gathering and the recommendation engine itself, which will be described in the next subsections. Please note that data gathering module it's not in the scope of the designed model and therefore, only a brief explanation of it is described. The emphasis of the model design is the recommendation engine itself. In this model, the users will be the costumers and the items will be the deals available for them to buy in the web.

4.3.2 Data gathering

It's not really in the scope of the model design to create the data gathering module. However, it's important to known which kind of data will be available as input to the recommender system.

Before make any recommendations the system must to collect data. The goal of data collection is to obtain an idea of the user preferences, which will be used later to make predictions in the recommender engine.

Data can be collected in two different ways:

- Explicit data gathering: in this method, one asks for explicit ratings from a user, typically on a concrete rating scale (such as rating a product from one to five stars).

- Implicit data gathering: in this way, one gather the data implicitly as the user is in the domain of the system, which means that is necessary to log the actions of the user on the web site.

It's obvious that explicit data gathering is easier to work with. Assumedly, the ratings that a user provides can be directly interpreted as the user's preferences, making it easier to make extrapolations from data to predict future ratings. However, the drawback with explicit data is that it puts the responsibility of data collection on the user, who may not want to take time to enter ratings.

On the other hand, implicit data is easy to collect in large quantities without any extra effort on the part of the user. Unfortunately, it is much more difficult to work with. The goal is to convert user behavior into user preferences, but it requires getting over one obstacle: how exactly does one infer preference based on actions in a system? This can be a difficult question to answer.

Of course, these two methods of gathering data are not mutually exclusive. A combination of the two approaches have the possibility for the best overall results – one could gain the advantages of explicit voting when the user chooses to rate items, and could still make recommendations when the user does not rate items by implicitly collecting data.

In the designed model, it's assumed that the data is gathered implicitly. Although implicit data can be harder to work with, it's more realistic for a selling and re-selling deals web site. The data will be collected from the users' actions, which can be:

- View an item description;

- Send the item by email, forwarding a link, share/like the item with social networks;

- Add an item to the shopping cart;

- Purchase an item.

In this model it will be assumed that the implicit data gathered are the views of the user. In practice, in the model being designed, when a costumer views a deal's description the system will interpret that the costumer is interested in this deal and will make the recommendations based on that information.

4.3.3 Recommendation engine

Based on the recommender systems' theory described in Chapter 3, there are several approaches for building a recommendation engine. The objective of this section is to describe the development of the recommender engine for the proposed model by finding the approach, which best fits for the case in question.

4.3.3.1 Item- or used-based analysis

In a selling and re-selling web site the items list may change frequently. This may be an indicative for use a user-based instead of an item-based analysis. However, this kind of web will have much more users than items (at least an order of magnitude fewer items than users), which means it will be easier to work with an item-based analysis.

If user-based analysis was used, the system would have to work with sparse matrices: for example, a typical user may have bought just a dozen of items from the thousands items that are available in the application.

Therefore, the model will perform an item-based analysis. The problem of the frequent variation of the items (deals) list can be solved with a periodically offline update of the model.

4.3.3.2 Collaborative or content-based approach

Now that is defined that the model will use an item-based analysis, the approach to compute similarity has to be decided: content- or collaborative-based approach (detailed described in Chapter 3).

When using content-based analysis, the recommendation system cannot predict how popular is an item or how a user will like this item, which can affect directly the quality of the recommendations. On other hand, in collaborative approach items that are popular (often viewed, purchased, etc.) will appear often in their interaction history. The frequency of occurrence provided by users is an indicative of the quality of the item to the appropriate segment of the user population. Also, collaborative-based analysis doesn't use any information of

the item itself. In a selling/re-selling web site where the users will be able to offer their products directly and post them online, it's good that the system doesn't have to rely so strongly on items content.

Because of that and the other advantages of collaborative filtering described in Chapter 3, and also being aware of its drawbacks, it's decided that the recommender system model will use collaborative-based to generate the recommendations. This means that in the model generated, the information used to recommend items will be the interactions between costumers and deal (for example, the fact the costumer viewed or purchased a deal).

4.3.3.3 Recommendation engine overview

The recommendation engine proposed uses an item-based collaborative-filtering approach. In order to develop this engine, the next steps will be followed:

1- Find the deals viewed by a costumer: when a costumer views an item description (in this model, the items are the deals for sale), this interaction data will be used to generate the recommendations.

2- Computing similarity: now, the system has to find a group of deals that are similar to the deals already viewed by the user. The collaborative computation of the similarity will be done based on the other users' history of interactions with the items in the system.

3- The last step is to rank the deals similar to the ones already viewed by the costumer and then recommend them.

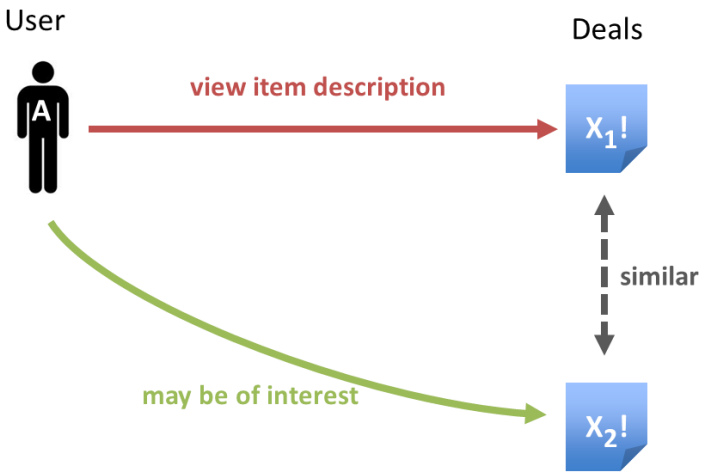

Fig. 4.1 Steps in recommendations generation

Looking at those three steps, computing similarity is the next challenge in building the model.

4.3.3.4 Computing similarity

Since collaborative filtering techniques are used in this model, items should be considered black boxes (their content is not relevant) and user interactions (viewing an item description) with the items are used to recommend an item of interest to the user. This means that the system will generate recommendations using the dataset of users' actions on items (user-item dataset). As already explained in the model scope how this dataset is obtained is not in the scope of this project.

The user-item dataset will be used in the form of the following matrix:

	Deal 1	**Deal 2**	**…**	**Deal m**
User 1	1	0	…	0
User 2	0	1	…	1
…	…	…	…	…
User n	1	0	…	1

Fig. 4.2 User-item matrix example

The cells in the matrix are set to "1" if the user has viewed the corresponding deal or to "0" if the deal in question was not viewed by the user yet. In the case of the selling and re-selling web site, this should be a sparsely populated dataset, since each user has viewed probably only a small number of the available items.

Using this data, the following answers have to be found:

- What other deals have been viewed by other users who viewed the same deals as a specific user? The answer to this question can be easily extracted from the user-item matrix.

- What are the related items based on the viewing patterns of the users?

To determine the answer to the last question, the similarity between the items in the user-item matrix has to be calculated. The approach used in this model to compute similarity between items is the cosine-based computation and is described as follows.

Cosine-based similarity computation

In order to compute the similarity between two items, each item will be treated as a vector in the space of users. The cosine between these vectors will be then used as a measure of similarity.

Formally, *R* is the *n*×*m* user-item matrix, then the similarity between two items *i* and *j* is defined as the cosine of the *n* dimensional vectors corresponding to the i_{th} and j_{th} column of matrix R. The cosine (or similarity) between these vectors is given by:

$$sim(i, j) = \cos(\vec{R}_{*,i}, \vec{R}_{*,j}) = \frac{\vec{R}_{*,j} \bullet \vec{R}_{*,i}}{\left\|\vec{R}_{*,i}\right\|^2 \left\|\vec{R}_{*,j}\right\|^2}$$

where "$\bullet$" denotes the vector dot-product operation.

The cosine function measures the angle between the two vectors (two items). As a result, frequently viewed items will tend to be similar to other frequently viewed items and not to infrequently viewed items.

The first step to implement the cosine-based similarity computation is to transpose the matrix in Fig. 4.2. Next, the values of each row are normalized. This is done by dividing each of the cell entries by the square root of the sum of the squares of entries in a particular row. Then, the similarities between the items can be found by taking the dot product of their normalized vectors. Using this, the item-to-item similarity table can be developed. The closer to 1 a value in the similarity table is, the more similar the items are to each other.

4.3.3.5 Applying the model

The following steps have to be followed for applying the collaborative item-based model developed:

1 – The inputs to this algorithm are the item-to-item similarity matrix *M,* the vector *U* (that stores the items that have already been viewed by the active user) and the number *N* of items to be recommended.

2 – The active user's information in vector *U* is encoded by setting $U_i = 1$ if the user has already viewed the i_{th} item and zero otherwise.

3 – The output of the recommender model is a vector *x (mx1)* whose entries different from zero correspond to the top-N items that will be recommended. In order to compute the vector *x* the following steps are needed*:*

- First, vector *x* is computed by multiplying *M* with *U*. This way, the nonzero entries of the *x* will correspond to the union of the *k* most similar items for each item that has already been viewed by the active user. The weight of these entries is nothing more than the sum of these similarities.

- Second, the entries of *x* that correspond to items already viewed by the active user are set to zero.

- And finally, the algorithm sets to zero all the entries of *x* that have a value smaller than the *N* largest values of *x.*

4.3.3.6 New items problem

As a collaborative approach, the designed model will occasionally have the new items problem. Collaborative-based systems won't recommend new items added to the system unless they've been rated by a substantial number of users. For example, in the designed model, if a new deal has just been added to the web and it has none or just a few views by the users, this deal won't be recommended.

To overcome this problem, one possible solution would be to build a hybrid engine by adding a content-based recommender module to the recommendation system. This module would classify the item (deal) based on its metadata – price, category, text description, etc. – to generate recommendations.

Another simpler solution would be to add a simple separated module which recommends the new (most recently added) items randomly with the aim of promoting new items.

This is also a way to recommend deals that aren't in the user's current spot: in essence, using the proposed recommender system, the users will never be presented with items that are outside their current spot. Randomly recommending the newest deals in the system will build in some diversity in the recommendations set provided to the user and maybe a new user's preferences spot can be found and explored later.

In the recommender system developed in this project, I would say that the best choice to solve the new items problem would be to add the separated module that randomly recommends the new deals to the user. This would be much simpler than build a hybrid recommender with a content-based engine and wouldn't increase much the computational cost. Besides, this way the user is also receiving suggestions of deals that normally wouldn't be recommended to him, which can maybe lead the user to try different products and explore new consumption spots.

A general illustration of what the recommender system would look like to the costumers can be seen in the following figure.

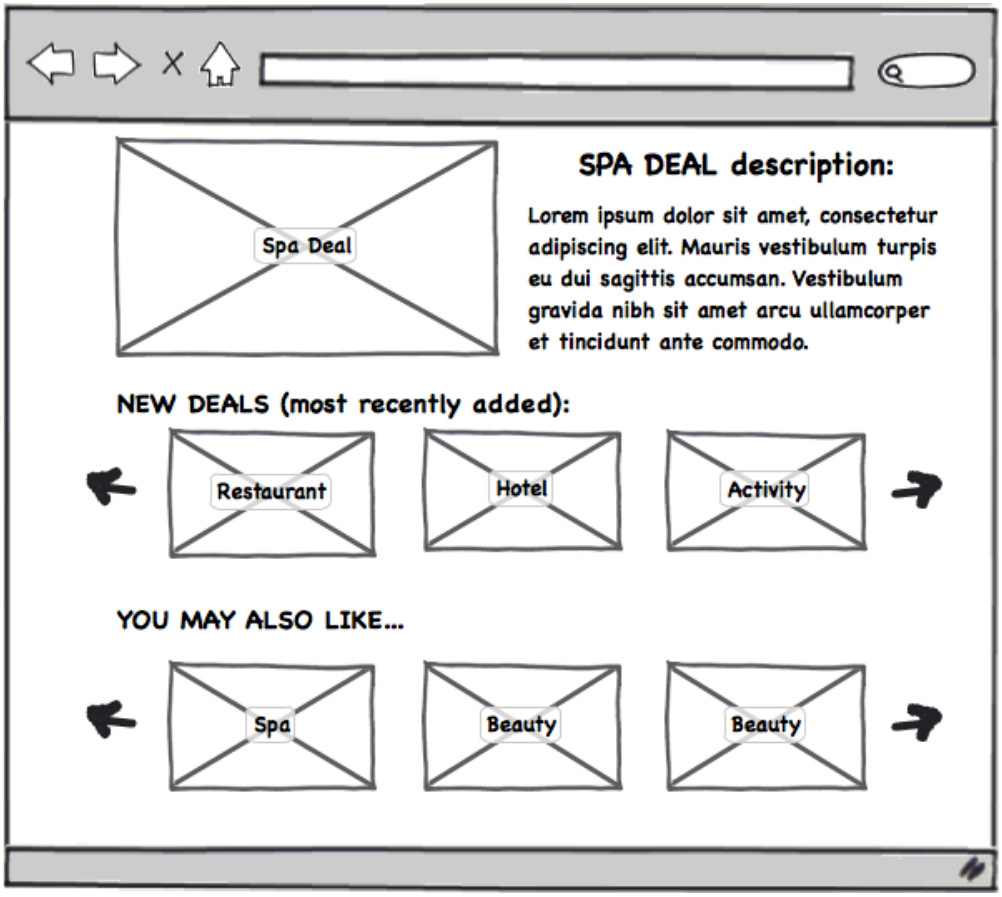

Fig. 4.2 Illustration of the recommender system in the web

To develop this extra module to randomly recommend new items isn't in the scope of this project. It's only a proposed solution to the new items problem and it wouldn't be taking into account in the proof of concept generation.

4.4 Final overview

The recommender system here developed meets the requirements in section 4.2.1, which can be conclude from the following affirmations:

- An item-based collaborative filtering was designed. Recommendations are made based in the users' actions on items history. Since the items are treated as black boxes, the same infrastructure is applicable across domains and languages.

- The recommender model uses an item-to-item similarity matrix that has to be updated periodically in order to respond to the new information

from the costumers and items. This similarity matrix will be generated offline and then, related items for an item are stored in the database.

- Once the related item table has been computed for each item, displaying related items to a user is a simple lookup. This approach scales independently of the number of users on the system.

- Since the recommender developed uses an item-based approach, the problem of limited information about new costumers isn't a really big problem. The recommendations are made based in the similarity between items, and when the user views an item description, items similar to the viewed item will be recommended. This means that the system will be able to generate recommendations even though the user has viewed only a few items.

- In order to promote items that are new in the system and will not be recommended until they've been rated by a substantial number of users, it was proposed to add a module that randomly recommends the newest items.

Since the requirements are met, the next step in this project will be to implement the designed model as a proof of concept. This implementation will be described in the next chapter.

As could be noticed in this chapter, the focus of the model design was always to make recommendations in the web. However, in the last section of Chapter 2, I also proposed to make recommendations through mailing. Although that was my initial idea, after study recommender systems and design this model I had the feeling that the implementations would be pretty much the same for recommend deals in the web or through mailing and that the main differences would be in details that aren't in the scope of the project. Therefore, I decided to implement a recommendation system thought to work only in the web.

CHAPTER 5. PROOF OF CONCEPT

5.1 Introduction

The objective of this chapter is to generate a proof of concept of the model design developed in the last chapter.

A proof of concept can be positioned between the conceptual model and the prototype implementation, and is useful to verify the usability of the model before putting effort into implementing a prototype. A proof of concept is therefore more abstract than a prototype, and focuses on the basic principles and assumptions made in the model. Its purpose is to verify that some concept or theory has the potential of being used.

In this context, the main goal here is to implement a proof of concept of the main core of the recommendation model designed, treating the data gathered (assuming this data is already available) and using the collaborative techniques in order to find similar items and generate relevant recommendations for the users.

5.2 Generating the proof of concept

5.2.1 Weka

Weka is an open source collection of machine learning algorithms for data mining tasks, developed by the University of Waikato, New Zealand [12]. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

The Weka workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. It is freely available software. It is portable and platform independent because it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.

Weka libraries for Java will be used to implement the recommender designed in this Chapter and also in the evaluation of the system which will be done in the next Chapter.

5.2.2 Data set

As the real web was not implemented yet and therefore, any real data could be gathered, it's necessary to use an existing and available data set to create the model and train the recommender. The chosen data set was the MovieLens Data Set, a classic data set in the literature of data mining and recommender systems, collected by the GroupLens Research Group from the University of Minnesota. The data sets by collected are available in the MovieLens web site [11].

The data set used to perform this proof of concept was the MovieLens 1M, which consists of approximately 1 million anonymous ratings of 6040 movies made by 3952 users who joined MovieLens in 2000.

The data available in GroupLens web is already cleaned up – users who had less than 20 ratings were removed from this data set. The data file used in this project is the "ratings.dat", which contains all ratings from the users on the movies, made in a 5-star scale (whole-star ratings only). The data format is:

UserID::MovieID::Rating::Timestamp

The UserIDs go consecutively from 1 to 6040 and the MovieIDs from 1 to 3952. The time stamps won't be used in this project.

5.2.3 Generating user-item matrix

Before implement the recommendation engine itself, the data set available need to be adapted in order to be more suitable to implement the proof of concept.

Since in the designed recommender engine the data gathered will be the user's action of viewing an item description, the original data set will be changed in order to be more near to the designed model's context. That will be done by substituting the ratings from 1 to 5 by "1", which means only that the user watched the movie no matter which was the given rating. The non-rated movies will remain as "0", which means that the user didn't watch these movies.

The user-item matrix used is therefore obtained from the modified data set. In the created matrix, the rows represent the users and the columns the movies. To each cell a value of "0" or "1" is assigned: "0" if the user didn't watch the movie and "1" if the movie was already watched by the user.

This means that for the real model to be implemented, in the user-item matrix the rows will be the users, the columns will be the deals offered in the web and each cell may have a value of "0" when the user didn't view the deal description or "1" when the deal description was already viewed by the user.

5.2.4 Computing item-to-item similarity matrix

The user-item matrix created in the last section will be used to compute the item-to-item similarity matrix that will be used in the designed recommender. It's worthy to say that the items in this implementation will be the movies of the available data set. When the recommender is implemented in the real life system and real data is available, the items will be the deals offered to the costumers in the selling/re-selling deals web. To better illustrate the implementation, the user-item matrix for the first 5 items (movies from the data set) and 10 users would be:

	item1	item2	item3	item4	item5
user1	0	1	1	1	1
user2	1	1	1	1	0
user3	1	1	1	1	1
user4	0	0	0	0	1
user5	0	0	1	0	0
user6	0	1	0	1	1
user7	0	0	1	1	1
user8	0	0	1	1	1
user9	1	1	1	1	0
user10	0	0	0	1	0

Fig. 5.1 User-item matrix for 5 items and 10 users

As explained in the last chapter, after the matrix containing the interactions between users and items is created, the similarity between the items will be finding by using the cosine-similarity computation. In order to make these computations the Weka Class Matrix (*weka.core.matrix.Matrix*) will be used to perform operations on a matrix of floating-point values.

As the similarity will be computed between items, the first step is to transpose the user-item matrix created in the last section, generating the item-user matrix so that a row corresponds to a deal offered in the web (or a movie, when thinking in the used data set) while the columns (users) correspond to dimensions that describe the deal (or movie). The item-user matrix considering only the first 5 items and 10 users would be:

	user1	user2	user3	user4	user5	user6	user7	user8	user9	user10
item1	0	1	1	0	0	0	0	0	1	0
item2	1	1	1	0	0	1	0	0	1	0
item3	1	1	1	0	1	0	1	1	1	0
item4	1	1	1	0	0	1	1	1	1	1
item5	1	0	1	1	0	1	1	1	0	0

Fig. 5.2 Item-user matrix for 5 items and 10 users

Next, the values for each row are normalized. This is done dividing each row by the Frobenius norm of the matrix consisted of this same row. The Frobenius norm corresponds to the square root of the sum of the squares of all elements. This calculation can be seen in the code fragment bellow (the Frobenius norm is computed by the function normF() from the matrix class of Weka):

```
for (int i=0; i<rowsNumber; i++){

        Matrix m = itemUserMatrix.getMatrix(i,i,0,columnsNumber-1);
        double norm = m.normF();
        double inverse = 1/norm;
        Matrix n = m.times(inverse);
        normalizedMatrix.setMatrix(i,i,0,columnsNumber - 1,n);
}
```

The normalized item-user matrix considering only the first 5 items and 10 users would be:

	user1	user2	user3	user4	user5	user6	user7	user8	user9	user10
item1	0,000	0,577	0,577	0,000	0,000	0,000	0,000	0,000	0,577	0,000
item2	0,447	0,477	0,447	0,000	0,000	0,447	0,000	0,000	0,447	0,000
item3	0,378	0,378	0,378	0,000	0,378	0,000	0,378	0,378	0,378	0,000
item4	0,354	0,354	0,354	0,000	0,000	0,354	0,354	0,354	0,354	0,354
item5	0,408	0,000	0,408	0,408	0,000	0,408	0,408	0,408	0,000	0,000

Fig. 5.3 Normalized item-user matrix for 5 items and 10 users

The similarities between the items can be found by taking the dot product of their vectors. Using this, the item-to-item similarity matrix can be developed. In the example considering only the first 5 items and 10 users, the item-to-item similarity matrix would be:

	item1	item2	item3	item4	item5
item1	1,000	0,755	0,655	0,612	0,236
item2	0,775	1,000	0,676	0,791	0,548
item3	0,655	0,676	1,000	0,802	0,617
item4	0,612	0,791	0,802	1,000	0,722
item5	0,236	0,548	0,617	0,722	1,000

Fig. 5.4 Item-to-item similarity matrix for 5 items and 10 users

If the data available was only composed by the first 5 items and 10 users, the similarity matrix above would be used to make the recommendations. In this case, one can see that for the first item (movie, in this case); the most similar

item would be the item2. That means that if a user in the system has viewed item1, item2 would be recommended to this user.

It's obvious that such a small amount of data won't lead to any meaningful conclusions. A fragment of the item-to-item similarity matrix obtained if the same steps are done to the first 50 items and 6000 users can be seen as follows:

	1. American Beauty	2. Star Wars Ep. IV	3. Star Wars Ep. V	4. Star Wars Ep. VI	…	25. The Godfather	…	49. Jaws	50. The Godfather II
1. American Beauty	1,000	0,561	0,573	0,537	…	0,529	…	0,439	0,466
2. Star Wars Ep. IV	0,561	1,000	0,788	0,719	…	0,613	…	0,619	0,553
3. Star Wars Ep. V	0,573	0,788	1,000	0,759	…	0,604	…	0,589	0,553
4. Star Wars Ep. VI	0,537	0,719	0,759	1,000	…	0,540	…	0,533	0,484
…	…	…	…	…	…	…	…	…	…
25. The Godfather	0,529	0,613	0,604	0,540	…	1,000	…	0,567	0,751
…	…	…	…	…	…	…	…	…	…
49. Jaws	0,439	0,619	0,589	0,533	…	0,567	…	1,000	0,530
50. The Godfather II	0,466	0,553	0,553	0,484	…	0,751	…	0,530	1,000

Fig. 5.5 Fragment of the item-to-item similarity matrix to the first 50 items and 6000 users

The cells with the black frame are the ones with the higher similarity in their row. It's interesting to see that:

- For item2 – the movie Star Wars, Episode IV – the most similar item is the number 3 – Star Wars, Episode V and vice-and-versa.

- For the item number 4 – Star Wars, Episode VI – the most similar item is the number 3 – Star Wars, Episode V.

- For the movie number 25 – The Godfather – the most similar movie is the number 50 – The Godfather Part II – and vice-and-versa.

These isolated results suggest that the similarity computed obeys a logical rule, since movies and their sequels are associated as most similar. However it's just a hint of the recommender system results – true evaluation techniques will be presented in the next Chapter.

5.2.5 Making recommendations

Now that the similarity matrix is computed, the system needs to make recommendations. All the matrix operations performed in this section were made using the Weka Core Matrix Class.

Considering the input to the recommendation engine a vector *U,* each position U_i is set to "1" if the i_{th} item was already viewed by the user and to "0" otherwise. This vector contains all the items already viewed by the active user. Continuing with the model generated in the last section with the first 50 items (movies) and 6000 users were used to compute the item-to-item similarity matrix, the input vector *U* must contain which ones of those 50 movies were already watched by the active user. Let's make the active user the last user in the corresponding user-item matrix. A fragment of vector *U* is presented as follows:

	1. American Beauty	2. Star Wars Ep. IV	3. Star Wars Ep. V	4. Star Wars Ep. VI	…	49. Jaws	50. The Godfather II
Active User	1	1	0	1	…	0	0

Fig. 5.6 Input vector *U* – active user interaction data

Given this input, the output will be a vector *x* whose entries different from zero corresponds to the top-N items that will be recommended. To compute the vector *x* the first step is to multiply the computed similarity matrix with *U.* This way, all nonzero positions of the *x* will correspond to the union of the *k* most similar items for each item that has already been viewed by the active user. The weight of these entries is nothing more than the sum of these similarities.

Once the multiplication is done, all entries of *x* that correspond to the items already viewed by the user will be set to zero. The input vector and the output vector are represented as follows:

	1. American Beauty	2. Star Wars Ep. IV	3. Star Wars Ep. V	4. Star Wars Ep. VI	…	49. Jaws	50. The Godfather II
Active User	1	1	0	1	…	0	0
Output	0,000	0,000	15,800	0,000	…	12,300	11,400

Fig. 5.7 Input and Output vectors

In order to make the recommendations to the user, all the positions of *x* that have a value smaller than the *N* largest values of *x* are set to zero, being *N* the number of items to be recommended. Following with the example, if *N* is set to 3, than the three movies that would be recommended to the active user are movie3 (Star Wars Ep. V), movie12 and movie9 (this can be seen in the complete output vector). As expected from the item-to-item similarity matrix, since the user already watched Star Wars Ep. IV and Star Wars Ep. VI, the movie Star Wars Episode V should be recommended to him.

Unfortunately, as already said, real data for the selling/re-selling deals web environment is still not available since the web isn't implemented yet. In the next chapter some evaluation techniques that can be used in the implemented model once real data is available will be discussed.

CHAPTER 6. EVALUATION

In this chapter the main techniques for evaluating a recommender system will be presented. It's important to say that as the real-life system doesn't exist yet this results cannot give a real evaluation of the recommender system working in the real-life web site – the objective here isn't to obtain a real evaluation of the recommender engine implemented but to know how the evaluation can be performed, proposing a solution that may be employed in future in the real-life working system.

As already mentioned in the last Chapter, Weka workbench contains will be used to perform the main evaluation techniques on the developed recommender system.

6.1 Introduction about the recommender system evaluation

The recommender system developed in this project is a sub-domain of information filtering system techniques that attempts to recommend information items – the offered deals in the selling and re-selling web – that are likely to be of interest of the user. The final product of the recommender system is a top-list of items recommended for the user ordered by an evaluated score that represents the preference of that item for the user. So highest the value, more interested the user will be. But to produce the right recommendations is not trivial and it's important to evaluate the results of the designed engine.

Therefore, once the recommender engine is developed, the main question after all work is done is: "*What are the best recommendations for the user?*". In order to answer this question, it's necessary to know what exactly a good recommendation means and how to known when the recommender system is producing them. Evaluation techniques are used to try to answer those questions.

6.2 Classification accuracy metrics

Classification metrics measure the frequency with which a recommender system makes correct or incorrect decisions about whether an item is interesting to the user. In the case of the developed recommender, items have true binary preferences (viewed or not viewed by the users), and therefore classification metrics are appropriated to evaluate the system.

The particular metrics that will be discussed are Precision and Recall, Kappa statistic and ROC Curves and the area underneath them.

6.2.1 Precision and recall

For the final the user of the recommender system the most important result is to receive an ordered list of recommendations, from best to worst. In fact, in some cases the user doesn't care much about the exact ordering of the list - a set of few good recommendations is fine. Taking this fact into evaluation of recommender systems, classic information retrieval metrics could be applied to evaluate those engines: Precision and Recall. These metrics are widely used on information retrieving scenario and widely applied to domains such as search engines, which return some set of best results for a query out of many possible results.

For a search engine for example, it should not return irrelevant results in the top results, although it should be able to return as many relevant results as possible. We could say that the 'Precision' is the proportion of top results that are relevant, considering some definition of relevant for your problem domain. The 'Recall' would measure the proportion of all relevant results included in the top results.

Adapting these metrics to recommender systems can be simplified as: “Precision is the proportion of recommendations that are good recommendations, and recall is the proportion of good recommendations that appear in top recommendations" [13]. This could also be stated in a different way:

- Precision is the proportion of the examples which truly have class x among all those which were classified as class x.

- Recall is the proportion of examples which were classified as class x, among all examples which truly have class x, i.e. how much part of the class was captured.

For recommender systems, a perfect precision score of 1.0 means that every item recommended in the list was good while a perfect recall score of 1.0 means that all good recommended items were suggested in the list.

One of the primary challenges to using precision and recall to compare different algorithms is that precision and recall must be considered together to evaluate completely the performance of an algorithm. It has been observed that precision and recall are inversely related and are dependent on the length of the result list returned to the user. When more items are returned, then the recall increases and precision decreases. Therefore, if the information system doesn’t always return a fixed number of items, we must provide a vector of precision/recall pairs to fully describe the performance of the system.

Several approaches have been taken to combine precision and recall into a single metric. One approach is the F-Measure used by Weka, which can be seen in the following equation:

$$F - Measure = \frac{2 \times P \times R}{P + R} \quad \textbf{(6.1)}$$

This equation combines precision and recall into a single number. The F-measure is a measure of a statistic test's accuracy. It considers both precision and recall measures of the test to compute the score. This could be interpreted as a weighted average of precision and recall, where the best F-Measure score has its value at 1 and worst score at the value 0. For a recommender system, it is considered a single value obtained combining both the precision and recall measures and indicates an overall utility of the recommendation list.

6.2.2 The Kappa Statistic

The Kappa statistic or Cohen's kappa coefficient is a statistical measure of inter-rater or inter-observer agreement for categorical items. In general, it is thought to be a more robust measure than simple percent agreement calculation since κ takes into account the agreement occurring by chance [14].

An inter-observer agreement can be measured in any situation in which two or more independent observers are evaluating the same thing. The Kappa statistic measures the agreement between two observers or raters who each classify N items into C mutually exclusive categories. The value of Kappa is calculated with the following equation:

$$\kappa = \frac{P_r(a) - P_r(e)}{1 - P_r(e)} \quad \textbf{(6.2)}$$

where $P_r(a)$ is the percentage agreement (e.g., between the recommender system and ground truth) and $P_r(e)$ is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly saying each category. If the observers are in complete agreement the value of kappa is 1 and if there is no agreement among them other than what would be expected by chance (as defined by $P_r(e)$), $\kappa = 0$.

In a recommender system, the kappa statistic measures the agreement of prediction with the true class, being kappa equal to 1 the complete agreement. Therefore, the Kappa statistic measures whether the proposed recommendations are better than just random guessing.

The closer to 1, the more recommendation power the system has. According to a classification proposed by Landis and Koch [15], values of kappa are characterized as:

Table 6.1 Characterization of kappa values

Kappa statistic	Classification accuracy
<0,00	no agreement
between 0,00 and 0,20	slight
between 0,21 and 0,40	fair
between 0,41 and 0,60	moderate
between 0,61 and 0,80	substantial
between 0,81 and 1,00	almost perfect

6.2.3 ROC curves and ROC area underneath the curve

An alternative to precision and recall to measure accuracy in the recommender system could be the ROC curve-based metrics. ROC stands for “receiver operating characteristic” and its model attempts to measure the extent to which an information filtering system can successfully distinguish between signal (relevance) and noise. The ROC model assumes that the information system will assign a predicted level of relevance to every potential item. Basically, the ROC curve is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied.

Some other concepts may be useful to understand the ROC curve computations. They are:

- The confusion matrix (or contingency table): it’s a square matrix which specifies the classes of the obtained results. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

- The True Positive Rate (TPR) is the proportion of examples which were classified as class x, among all examples which truly have class x, i.e. how much part of the class was captured. Simplifying, it’s the fraction of true positives out of the positives and is equivalent to Recall. In the confusion matrix, this is the diagonal element divided by the sum over the relevant row.

- The False Positive Rate (FPR) is the proportion of examples which were classified as class x, but belong to a different class, among all examples which are not of class x, which means the fraction of false positives out of the negatives. In the confusion matrix, this is the column sum of class x minus the diagonal element, divided by the rows sums of all other classes.

The ROC curve is created by plotting the TPR vs. the FPR at various threshold settings.

If the recommender system is perfect, the generated ROC curve will go straight upward until 100% of relevant items have been encountered, then straight right for the remaining items. If the recommender system is based in a random predictor, the ROC curve will be a straight line from the origin to the upper right corner. Similar to Precision and Recall measures, ROC curves assume a binary relevance. The recommended items can be either successful recommendations (relevant) or unsuccessful recommendation (non-relevant). One consequence of this assumption is that the ordering among relevant items has no consequence on the ROC metric – if all relevant items appear before all non-relevant items in the recommendation list, the generated ROC curve will be perfect.

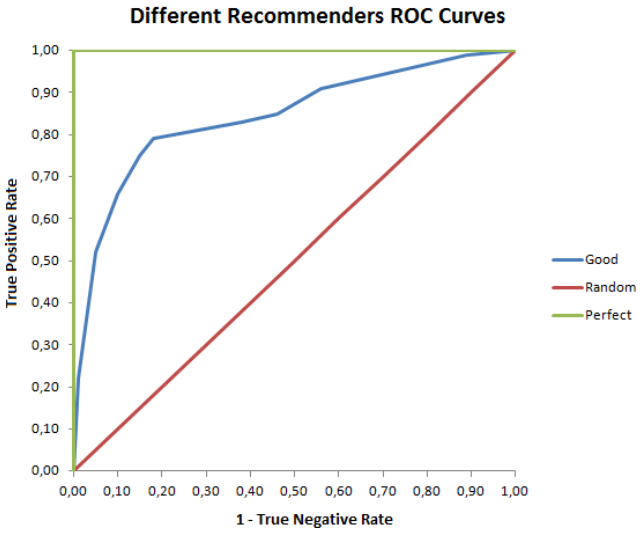

Fig. 6.1 ROC curves for different systems

A standard measure for system comparison is the area underneath the ROC curve (AUC), which can be obtained by numerical integration, such as, for example, the trapezoidal rule and is simpler to use when evaluating than the ROC curves. Theoretically, the higher the AUC, the better the system. The AUC can be used as a single metric of the system's ability to discriminate between good and bad items.

6.3 Testing results

In this section, the evaluation methods previously presented will be applied. Unfortunately, real data is not available since the web were the recommender will be used doesn't exist yet. Therefore, the accuracy calculated here is not the real life accuracy of the developed recommender system, but only an illustration using the data set described in the last chapter.

In order to be able to evaluate the developed recommender, the first 50 items (movies) of the data set provided in last Chapter will be used and the resulting data will be divided into training (first 4000 users) and testing (remaining 2040

users) sets. This means the recommender model will be generated for 50 items and 4000 users, and the tests will be performed in a 2040 users set.

The tests were performed using Weka explorer. The main results can be seen in the following tables:

Table 6.2 Basic evaluation measures

EVALUATION ON TEST SET:		
Correctly Classified Instances	34	68%
Incorrectly Classified Instances	16	32%
Kappa statistic	0,665	
Total Number of Instances	50	

Table 6.3 Fragment of the table with the detailed accuracy by class

Class	TPR	FPR	Precision	Recall	F-Measure	ROC Area
1	1	0,021	0,667	1	0,8	0,99
2	1	0,023	0,857	1	0,923	0,989
3	0	0	0	0	0	?
4	1	0	1	1	1	1
5	0	0	0	0	0	0,5
6	0,667	0,021	0,667	0,667	0,667	0,823
7	1	0,061	0,25	1	0,4	0,969
8	0	0	0	0	0	0,5
9	1	0	1	1	1	1
10	0,5	0	1	0,5	0,667	0,75
…	…	…	…	…	…	…
49	1	0	1	1	1	1
50	0	0	0	0	0	?
Weighted Avr.	**0,68**	**0,007**	**0,775**	**0,68**	**0,693**	**0,837**

In table 6.1 the accuracy of the recommender can be found in the form of percentage of correctly classified instances. In this recommender system trained with the data set already described, this percentage is of 68%. As explained in the previous section, a more robust measure than only this percentage would be the kappa statistic, which is in this case 0,665. Since the kappa is a chance-corrected measure of agreement between the classifications and the true classes, the value of kappa says if the recommendations are better than just randomly guessing items to recommend. In this example, a kappa of 0,665 means that the recommender system developed is more accurate than a

random engine. Using the classification presented in the last section, this value of kappa indicates that the system's accuracy may be considered substantial.

Looking at table 6.3, evaluation metrics as Precision, Recall, F-Measure and the area underneath the curve ROC can be found detailed for each class.

Looking to the weighted average results, an average precision of 0,775 means that 77,5% of the items recommended in the recommendations list were good items (as explain in the previous section), while an average recall of 0,68 means that 68% of the good items were recommended in the list. As a combination of precision and recall, the average weighted F-Measure obtained indicates that the developed recommender has an accuracy of 69.3%.

The TP and FP rates can also be used to evaluate the recommender. The True Positive Rate is equivalent to Recall, while the False Positive Rate gives the proportion of examples classified as class x but that belong to a different class. The closer to zero the FPR is, the better is the recommender. The weighted average value obtained of 0,007 is a good value for the FPR in a recommender system.

As described in the previous section, the ROC Area is a good indicative of recommender accuracy. The higher the ROC Area is, the better is the system. In the developed recommender, the weighted average ROC Area of 0,837 indicates that the system has is good in distinguish between good and bad items.

All these results are just an example of how the developed recommender system could be evaluated in future using real data. They don't represent the real accuracy of the developed system but are just a hint of how the real system will perform.

6.4 Refining the recommender engine using feedback

Recommendation systems can be evaluated and improved by letting the users express their needs, preferences or restrictions about each concrete item. For this to happen, the users' feedback must be collected during the recommendation process.

The feedback can be explicitly collected. In this case the users would be directly asked to classify the offered recommendations as relevant or not. In the developed recommender engine, one option to collect explicit feedback would be to ask the costumer to "like" or "dislike" the recommended deal.

Also, feedback can be implicit collect: if the user takes action on a recommended item, this action may be considered as positive feedback. In the developed recommender system, the way to collect explicit feedback would be:

- If the costumer views the recommended deal description, the system will understand it as a positive feedback, just as the “like” in the explicit feedback.

- If the costumer doesn’t click in the recommended deal, this would mean that the recommendation wasn’t right and would be interpreted as a negative feedback, just as the “dislike” in the explicit feedback.

The feedback information can be used to update and re-train the recommender and refine it. Adding the information feedback to the data, a new similarity matrix can be computed. If the feedback data adds relevant information to the system and not only noise, the new computed matrix should be more accurate than the older one. In order to compare the accuracy of the old and the re-trained systems, the metrics presented in the last section can be used and computed through the Weka explorer. Summarizing, the steps to refine the recommender engine using feedback are:

- Use the feedback data to re-train the recommender generating a new similarity matrix.

- Evaluate the new model and estimate its error (using the metrics previously presented). Compare it with the old model and if it has a lower estimated error, update the similarity matrix. Otherwise, keep using the former model. Note that’s important to use the same data set to evaluate both new and old matrixes in order to make a valid comparison.

Unfortunately, as there isn’t real feedback data available to the recommender system developed, the real update of the recommender system cannot be tested in this project.

CHAPTER 7. CONCLUSIONS

At the first chapter of this Thesis, the main goals of the project were clearly defined. They are in summary:

- Perform a benchmarking analysis comparing the web functionalities of the main web pages related to the daily deals concept.

- To choose one web functionality among the entire set of web functionalities analyzed that may possible add some value to the selling and re-selling deals web presented. The functionality I chose is the use of recommendation systems and my proposal at the end of Chapter 2 was to implement a recommender system to work through mailing and also in the web.

- Study recommendation systems and learn the main techniques to implement a recommender.

- Implement a proof of concept of a recommender system.

- Present a set of evaluation techniques that may be used to proof the implemented recommender.

All these objectives were accomplished as described in the different chapters, with the distinction that instead of implement a recommender system to work through mailing and also in the web, I preferred to focus only in web recommendations. As explained at the end of Chapter 4, I thought that the implementations would be very similar for these two contexts, being their main differences in characteristics that are out of the scope of this project.

The accomplishment of these objectives led to some results. Firstly, there are the results of the benchmarking analysis. As more detailed at Chapter 2, Section 2.2, this analysis result in some conclusions, being the most relevant to this project the one about the use of recommendations in the analyzed webs. Just a handful of players in the market are using it. Among these players are the giant e-commerce web sites Amazon and eBay which are using recommender systems as a tool to perform efficient cross-selling and up-selling. It's because of the conclusions in this benchmarking analysis that I decided to implement a recommender engine for an e-commerce web.

Secondly, there are the results of the recommender engine implemented. As explained in Chapters 5 and 6, unfortunately it wasn't possible to generate and evaluate the model with real data from the real life web, as it doesn't exist yet. However, the recommender engine was implemented and tested using the described data set MovieLens, which gives some results that may be interpreted as a hint of what would be the results in the real system. The results presented in section 6.5 of Chapter 6 says that the recommender works better than a random recommender and the percentage of "good" recommendations is

approximately 70%, which may be raised by increasing the number of recommendations made. This can be considered a positive indicative that the recommender engine would work well in the real environment.

To implement the recommender and get the results, a number of techniques and tools were necessary. A brief evaluation of these tools and techniques is made as follows:

- The literature available with techniques to implement recommendations systems is large and was satisfactory for this project.

- Respect the used evaluation tool, Weka, I found it very user-friendly and useful in this context. However, it's a tool thought to data mining for artificial intelligence in general, and not focused in recommender systems and maybe because of that its integration with the recommender is a little laborious.

- A data set was also necessary to implement the engine. There are a lot of trustful data sets available that can be used in recommender systems, although none of them contains e-commerce data. Even though the MovieLens data set doesn't contain e-commerce data, it could be adapted and was adequate to implement the model.

Personally, this project made me learn a lot of new concepts. I was not familiarized with recommendations systems before and I had to learn all the concepts and techniques from the beginning.

As a future work, I would say that it would be interesting to follow with the system implementation and develop the front-end application for the recommender, integrating it in the selling and re-selling deals platform to be developed.

BIBLIOGRAPHY

[1] Wikipedia – E-commerce [online], April 2012. Available online: <http://en.wikipedia.org/wiki/E-commerce>.

[2] SmartMoney [online], April 2012. Available online: <http://www.smartmoney.com/spend/family-money/10-things-daily-deal-sites-wont-say-1301404072442/#articleTabs>.

[3] Ho, Ricky. "Pragmatic Programming Techniques" [online], May – December 2012. Available online: <http://horicky.blogspot.com/2011/09/recommendation-engine.html>.

[4] Schafer, J. Ben. "The Application of Data-Mining to Recommender Systems". Available online: http://math-cs.cns.uni.edu/~schafer/publications/dmChapter.pdf.

[5] Abernethy, Michael. "Data mining with WEKA" [online], May – December 2012. Available online: <http://www.ibm.com/developerworks/opensource/library/os-weka2/index.html>.

[6] Schafer, J. Ben; Konstan, Joseph and Riedl, John. "Recommender Systems in E-Commerce". Available online: <http://www.win.tue.nl/~laroyo/2L340/resources/recommender-systems-e-commerce.pdf>.

[7] Linden, Greg; Smith, Brent; and York, Jeremy. Amazon.com recommendations: item-to-item collaborative filtering. Available online: < http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf >

[8] Alag, Satnam. Manning – Collective Intelligence in Action (2008).

[9] Rajaraman, Anand and Ullman, J. David. "Recommendation Systems", Chapter 9 in "*Mining of Massive Datasets*", pp. 277-309 (2011).

[10] Recommender systems – A Computer Science Comprehensive Exercise – Carleton College, Northfield, MN, [online], November 2012. Available online: <http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/collaborativefiltering.html>

[11] GroupLens Research Group Data Sets [online], September – November 2012. Available online: <http://www.grouplens.org/node/12>

[12] WEKA – Machine Learning Group at University of Waikato [online], May – December 2012. Available online: <http://www.cs.waikato.ac.nz/ml/weka/>

[13] Blog Artificial Intelligence in Motion [online], September – November 2012. Available online:
<http://aimotion.blogspot.com.es/2011/05/evaluating-recommender-systems.html>

[14] Wikipedia – Cohen's kappa coefficient [online], November – December 2012. Available online:
<http://en.wikipedia.org/wiki/Cohen's_kappa>

[15] Landis, J.R.; Koch, G.G. "The measurement of observer agreement for categorical data". (1977).

[16] Deshpande, Mukund and Karypis. "Item-based top-N Recommendation Algorithms". (2004). Available online:
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.1279>