UNIVERSITY OF NICE - SOPHIA ANTIPOLIS
UNIVERSIDAD DE LOS ANDES

DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University of Nice - Sophia Antipolis

and to obtain the title of

Ph.D. in Engineering

of the Universidad de los Andes - Bogotá Colombia
Specialty : COMPUTER SCIENCE

Defended by

Andrés Dario MORENO BARBOSA

Privacy-enabled scalable recommender systems

Thesis Advisors:
Harold CASTRO and Michel RIVEILL

defended on December 10, 2014

Jury :

Advisors :	Harold CASTRO	-	Universidad de los Andes
	Michel RIVEILL	-	Université de Nice
President :	Frederic PRECIOSO	-	Université de Nice
Examinators :	Claudia Jiménez-Guarin	-	Universidad de los Andes
	Florent MASSEGLIA	-	INRIA Sophia Antipolis

A mi familia
A Diana

Acknowledgments

I would like to thank my advisors Harold Castro and Michel Riveill for their invaluable guidance during this project and for their general support during these years. Without their encouragement to pursue the phd this thesis would not have been possible.

I would also like to thank Francisco Rueda and Claudia Jimeéz for providing guidance during my masters and doctoral studies.

Thanks to Universidad de los Andes and Departamento de Ingenieria de Sistemas y Computación for offering the graduate student assistant position during my stance at Universidad de los Andes, teaching has been a highly rewarding experience. Thanks to I3S laboratory and the members of the GLC team for their support during my stance in France. Finally thanks to Colfuturo for providing financial support during the thesis.

Many thanks to the close friends made during my stance at Université de Nice, being abroad far from family and friends was at times overwelmening, but much easier with your help.

Thanks to the close friends made at Universidad de los Andes. Thanks for your friendship and the warm environment you have provided these years. Without your help and advice it would have been impossible to finish this work.

Agradecimientos

Muchas gracias a mi familia. Particularmente a mi papá , a mi mamá y a mi hermana, a ustedes dedico este trabajo. A Diana por su ayuda, paciencia, consejo y compañia durante estos años.

Contents

List of Figures

List of Tables

Part I

Introduction and related work

Chapter 1

Introduction

Contents

In this chapter the problem motivation, research goals, main contributions and structure of the thesis are presented.

1.1 Motivation

Electronic content is ubiquitous in our daily lives. Several factors such as the development of Web 2.0 technologies, the increased access to mobile devices and the deployment of mobile networks has undoubtedly augmented the amount of information easily available to users. Given the limited attention span of the user and the extensiveness of the available streams of information ready to be consumed, automatic systems must be available for the user to prioritize, suggest or screen content suitable for the user interests and situation.

One of the most popular initiatives created to solve the information overload problem are *Recommender Systems* [Adomavicius 2005]. Recommender Systems are information filtering systems that use the historical information about the user (what the user has considered relevant or irrelevant on the past, among other information) to build and accurate representation of the user's interests that is used to predict the relevance of a large collection of available items for a specific user. Recommendation systems are used by several online retailers, online content streaming services and social networking sites to improve the user's experience of their services by automatically filtering their content or offers of items to the ones most likely to interest the user.

Generally speaking, recommender systems can be classified into two categories: *Content Based* and *Collaborative Filtering*. The former category relies on the definition of explicit features that describe the item domain and assigns them weights to describe the affinity between the feature and the item. For example in the movie domain, items can be described by features such as the genre to which they belong, the director, writer and actors that take part in the movie. On the other hand Collaborative Filtering is content-agnostic and relies on correlations between

users and items based on the historical consumption patterns between the users and the items. It has been shown generally that Collaborative Filtering methods present better results than Content-Based [Pilászy 2009], however due to inherent shortcomings of single approaches, a better predictive performance is achieved by developing a model that integrates different paradigms (Hybrid approaches).

To keep their users satisfied, personalization services that operate recommendation methods should present relevant recommendations even when the number of users, items and user-item interactions in the system increase. As it will be shown in Chapter 2, **the computational complexity of Collaborative Filtering methods for keeping a user profile up to date depends directly on the number of users and items available in the system and the amount of registered user-item interactions**. Current large scale personalization systems such as Netflix [Netflix 2013] (a content streaming service of movies and series) has an estimated number of users of 44 million[1] while the number of available items to watch fluctuates around 13000 titles[2]. The number explicit user-item interactions (assigned ratings) is estimated at 5 billion ratings [Schelter 2013].

To account for these large numbers, Recommender System's adopters employ the support of cloud computing frameworks. Recommender Systems are now highly scalable solutions that are able to: (1) gather and store as much information as possible about users and items supported by the current availability of cheap storage, (2) apply computational intensive algorithms to train recommendation models that scale up to the size of the collected data and (3) use the trained models to adequately answer to a large amount of recommendation requests. However, as it will be shown in Chapter 3 **the current architecture of data gathering and processing of recommender systems places a conflict with users**

Following the definition given by [Foner 1999], *privacy* can be defined as the ability of an individual to protect *the disclosure of personal information to third parties who are not intended recipients of the information.* While users trust recommender engines to use their information for filtering or personalization purposes, it will be argued that the centralized consolidation of information increases the likelihood of misuse of the user information, misplacing user trust.

A question that arises after this claim is: *why information gathered and processed by a recommendation system is privacy-sensitive?* After all, due to the availability of personal micro blogging and social networks, users seem avid to share their information with others. Opinions given by users on items reveal at a great extend the personality of the user, opinions on items might reveal political inclination, sexual orientation, physical or mental treatments the user is taking or religious inclination of a particular user. An iconic case of how important are personal opinions on items in recommender systems came with the *Doe Vs Netflix* class

[1] 2013 Annual report, http://files.shareholder.com/downloads/NFLX/3461178757x0x748407/76a245dc-3314-401c-baba-ed229ca9145a/NFLX_AR.PDF [Accessed August 2014]

[2] http://www.fastcompany.com/1830524/amazon-massively-inflates-its-streaming-library-size [Acceded August 2014]

action lawsuit [Singel 2009], when ratings from users were de-anonymized after being made public by the *Netflix Prize Competition* [Netflix 2009]. The plaintiff claimed that:

> ...information tending to identify or permit inference of her sexual orientation constitutes sensitive and personal information. She believes that, were her sexual orientation public knowledge, it would negatively affect her ability to pursue her livelihood and support her family and would hinder her and her childrens' ability to live peaceful lives within ...(her)... community.

The aim of privacy-enabled recommendation systems is to give users tools to protect their privacy and keep the choice to themselves if they want to reveal their information.

1.2 Research objective

Traditional Recommender Systems are usually evaluated in terms of their predictive performance [Shani 2011], for this end recommendation systems use increasingly more complex models and include more information about users which places a tradeoff between the computational complexity of training the model and the predictive performance the model can attain. In this work, a third axis is introduced into these traditional concerns: Privacy.

Unfortunately, as it will be shown in Chapter 3, protecting user privacy imposes architectural restrictions to the data storage and processing tasks, having either to perform costly cryptographical operations on the user profile that have an impact on the scalability of the system, or prevent the usage of traditional cloud computing architectures, limiting the predictive performance of the system when compared to privacy-agnostic recommendation systems.

Research objective

The main topic of the thesis is to explore the tradeoff between the predictive performance, user privacy and the system scalability evaluated in terms of the computational complexity of the recommendation system (Figure 1.1). Particularly, the research objective is to create a new recommendation system that keeps into account privacy without sacrificing the scalability of the solution.

1.3 Thesis contributions and document outline

In order to attain the research goal expressed in the previous section, the following contributions are presented in this document:

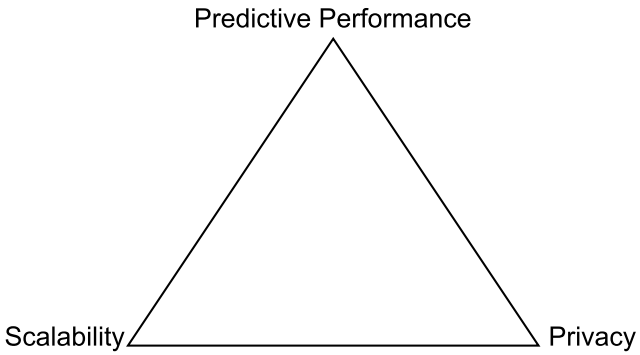


Figure 1.1: Three axis objective

- An introduction on Recommender Systems is presented in Chapter 2.The chapter presents the existing paradigms for recommendation, how they are evaluated and the tradeoff between predictive performance and scalability that exists in these kind of systems.

- A privacy-enabled recommender system survey is presented in Chapter 3. The presented survey analyzes the chosen strategies used by recommender systems to keep the privacy of the users. These strategies are analyzed in terms of the *exposure* risks and in terms of the scalability of the approach. From these strategies some design choices are made in order to create a new highly.scalable privacy-enabled recommender system.

- In Chapter 4 a highly scalable client-based approach for Collaborative Filtering recommendation is presented. The system keeps the information about the user at the client-side, and doesn't reveal the ratings of the user to the recommendation server. Placed under the *online learning* setting, the system has a low computational complexity when updating either the user or item representations at both training and prediction phases, scaling up to the number of items present in the system and the number of predictions the agent must make over time.

- In Chapter 5, the Collaborative Filtering algorithm presented is extended into an Hybrid one by the use of a Content Based recommender system. The hybridization runs both models in parallel and outputs a *weighted* prediction of both models according to their historical regret. The hybrid model allows the system to improve its predictive performance on the cold-start scenario, while keeping a low computational complexity at both training and prediction phases.

- In Chapter 6 the proposed hybrid system is analyzed in terms of the information that is exposed to the recommendation server. The privacy of the recommender system is increased by adding random noise perturbation to the output of the recommender using differential privacy notions [Dwork 2006], so that an

attacker can't simulate the internal state of the client-side agent. Finally a keyword based strategy is used as a mean to keep the client-side agent from reporting back information on items the user doesn't want to be linked with. The hybrid approach proves useful on the privacy-protective setting as well.

Chapter 2

Recommender systems: Related work and evaluation

Contents

In this chapter a revision of recommendation systems will be presented. In order to review them we will present a review of existing recommendation technologies (Section 2.1), how a recommender system can be evaluated in terms of its predictive accuracy and scalability (Section 2.2). This will establish the bases of the design choices that privacy-enabled models must make and their tradeoffs.

2.1 Recommender systems

Recommender systems are information filtering systems that present relevant item suggestions to users from a large collection of possible items, for this purpose recommender systems traditionally rely on the historic interaction of the user with the system to build and accurate representation of the user's interests. We understand *relevance* defined in [Borlund 2003] as the: *"utility or usefulness of the information in regard to the user's task and needs"*. The recommendation process can be defined formally as follows:

Let U=$\{u_1, u_2 \ldots ,u_m\}$ be the set of m users available in the system and let I=$\{i_1, i_2 \ldots ,i_n\}$ be the set of n possible data items that are available for the users. A recommender system can be viewed as a mapping that calculates the relevance of an item for a user. *RelevanceEstimation:* $\hat{r}(U \times I) \rightarrow \mathbb{R} \cup \{null\}$. Recommender systems use the relevance function to select a subset of items from the set I not seen before by the user that maximizes the perceived relevance, rank lists of items

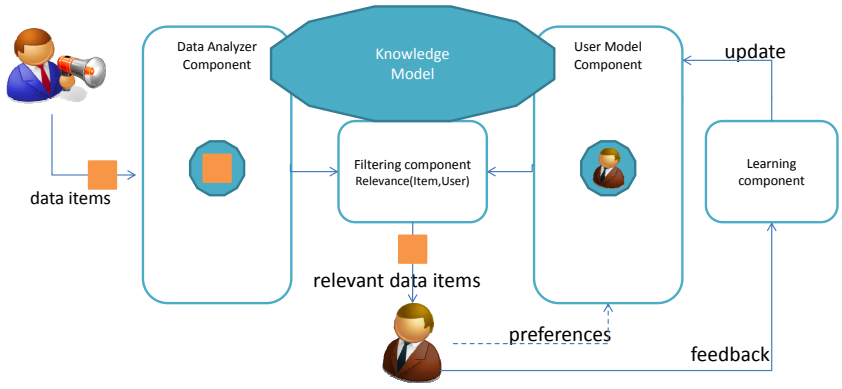


Figure 2.1: Generic CB filtering system

based on their relevance, or screen out irrelevant items from a stream of incoming information.

There are two types of information recommender systems use in order to predict the relevance of an item for a user: *Explicit feedback* consists on the direct feedback of the user on an item. Ratings are an usual representation of the user opinion of a user for an item, for example a numerical rating (1 to 5), a rating on a likert scale (`Strongly disagree`, `Disagree`, `Neither agree nor disagree`, `Agree`, `Strongly agree`) or binary ratings (`like`, `dislike`). *Implicit feedback* on the other hand is the collection of actions users exert on items, these actions indirectly reflect the opinion of the user on the item, for example a user can `view`, `click` or `buy` an item.

According to [Adomavicius 2005], recommender systems can be divided into three general categories according on how the systems employ the user information to calculate the relevance of an item: Content based filtering (CB) which uses the features or characteristics of the items to find out the relevance for the user, collaborative filtering (CF) which uses only the opinions of the users on the items, and hybrid systems(HS) that use an ensemble of different systems. Other authors such as [Burke 2002] identify other categories such as knowledge based recommender systems and demographic based recommender systems but since they rely heavily on user and item features we classify them under the CB approach.

2.1.1 Content based filtering (CB)

Content based (CB) filtering systems operate under the assumption that the user will like similar items to the ones she has liked in the past. CB systems extract the characteristics or features of the data items and use these characteristics to represent the item and the user under a common knowledge model.

To calculate the relevance function $Relevance(U \times I)$ a generic CB filtering system shown in figure Fig.(2.1) is in charge of the following tasks [Hanani 2001]:

- **Data item representation:** The data analyzer component creates a data

item representation under a knowledge model that reflects the relevant characteristics of the data item that are useful for the filtering component.

- **User representation:** The user model component creates a user profile that reflects the current user's situation and desires and represents it under a knowledge model. To properly represent the user's situation and interests explicit information from the user can be used.
- **Matching:** The filtering component is in charge of using the item and user representation to predict the relevance of the item for the said user.
- **Learning:** The learning component keeps up to date the user model representation based on the feedback received by the user.

Based on the work on document representation in Information Retrieval, one of the first knowledge representations available for recommender systems was based on the Vector Space Model [Salton 1975]. In this model both item and users are represented by a vector $\mathit{ContentBasedProfile} = (w_0, w_1 ..., w_{\|Con\|}) \in \mathbb{R}[0,1]^{Con}$ where Con is the set of features that describe the knowledge domain of the items in the set I. For each coordinate w_c in the vector, its weight represents a numerical indication of the degree of affinity between the item and the concept represented by the coordinate for a item profile, or the degree of interest of the user towards the concept in an user profile.

To find out the weights for each coordinate in the item's vector, the term frequency - inverse document frequency ($TF\text{-}IDF$) [Salton 1988] strategy is used. This strategy consist on assigning a weight for a concept proportional to the number of times the concept appears in the document (TF) and inversely proportional to the number of documents it appears (IDF). This strategy has been used on content bases systems with with text-based items such as web pages [Balabanović 1997] and news [Lang 1995].

On the user profile side, as the user expresses her opinion on items, relevance feedback methods are used to continuously update and refine the user's profile weights, for example Rocchio's algorithm [Buckley 1995] is commonly used in CB filtering learning. Let x_u^t be the *ContentBasedProfile* for user u at time t, $D^+ \subset I$ be a set of items the user has manifested positive feedback, and $D^- \subset I$ a set of items the user has expressed negative feedback, x_u^t is updated according to the next formula:

$$x_u^t = \alpha x_u^{t-1} + \beta \left(\frac{1}{|D^+|} \sum_{y_i \in D^+} y_i \right) - \gamma \left(\frac{1}{|D^-|} \sum_{y_i \in D^-} y_i \right) \tag{2.1}$$

After having learned the user and item profile, vector similarity functions are used to assess the relevance of an item. To find out the relevance of an item one of the most used heuristics the cosine similarity. If x_u is the *ContentBasedProfile* for user u and y_i is the *ContentBasedProfile* of item i, then the relevance of item i for user u is:

$$cos\,(x_u \times y_i) = \frac{x_u \cdot y_i}{\|x_u\| \|y_i\|} \tag{2.2}$$

Another strategy for learning a user profile is to make use of the vector representation of the items and the feedback to apply machine learning classifiers that categorize items into two classes (relevant, not relevant) by training a classifier with the items the user has seen, for example: Naive Bayesian classifiers [Pazzani 1997], artificial neural networks [Hsu 2007] and support vector machines [Oku 2006].

Other existing knowledge models that extend the Vector Space Model were developed to address the problems present with the textual extraction of features used in the TF-IDF strategy, particularly (1) The string matching process can be susceptible to polysemy (single words with multiple meanings) or synonymy (multiple words with the same meaning), and (2) the TF-IDF strategy is not applicable to items with little or no text content. Keyword-based profiles restrict the possible set of concepts to a controlled vocabulary [Lieberman 1995], this strategy can be used for non-text based items where weights for each concept can be manually assigned by a domain expert(i.e. in the music domain Pandora [1]) or automatically by domain specific automatic tools (i.e. images and video [Dasiopoulou 2011]).

Since using automatic tools for item categorization is not always available, efficient or maintainable due to the size or characteristics of the available item set, some systems use their users to help catalog and annotate their items. In the Web 2.0 paradigm, the user is not only a consumer of content but now the user is allowed to publish and edit content. In particular Social Tagging Systems(STS) [Marinho 2011] allow their users to describe data items using arbitrary keywords called *tags*. The collection of the collaborative created tags is called a *folksonomy*. Folksonomies have been used with success by hybrid filtering approaches [Zhen 2009].

On the other hand, approaches to remove ambiguity in keyword and tag based systems introduced the use of ontologies for user profiling. Ontologies are a formal representation of the concepts present in a knowledge domain and the relations between them. Middleton et. al. [Middleton 2004] stated that ontology based profiles encompass the adequate formality and granularity to describe a data item, reducing the conceptual gap between the data item semantics and the chosen representation which results in an improvement of the accuracy of the system when compared to keyword-based approaches.

As defined by [Ehrig 2004], an ontology is defined as a data structure $\mathcal{O} := (C, T, \leq_C, R, A, \sigma_R, \sigma_A, \leq_R, \leq_A, In, V, \mathcal{L}_C, \mathcal{L}_R, \mathcal{L}_A)$ where C, T, R, A, In, V are the sets that contain the classes, data types, binary relations, attribute relations, instances and data values present in the knowledge domain described by the ontology. $\leq_C, \leq_R, \leq_A$ are the partial orders that define the class, relation and attribute hierarchy. $\sigma_R : R \to C \times C$ is the function that provides a signature for a relation between classes. $\sigma_A : A \to C \times T$ is the function that provides a signature for an attribute for a class. $\mathcal{L}_C : C \to In$ is the instantiation function. $\mathcal{L}_T : T \to V$ is the data type instantiation function. $\mathcal{L}_R : R \to In \times In$ is the relation instantiation function and $\mathcal{L}_A : A \to In \times V$ is the attribute instantiation function.

The current W3C recommendation for specifiying ontologies is OWL2 [W3C 2012].

[1] www.pandora.com

The OWL2 language accounts for the sufficient expressiveness to account for the structure defined by [Ehrig 2004].

Under this knowledge model, the set of concepts that describe the knowledge model are the classes and instances present in the ontology $Con = \{C \cup In\}$. As in the Vector Space Model, the weights on the vector represent the relatedness between the item and a concept. One of the most important advantages of using ontologies as a knowledge model is that they allow the system to understand the relationships between the concepts that describe the knowledge domain of the items when learning a user profile. This allows the learning component to modify not only the weights of the concepts directly involved in the item profile description but to activate related concepts as well, one strategy to achieve this is to use the constrained spreading activation technique [Crestani 1997].

After a user has manifested a preference for an item (relevant, not relevant) the user profile is modified as in Rocchio's feedback algorithm [Buckley 1995] (Equation 2.1) not only on the weights expressed directly on the item profile but other weights are activated on the user profile by using the relations (R and the partial order $\leq_C$) that relate the activated preference to other preferences present in the knowledge domain.

Let Con_c be a concept describing an item a user u has marked as relevant at time t, and Con_d a concept that is connected through a relation to the concept Con_c. The value for w_d on the user's profile x_u is given by the following equation [Papadogiorgaki 2008]:

$$x_{u_d}^{\ t} = x_{u_d}^{\ t-1} + \mathcal{W}\left(Con_c, Con_d\right) \times x_{u_c}^{\ t} \times \partial \tag{2.3}$$

Where ∂ is a decay factor proportional to the time passed since the last time the preference was updated and Con_j and Con_k are the concepts related by a relation in the ontology. Coordinate k is updated if: $(\exists r | Con_j \in C \wedge Con_k \in C \wedge r \in R : \sigma_R^{-1}(Con_j, Con_k) = r\ )$ if both Con_j and Con_k are classes, if $(\exists r | Con_i \in In \wedge Con_j \in In \wedge r \in R : \sigma_R^{-1}(Con_j, Con_k) = r)$ if both Con_j and Con_k are instances or $L_C^{-1}(Con_j) = Con_k$ if Con_k is instance of the class Con_i.

$\mathcal{W} : (Con \times Con) \rightarrow \mathbb{R}[0, 1]$ is a function calculating the semantic relatedness between the concepts connected by a binary relation or by an instantiation relation. This function has been established as a fixed value over the the partial order $\leq_C$ [Middleton 2004] [Sieg 2007] [Blanco-Fernández 2008], ontological similarity measures [Vallet 2006] or by an statistical approach using machine learning algorithms to learn the factor value based on frequency of occurrence of both concepts [Jiang 2009]. Other implicit information from the user-item interaction can be included into the ∂ parameter.

Although CB systems are simple, easy to implement and its easy to explain why an item has been classified as relevant, several researchers such as [Adomavicius 2005] have noted the limitations of these kind of systems in their filtering performance, in particular researchers have remarked the following limitations :(1) *Limited content analysis*: In order to have a good representation of data items, data items must

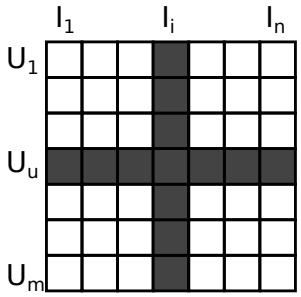


Figure 2.2: Matrix V for modeling user-item information in CF, u row is the user profile of user U_u and the i column is the data item profile of item I_i

be characterized by features. If this feature extraction of data items is difficult or impractical the representation of the data items will not be accurate and a conceptual gap between the representation and the real features will occur, this inaccurate classification of items will affect the accuracy of the filtering system. (2)*Overspecialization*: A CB system will only classify as relevant data items that are similar to the ones that the user has classified as relevant in the past. This means that the system cannot predict the relevance of a data item that is unlike the ones the user has seen. (3)*New user problem*: A user has to express her opinion on a sufficient number of data items in order to build an useful user profile.

2.1.2 Collaborative filtering (CF)

Collaborative filtering (CF) systems are IF systems that operate under the assumption a user will like the same data items that other users have liked in the past. Generally speaking, CF systems try to predict the relevance of a data item for a user by taking into account the opinion that other similar users have manifested about that item instead of taking into account the features of the data item. CF solves some of the problems of CB approaches: It doesn't need to know the features of the data items, therefore is not prone to the limited content analysis. Also it can detect the relevance of a item that is very different from what the user has seen, reducing the impact of the overspecialization problem.

Collaborative filtering algorithms subsequent representation of the users and items information is a matrix V of size $m = |U| \times n = |I|$ as seen in Fig.(2.2): The rows on the V matrix represent the user profiles, each user ($u \in U$) has a profile defined as a vector over a vector space of size $|I|$, each coordinate of the vector $w_i \in \mathbb{R}^n \cup \phi$ is registered as the opinion of the user for the item I_i. Most collaborative filtering systems use ratings as a way of registering the opinion (for example a scale of 1 to 5 is used $\mathcal{O} = \{1, 2, 3, 4, 5\}$). Subsequently the columns of the V matrix represent the item profiles, each item has a profile defined as a vector over a vector space of size $|I|$ and each coordinate of the vector w_i is calculated as the opinion of the item expressed by user U_i.

One way of classify collaborative filtering approaches by the way they use the

ratings in order to generate relevance predictions: *Neighborhood-based systems* use similarities of users or items directly from the information included on rating matrix V in order to produce a relevance prediction, on the other hand *model based systems* go through an offline process to aggregate the information registered on the rating matrix V to build a model that will be used to produce a relevance prediction.

Neighborhood-based systems can be further classified as well under two categories:*user-based CF* calculates the relevance for a data item by taking into account the opinion on that item of the k most similar users that have expressed an opinion about the item. *item-based CF* takes into account the opinion of the user to the most similar data items to the active item.

User-based CF systems [Resnick 1994] calculate the relevance of an item by two steps: The system selects the users that have rated the item in the past, then from this subset the system finds the most similar users to the active one, once this set is established (the neighborhood of the user) the relevance is calculated as the weighted average opinion of the users of the neighborhood for that item.

The neighborhood $N(U_u, I_i)$ is defined as the set of k user profiles that have rated item i that maximize the similarity between the user profile of U_u with the rest of the available user profiles. The relevance of the item I_i for a user U_u is calculated as the weighted average opinion on that item for the user profiles present in the neighborhood $N(U_u, I_i)$:

$$\hat{r}(U_u \times I_i) = \frac{\sum_{x \in N(U_u, I_i)} x_{vi} \times sim(x_u, x_v)}{\sum_{x \in N(U_u, I_i)} sim(x_u, x_v)} \tag{2.4}$$

Where the sim function is a distance measure between the profile representation(for example the cosine measure presented in equation 2.2) and x_{vi} is the opinion of user v on item i.

Another similarity function widely used in the CF scenario is the Pearson correlation. If x_u is the CF user profile for user u and x_v is the CF user profile for user v, then the Pearson correlation of both users is:

$$Pearson(x_u \times x_v) = \frac{\sum_{j=1}^{l} (x_{uj} - \overline{x_u})(x_{vj} - \overline{x_v})}{\sqrt{\sum_{j=1}^{l} (x_{uj} - \overline{x_u})^2 \sum_{j=1}^{l} (x_{vj} - \overline{x_v})^2}} \tag{2.5}$$

where $\overline{x_v}$ is the average rating given by the user and the index j is used only on the l common items on both profiles.

Item-based CF [Sarwar 2001] is similar to user-based ones but instead of creating a neighborhood of similar users, it creates a neighborhood of similar items based on their item profile, the relevance of the item for the user is the average opinion of the user about the items in the neighborhood of the active item. Even tough User and Item-based CF have the same computational complexity, item based is preferred in systems where the set of data items is relatively static when compared to the rate of change of users, for example in e-commerce models such as Amazon.com [Linden 2003].

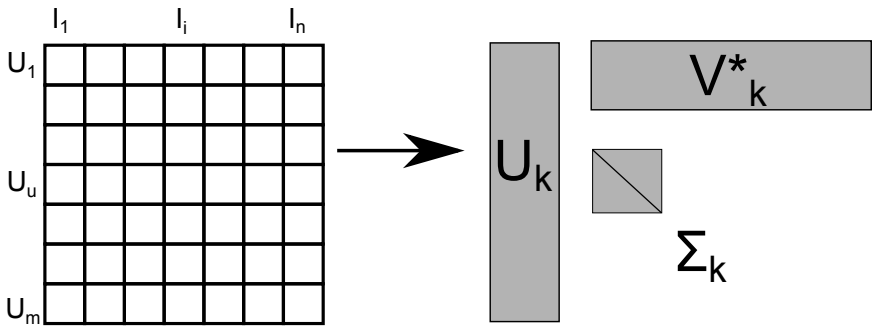


Figure 2.3: Matrix V is reduced using SVD intro matrices U, Σ and V^*

The neighborhood of a data item I_i can be defined as the set of data item profiles for which the user U_u has expressed an opinion $M(I_i, U_u)$ that maximizes the similarity between the data item profile of I_i and the set of item profiles $M(I_i, U_u)$. The relevance of item I_i for user U_u is calculated as the average of the opinion of the user for the data items in set $M(I_i, U_u)$:

$$\hat{r}(U_u \times I_i) = \frac{\sum_{y \in M(I_i, U_u)} (y_{j_u} \times sim(y_i, y_j))}{\sum_{y \in M(I_i, U_u)} sim(y_i, y_j))} \tag{2.6}$$

Where the *sim* function is a distance measure between the item profile representations and y_{j_u} is the opinion of user U_u about item I_j.

Rather than colsulting the opinion of similar users or items at the moment of prediction, model based systems learn the parameters of a predictive model from the user-item matrix V that is later used for predictions. First approaches based on linear algebra matrix factorization using singular value decomposition (SVD) [Sarwar 2002] which maps the original matrix into another space of dimensionality significantly smaller than the original one. A SVD factorization (as seen in figure 2.3) consists in finding a low rank approximation of the original V matrix by finding three matrices such that their multiplication reconstructs the original matrix: $SVD(V) = U_{m \times k} \times \Sigma_{k \times k} \times {V^*_{n \times k}}^T$ where k is significantly smaller than m and n.

Taking into account that the row u of matrix U is the representation of the user profile under a lower dimensionality and the row i of V^* is the representation of the item profile under a lower dimensionality, once the matrix is factorized the relevance is calculated by reconstructing the information of the user and item.

$$\hat{r}(U_u \times I_i) = average(I_i) + (U \cdot \sqrt{\Sigma^T})_u \cdot (\Sigma \sqrt{V^{*T}})_i \tag{2.7}$$

Following this work, other approaches have been adapted to separate different signals or effects that build up the rating prediction. Researchers have found that the global average of ratings (μ) and the *bias* or deviation from the mean that are observed for each user (b_u) and each item (b_i) are fundamental elements of the relevance prediction that must be included into the prediction model. Matrix factorization models build up two matrices representing the user ($X_{m \times k}$) and the

item ($Y_{n\times k}$) under a lower dimensionality such that $V \approxeq \mu + b_\star + XY^T$ [Koren 2008]. The relevance prediction of this model is given by the following equation:

$$\hat{r}\left(U_u \times I_i\right) = \mu + b_i + b_u + \left(x_u^T y_i\right) \tag{2.8}$$

In order to learn the parameters of the bias and the vector for each user, a least squares optimization is done to minimize the error over the entries of the matrix V that are known:

$$\min_{b_\star, x_\star, y_\star} \sum_{V_{ui} \neq null} \left(V_{ui} - \mu - b_i - b_u - x_u^T y_i\right)^2 + \lambda \left(\|x_u\|^2 + \|y_i\|^2\right) \tag{2.9}$$

These parameters can be learned using an alternating least squares strategy where one matrix (X or Y) is fixed and the parameters of the other one are adjusted [Bell 2007], or by an stochastic gradient descent technique popularized by [Funk 2006].

An extension to include more information for users is added in [Koren 2008]. The SVD++ algorithm introduces another set of factors to the items to account for the implicit information of user-item interaction. Each item is represented by an extra vector z_i that is used by the prediction rule to represent the items the user has rated into her profile. Let $R(u)$ the set of items the user u has rated, the prediction under the SVD++ model is given by the following equation:

$$\hat{r}\left(U_u \times I_i\right) = \mu + b_i + b_u + y_i^T \left(x_u + \left|R\left(u\right)\right|^{-\frac{1}{2}} \sum_{j \in R(u)} z_j\right) \tag{2.10}$$

This model can be extended easily to account for other types of implicit information different than using the information about the `rated` implicit action, a general model that includes other implicit information based on the SVD++ criterion. Let $\mathcal{I}$ be the set of implicit information about users present in the system, the generalized version is given as follows:

$$\hat{r}\left(U_u \times I_i\right) = \mu + b_i + b_u + y_i^T \left(x_u + \sum_{w \in \mathcal{I}, j \in R(u)^w} \alpha^w z_j^w\right) \tag{2.11}$$

Other extension to matrix factorization is introduced in [Koren 2011] to account for the ordinal nature of the ratings: In addition to the parameters learned in equation 2.9, a set rating thresholds are learned. Given the possible set of ratings as $\mathcal{O}$ (e.g: $\mathcal{O} = \{1, 2, 3, 4, 5\}$), for each user a set of thresholds $(t_1^u \leq t_2^u \ldots t_{\mathcal{O}-1}^u)$ is used to map the prediction of an inner model $\hat{X}_{ui}(\Theta) \rightarrow \mathbb{R}$ to one of the possible ratings. Considering the probability of a prediction $\hat{r}\left(U_u \times I_i\right)$ taking a specific value in $\mathcal{O}$ as $P\left(\hat{r}_{ui} = \mathcal{O}_r | \Theta\right) = P\left(\hat{X}_{ui}(\Theta) \leq \mathcal{O}_r | \Theta\right) - P\left(\hat{X}_{ui}(\Theta) \leq \mathcal{O}_{r-1} | \Theta\right)$ and taking the probability of an estimation being less or equal than a specific value in $\mathcal{O}$ as $P\left(\hat{r}_{ui}(\Theta) \leq \mathcal{O}_r | \Theta\right) = 1/(1 + \exp(\hat{X}_{ui}(\Theta) - t_r))$. The optimization criterion for calculating the parameters is the following:

$$\max \sum_{u \in U, i \in I, r \in \mathcal{O}} \ln P\left(\hat{r}_{ui}(\Theta) = \mathcal{O}_r | \Theta\right) - \lambda_\Theta \|\Theta\|^2 \tag{2.12}$$

The threshold parameters are learned using a stochastic gradient ascent strategy maximizing the log likelihood of the parameters over the known ratings of the matrix V.

Instead of focusing on the rating prediction task, other works optimize a ranking of items for each user. Rendle et al. [Rendle 2009] propose to reconstruct a personalized total order $>_u \subset I^2$ over the total set of items based on the partial views of $>_u$ seen on the user-item interaction log, either by observing the ratings of matrix V to infer a implicit action on the item or by using only implicit information. Taking Θ as the parameters of the model (e.g $X_{m \times k}$ and $Y_{n \times k}$) and the function $\hat{X}_{uij}(\Theta) \rightarrow \mathbb{R}$ as a arbitrary real function that describes the order between items i and j for user u (e.g $\hat{X}_{uij}(\Theta) := x_u^T y_i - x_u^T y_j$), the optimization criterion to calculate the parameters is the following:

$$\max \sum_{V_{ui} \neq null, V_{uj} \neq null, V_{ui} > V_{uj}} \ln\left(\sigma\left(\hat{X}_{uij}(\Theta)\right)\right) - \lambda_\Theta \|\Theta\|^2 \tag{2.13}$$

Where σ is the logistic sigmoid function. The learning algorithm is expressed also as a stochastic gradient learning method.

A close approach to this one is presented in [Shi 2012], where the optimization criterion is also based on a ranking metric; the Mutual Reciprocal Rank (MMR). The mutual reciprocal rank evaluates the quality of a process that produces an ordered list of responses (in this case a recommendation list ordered from the most relevant to the least relevant). The MMR is defined as the inverse of the position of the first correct result on the recommendation list. For a user u, her reciprocal rank is defined as $RR_u := \sum_{i \in I} \frac{\hat{Y}_{ui}}{\hat{R}_{ui}} \prod_{j \in I} \left(1 - \hat{Y}_{ui} \mathbb{I}\left(\hat{R}_{uj} - \hat{R}_{ui}\right)\right)$ where $\hat{Y}_{ui}$ is a binary function that tells if item i is relevant for user u, $\hat{R}_{ui}$ is the rank of item i for a user u and $\mathbb{I}$ and indicator function that is 1 if the parameter is true or 0 otherwise. The previous formulation of the reciprocal rank cannot be applied as a optimization criterion using gradient methods since is non-smooth and non tractable due to the multiplicative function across all items, therefore a substitution function has to be applied. By approximating $\mathbb{I}\left(\hat{R}_{uj} - \hat{R}_{ui}\right) \approx \sigma\left(x_u^T y_j < x_u^T y_i\right)$ and $\frac{1}{\hat{R}_{ui}} \approx \sigma\left(x_u^T y_i\right)$ and reformulating the optimization criterion to avoid the product of sums using the Jensen Inequality [Pentland 2001], the optimization criterion to calculate the parameters is the following:

$$\max \sum_{u \in U} \sum_{i \in I} \hat{Y}_{ui} \left[\ln \sigma\left(x_u^T y_i\right) + \sum_{j \in I} \ln\left(1 - \hat{Y}_{uj} \sigma\left(x_u^T y_j - x_u^T y_i\right)\right)\right] - \lambda_\Theta \|\Theta\|^2 \tag{2.14}$$

Researchers such as [Adomavicius 2005] [Burke 2002] have noted the limitations of these kind of systems in their predictive accuracy, in particular we remark the

following problems:(1) *Sparsity*: In some systems data items are rated rarely, making it difficult to find similarities between users or between items. For model based systems there is also a problem since (2)*New item problem*: When a new item is added to the system is difficult to predict its relevance because the user and item profile has little or no information, this problem is critical in systems where items appear and disappear frequently. (3)*Scalability*: As the number of items and users increase, the neighborhood formation process is more demanding computationally, also as vector representations increase their dimensionality, distance metrics to detect similarities become less significant. On model based systems, the computational cost of learning the model parameters as the number of users and items increase is not negligible (4)*Complexity and explainability*: Although model based CF gives better predictive performance than memory based CF, most of the times developer prefer using memory based CF for two reasons: Model based CF is more complex; adjusting parameters can be and time consuming and difficult to maintain. On the other hand it's difficult to find out what the latent dimensions in the model are representing and it's difficult to explain to the user how the relevance of an item has been calculated.

2.1.3 Hybrid Systems (HS)

Generally speaking, CF systems have better predictive accuracy when compared to CB systems [Pilászy 2009]. However in some scenarios *the new item problem* can be critical, for example in online advertisement the underlying data item set is very dynamic and items appear or disappear frequently (In [Guha 2009] is estimated that between 30% and 40% of available ads in an ad network change from hour to hour). Since collaborative information is not always available, a hybridization between systems is desirable these cases.

Burke [Burke 2002] identifies six ways in which different techniques could be integrated:

- **Weight:** Two or more recommender systems operate in parallel, the relevance score given by each one is weighted into a final relevance score.
- **Switching:** If the confidence of a recommender system when calculating a relevance prediction is not high, the system can switch to another recommender system with higher confidence in its output.
- **Mixed:** This paradigm operates when presenting the user a list of relevant items, instead of showing her a list originated by one recommender system, the output of two or more recommender systems is mixed into a single list.
- **Feature combination:** Create new features with information from collaborative filtering systems, for example: Create a new set of features for CB filtering describing items with a set of user ids that have liked the item.

- **Cascade:** A recommender system is used to obtain a coarse-grained list of possible items to recommend, then another system takes this list as an input and does a refinement of the list.
- **Feature augmentation:** Augment the data model of one recommender system with information used by another.
- **Meta-level:** A model of one recommender system is used as an input for another, for example: A CB filtering system where collaborative agents select a pool of possible items to be recommended based on aggregated profiles of similar users [Balabanović 1997].

Hybridization is still an open problem in the recommender system community. For weighted strategies, if correlation between the relevance prediction given by each single recommender used is high, there is no use in aggregating their results on a static basis. On the other hand a dynamic weighting switching strategy is proposed where weights change according to the *clarity* of the user profile [Bellogín Kouki 2012]. This findings indicate that for enabling a good hybridization strategy is better to use recommender with heterogeneous paradigms. For example in the Google news personalization system [Das 2007] three different systems are used: The first one builds a memory based item to item model based on the co-visitation of news items, the other two models are model based: A minhash algorithm that builds clusters of users based on the overlap of common viewed items and a probabilistic latent semantic indexing algorithm that learns a latent variable linking the behavior between users and items. The final calculation of relevance is a weighted response of the scores of the three models.

Other hybridization paradigms, as observed by [Burke 2002], use mixed content based and collaborative strategies in order to avoid the problems registered on pure content based and collaborative based strategies. For example the meta-level algorithm *collaboration via content* [Pazzani 1999] makes recommendations using the similarity between CB profile of users in order to run a CF memory based approach.

Model based CF using CB features has shown good results as well by using the metadata information to adjust parameters of the optimization criterion. For example in [Pilászy 2009] proposes to use a metadata transformation of item vectors instead of learning the item latent factors $Y_{n\times k}$. In this system a matrix $C_{c\times n}$ where each row C_i has the CB item profile and a matrix $W_{c\times k}$ of factors that transform the metadata information into the latent features of a traditional matrix factorization system, in this way the matrix is approximated as $V \approxeq X(CW)^T$. The advantage of this work is that for a new item, it suffices to multiply its $ContentBasedProfile(I_i)$ with the matrix W in order to calculate a relevance estimation. A similar approach is exposed in [Gunawardana 2009] where a Restricted Boltzmann Machine learns item pairwise factors tied to factors that depend on the content based profiles of the items.

In [Gantner 2010] a CF matrix factorization model is applied obtaining the latent factor matrices ($X_{m\times k}$) and ($Y_{n\times k}$) optimized on the Bayesian personalized

ranking [Rendle 2009] (Equation 2.13). Then another regression is used to learn a mapping $\phi\left(ContentBasedProfile(I_i)\right) \rightarrow \mathbb{R}^{1\times k}$ between CB profiles and the latent factor matrix in order to infer latent factors from their CB specification using the same Bayesian personalized ranking criterion. The mapping can be used to calculate the relevance of an item with only CB information.

Despite the hybridization strategy, choosing the right hybridization depends heavily on the available information and the knowledge domain in which information is used, for example, [Pilászy 2009] observes that for movie recommendation even 10 new ratings for a movie are more useful than using their proposed hybrid strategy for rating prediction, still they recognize that this effect could change on other domains where CB systems perform better, for example on text based items such as news recommendation systems.

2.2 Evaluating Recommender systems: Predictive accuracy and scalability

The suitability of a recommender system can be evaluated by many properties, for example: how well they predict the relevance of an item for a user (*predictive accuracy*), for how many users or items the system is able to make a prediction (*coverage*), how much an user can *trust* a recommender system, and how much the recommendation system trusts that the recommendations it makes are relevant (*confidence*). Among many other factors (A survey of evaluation metrics is presented in [Shani 2011]), the predictive accuracy of recommendation systems is the most important measure taken for evaluating the suitability of recommendation systems and is used to validate most of the works.

In order to evaluate the predictive accuracy of a recommendation system, two kinds of experiments exist: online and offline. Online evaluation of recommender systems is done by diverging a small random part the recommender system requests to one or many different systems and then compare their performance to a specific metric [Kohavi 2009]. On the other hand, offline experiments calculate the prediction accuracy metrics using an historic account of user-item interaction, where one part of the historic log is used for training the predictive model (the V matrix) and the rest of the log is used to measure the predictive accuracy of the model (A matrix T with the test ratings information).

Other desirable quality of recommender systems is for them to have a high *Scalability*. Scalability can be understood as the ability of the system to process an increasing amount of work with respect to a desirable performance metric, for example the predictive accuracy of the system. The predictive accuracy of a recommender system can be observed as a quality of the system that depends on factors that can increase easily such as the number of users and items in the system, the amount of information available about users and items, the rate of arrival of new items, the rate of arrival of information about the interaction between users and items and the rate of recommendation requests made to the recommender. It

also depends on other fixed factors such as the setup of the type of recommender, particularly the choice of learning and prediction algorithm. In order to claim that a recommender system is scalable, the relationship between the desired qualities of the recommender system and the factors that affect it must be understood, in this section we will elaborate on both of these evaluation criteria.

2.2.1 Predictive Accuracy Measures

Following the presentation made in [Shani 2011], predictive accuracy measures can be classified under three broad categories: regression based, classification based and ranking based metrics.

Regression metrics measure the predictive accuracy of the system by comparing the difference between the rating a user gave to an item (T_{ui}) and the predicted rating ($\hat{r}_{ui}$). Regression metrics measure how well the recommendation system can guess the rating a user would have given to an item. One simple metric to evaluate the difference between predictions and true values across the test set is the *Mean Absolute Error* (MAE):

$$\mathsf{MAE} := \sqrt{\frac{1}{|T_{ui} \neq null|} \sum_{T_{ui} \neq null} |r_{ui} - T_{ui}|} \tag{2.15}$$

The *Root Mean Squared Error* (RMSE) is historically the preferred metric for evaluating the predictive accuracy of recommender systems [Shani 2011], this metric is more severe than the MAE because it penalizes heavily large differences and is defined as follows:

$$\mathsf{RMSE} := \sqrt{\frac{1}{|T_{ui} \neq null|} \sum_{T_{ui} \neq null} (\hat{r}_{ui} - T_{ui})^2} \tag{2.16}$$

Classification metrics are based on how the users react to items shown to them by the recommendation system. A recommendation system can be seen as an item classifier selecting relevant items for a user from the set of possible items I. Classification metrics measure how well the classification adapts to the choices of a user. For testing classification metrics using the matrix T, it is considered that the items that the user has selected ($T_{ui} \neq null$) are relevant for her.

To calculate classification metrics, a confusion matrix is used Table 2.1. Once a test is done, its results are divided into 4 sets: *True positives* (TP) is the set of items that has been correctly classified as relevant, *false positive* (FP) is the set of items that where not classified as relevant but where relevant for the user, *false negative* (FN) is the set of items that where selected but where not relevant and *true negative* (TN) is the set of items that where correctly classified as irrelevant. It is important to mention that when considering only the elements the user has selected as relevant the number of false positives is overestimated.

Categories/Selection	Selected	Not Selected
Relevant	True positive (TP)	False Positive (FP)
Irrelevant	False Negative (FN)	True negative (TN)

Table 2.1: Confusion matrix for categorization

Based on the confusion matrix, *precision* is defined as the fraction of items that were relevant from the set of selected items:

$$\mathsf{PRECISION} := \frac{TP}{TP + FP} \tag{2.17}$$

Recall is defined as the fraction of successfully retrieved items from the total of relevant items.

$$\mathsf{RECALL} := \frac{TP}{TP + FN} \tag{2.18}$$

Precision and recall are related since they both depend on the number of successfully classified items, a system that tries to improve its recall by increasing the number of selected items will cause a decrease in its precision. The most common metric used to relate both metrics is the *F-measure*, calculated as:

$$\mathsf{F\text{-}MEASURE} := \frac{2 \times \mathsf{PRECISION} \times \mathsf{RECALL}}{\mathsf{PRECISION} + \mathsf{RECALL}} \tag{2.19}$$

Most of the times, recommender systems present an item list of limited length to the user, in these cases it is useful to calculate the precision limited to a certain amount of results, this measure is called *Precision at N*. Another important measure the predictive performance on a result list is to plot the true positive rate vs the false positive rate (*ROC curves*), the area under the plotted curve is called the *area under the curve* (AUC).

Finally, rank based metrics evaluate how well the recommender system orders a list of recommendations based on the user preferences. These evaluation metrics are better suited to evaluate systems that offer a limited list of recommendations since they penalize a recommender system that places non-relevant items on the first positions of the recommendation list. Let J_u be a list of ordered recommendations offered for user u,$\hat{Y}_{ui}$ is a binary function that tells if item i is relevant for user u and $\hat{R}_{ui}$ is the rank of item i for a user u the *Discounted Cumulative Gain* (DCG) of the list is:

$$\mathsf{DCG} := \sum_{i \in J_u} \frac{\hat{Y}_{ui}}{\max\left(1, \log_b \hat{R}_{ui}\right)} \tag{2.20}$$

The *Normalized Discounted Cumulative Gain* (NDCG) is expressed as the DGC divided by the maximum DGC possible.

Other used metric for evaluating ranked lists is the *Reciprocal Rank* (RR), which only cares about the position in the list of the first relevant result. It is defined as the inverse of the rank of the first relevant result:

$$\mathsf{RR} := \sum_{i \in J_u} \frac{\hat{Y}_{ui}}{\hat{R}_{ui}} \prod_{j \in J_u} \left(1 - \hat{Y}_{ui} \mathbb{I}\left(\hat{R}_{uj} < \hat{R}_{ui}\right)\right) \tag{2.21}$$

The work presented in this thesis will evaluate its predictive accuracy using the RMSE metric. The choice of favoring this metric among others is twofold: (1) The proposed model adjusts its parameters using a regression that penalizes the square loss between the predicted and known value, therefore the most suitable way to evaluate its predictive accuracy is to use a regression metric such as the RMSE, and (2) it is a popular metric used by the recommender system's research community, therefore by providing the results of the proposed predictive model in terms of the RMSE can provide easily to researchers an idea of the predictive performance of the proposed system.

In order to provide an intuition on what a value of RMSE means, different algorithms presented in this chapter were trained and tested using the Movielens-1M dataset from the GroupLens research group[1] and the obtained RSME results on a test set are present in Table 2.2. Although the presented methods are not trained using optimal values, the results gives us valuable insights on how the RMSE metric works. Opinions from users are nosiy, incomplete and changing; an ideal value of RMSE should be 0 but state of the art methods only attain an RMSE of around 0.886 on this dataset. On the other side of the spectrum an algorithm that guesses randomly a rating from the available ones obtains an high RMSE of 2.059. A middle ground is a non-personalized method such as predicting the item average that has an RMSE of 0.9878, this is compatible with the intuition that predicting popular items is a good enough strategy, but personalized methods bring better predictive performance to the users.

Algorithm	Parameters	RMSE TEST
Random		2.0593
Item Average		0.9878
User Based KNN	$k = 50$ with Pearson Correlation	1.103
Item Based KNN	$k = n$ with Pearson Correlation	0.9948
Unbiased Factorization	$k = 10, \gamma = 0.01, \lambda = 0.001, iter = 300$	0.9163
Biased Factorization	$k = 10, \gamma = 0.01, \lambda = 0.001, iter = 300$	0.8887
SVD++	$k = 20, \gamma = 0.001, \lambda = 0.005, iter = 212$	0.8868

Table 2.2: RMSE accross different models on test set using Movielens-1M dataset

[1] http://www.grouplens.org

	Scaling factors	Non-scaling factors
Domain factors	Number of users Number of items User-item interaction information User-item metadata Recommendation requests	Algorithm
Architecture factors	Computational power Storage	

Table 2.3: Scalability factors for traditional recommender systems

2.2.2 Scalability

As defined in [Duboc 2007],

> *Scalability* is a quality of software systems characterized by the causal impact that scaling aspects of the system's environment and design have on certain measured system's qualities as these aspects are varied over expected operational ranges. If the system can accommodate this variation in a way that is acceptable to the stakeholder, then it is a scalable system.

Taking the premise that the main desirable quality of a recommender systems is to attain a high predictive accuracy while being able to respond to user requests for predictions, the scalability analysis should explain how these qualities are governed by the domain and infrastructure characteristics. *Scaling aspects* are the domain or infrastructure characteristics that increase easily in the system, on the other hand *non-scaling aspects* are domain or infrastructure characteristics that are fixed or change in a nominal scale. In Table 2.3 a classification of the scalability factors present in recommender systems is presented.

Information from users and items is considered a scaling factor due to the availability of information sources. On the other hand recommender systems gather and process user information on a centralized computational entity, under this configuration, computational power and storage are considered an utility that can scale up as needed. A scalability analysis of recommender system must vary different recommendation algorithms and consider how both of the dependent variables react to the scaling factors.

Broadly speaking, two phases are present in a recommendation algorithm (1) training and (2) prediction. While the training task can be related to the predictive accuracy of the system, the prediction task can be related to the recommendation response rate. Memory based systems make a tradeoff between having little training but having a higher prediction computational complexity, on the other hand model based systems go through an expensive training phase but their prediction computational complexity is lower. In terms of the prediction accuracy, memory based systems have no formal objective behind them, thus leaving them with lower

accuracy measures when compared to model based ones. As seen in section 2.1, each algorithm's predictive accuracy depends on their optimization criterion since they try to optimize a different predictive accuracy measures depending on the task the recommender engages. Despite some correlations between regression and classification metrics, there is no evident relationship between the regression and classification accuracy metrics [Bellogin 2011].

For model based recommender systems, the training task of model based systems is governed by the scalability of the training algorithm. Statistical learning theory [Vapnik 1999] can be used to explain how the convergence of the training process behaves in terms of the scalability factors involved in the recommendation process.

The objective of a learning task is to find the parameters Θ for a prediction function $f_\Theta(u,i)$ whose predictions generalize well on future examples. In order to adjust the parameters, learning algorithms use a loss function $\ell(f_\Theta(u,i),r)$ that scores the prediction against the true value of the user's choices r. The learning task consist in finding the adequate parameters that minimize the *expected risk function*:

$$e(\Theta) = \int \ell(f_\Theta(u,i),r)\,\mathrm{d}P(r) \tag{2.22}$$

Since the distribution of the true user's choices is unknown ($P(r)$), an approximation of the expectation of the error is calculated as the average loss across the known information about users, known as the *empirical risk function*:

$$\hat{e}(\Theta) = \frac{1}{|V_{ui} \neq null|} \sum_{V_{ui} \neq null} \ell(f_\Theta(u,i),V_{ui}) \tag{2.23}$$

Intuitively, a model having a small empirical risk over the known information about users should have a good predictive accuracy measured by the criterion it minimizes. However, it is likely that the empirical risk is higher than the true expected risk. Statistical learning theory establishes that the difference between the the unknown expected risk and the measured empirical risk is bounded by a function of the number of examples used to train the model (user-item interaction information) and the complexity of the loss function used to train the model. The complexity is measured using the *Vapnik-Chervonenkis (VC) dimension* [Vapnik 1999]. For a non-negative loss function bounded between $0 \leq \ell(f_\Theta(u,i),r) \leq B$, and letting h be the VC dimension of the loss function, the difference between the empirical risk and the expected risk is bounded by the following inequality with probability at least $1-\delta$:

$$e(\Theta) \leq \hat{e}(\Theta) + \frac{B\varepsilon}{2}\left(1+\sqrt{1+\frac{4\hat{e}(\Theta)}{B\varepsilon}}\right) \tag{2.24}$$

Where ε is defined as:

$$\varepsilon = 4\frac{h\left(\ln\frac{2|V_{ui}\neq null|}{h}+1\right)-\ln h}{|V_{ui}\neq null|} \tag{2.25}$$

Asymptotically [Schapire 2012, p. 37], with probability at least $1-\delta$ the inequality can be aymptotically expressed as:

$$e(\Theta) \leq \hat{e}(\Theta) + O\left(\sqrt{\frac{h\ln(h/|V_{ui} \neq null|) + ln(1/\delta)}{|V_{ui} \neq null|}}\right) \tag{2.26}$$

This bound gives a tradeoff in the scalability factors considered for model based recommender systems: the user-item information factor that is considered as an scaling factor in recommender system and the VC dimension of the model that is a non-scaling factor since it depends on the chosen recommendation algorithm criteria. The error decreases as the ammount of information increases, but a complex minimization criterion makes it harder to learn a low-error model.

One of the most popular algorithms for training the predictive model is the *Stochastic Gradient Descent Method* (SGD) [Bottou 2010]. Taking the rating matrix V, model parameters Θ, a function that uses the learned parameters to predict relevance $f_\Theta(u,i)$, a convex differentiable loss function $\ell(f_\Theta(u,i), V_{ui}, \lambda)$, a learning rate γ, and regularization parameter λ, the algorithm is presented as follows:

Algorithm 1: Stochastic gradient descent algorithm

Data: V, Θ, γ,$f_\Theta(u,i)$,$\ell(f_\Theta(u,i), V_{ui}, \lambda)$
Result: Θ
initialize $\Theta := \mathcal{N}(0,1)$;
repeat
  draw random V_{ui} from V;
  foreach $\theta \in \Theta$ **do**
    $\theta \leftarrow \theta - \gamma \frac{\partial}{\partial\theta}\ell(f_\Theta(u,i), V_{ui}, \lambda)$;
  end
until *convergence*;
return Θ

The computational complexity of the SGD algorithm in the training phase is dominated by how many iterations are needed until a low generalization error is achieved. This error $\mathcal{E}$ can be seen as the sum of three errors: The expected error of using a class of hypothesis that approximates the optimal solution ($\mathcal{E}_{app}$), the estimation error of minimizing the empirical risk instead of the expected risk ($\mathcal{E}_{est}$) and the optimization error linked to the time the optimization algorithm has been running ($\mathcal{E}_{opt}$) [Bottou 2008] [Bottou 2010]. Any learning task that wants to reduce the expected error presents a tradeoff between a tolerance error factor ρ and the number of training examples that the algorithm must see to attain a low error in terms of the excess error $\mathcal{E}$ of the predictive model. Taking the previous bound in Equation 2.26 and assuming a finite hypothesis size h, the asymptotic behaviour of the error behaves as:

$$\mathcal{E} = \mathcal{E}_{app} + \mathcal{E}_{est} + \mathcal{E}_{opt} = \mathcal{E}_{app} + O\left(\sqrt{\frac{\ln\left(/|V_{ui} \neq null|\right)}{|V_{ui} \neq null|}} + \rho\right) \tag{2.27}$$

Assuming that the loss function used has strong convexity properties, the author shows that the square root term in Equation 2.27 can be replaced by an exponent $\alpha \in [1/2, 1]$. Keeping into account that the three components of the error should decrease at the same rate, the computational complexity in time of attaining the lower bound of the excess error obeys the following asymptotic equivalences:

$$\mathcal{E} \sim \mathcal{E}_{app} \sim \mathcal{E}_{est} \sim \mathcal{E}_{opt} \sim \left(\frac{\ln\left(/|V_{ui} \neq null|\right)}{|V_{ui} \neq null|}\right)^{\alpha} \sim \rho \tag{2.28}$$

This means that SGC algorithm is a good optimization strategy for the recommendation scenario because it adapts to the scaling factors. The computational complexity of training a user-item example is O(1) and the expected time needed for it to reach a determined error $\mathcal{E}$ is not determined by how many training examples the model has seen but is inversely proportional to the expected error $O(1/\mathcal{E})$ in opposition to other optimization algorithms such as the normal gradient descent and second order gradient descent. These algorithms incur in a more complex update rule O(*training examples*) and the time to reach a determined expected error is $O(\frac{1}{\mathcal{E}^{1/\alpha}} \ln \frac{1}{\mathcal{E}})$.

Another factor to take into account in the training phase is how new information is included into the model. So far the presented models in this chapter train the prediction model in a bach process where the matrix of user-item information is static. In order to include new user-item information, some systems can manage to re-calculate the whole model every once in a while with the updated user-item information. However, in systems where information arrives at a fast rate, they leave a big part of the user-item interaction data outside the prediction model between the batch processes. For example Amazon on its selling peak sold 426 items per second[1], clearly a different strategy is needed in order to keep up to date the knowledge model.

For neighborhood models updating the user-user or item-item similarities each time a new user-item interaction arrives to the system is not feasible since it has a quadratic complexity along users or items. For user-based systems (Equation 2.4), [Papagelis 2005] proposes to re-balance user similarities based on the Pearson similarity (Equation 2.5) as each new example arrives in the system. A shortcoming of this approach is that for each user-item interaction the system has to check how other users have interacted with the current item in the new information. Authors show that there is no loss of predictive accuracy by using this method.

For model based systems, an approximation of new user and item profile from an already trained model can be done for systems based on the SVD decomposition

[1] http://article.wn.com/view/2013/12/26/Amazon_says_it_sold_426_items_a_second_on_Cyber_Monday/[Accesed May 2014]

of the user-item matrix (Equation 2.7). The *folding-in* technique is presented by [Sarwar 2002] takes advantage of the basis represented by matrix $\Sigma_{k\times k}$ to project a user profile $V_u(1 \times n)$ into a new row of matrix $U_(m \times k)$ as follows: $U'_u = V_{u1\times n} V^*_{k\times n} \Sigma_{k\times k}$. Since this approximation is not exact as the basis for users and items are not kept up to date, eventually the whole recalculation process has to be applied on the whole user-item representation.

Finally, in [Luo 2012] an extension of the biased model (Equation 2.8) is presented to incrementally adjust the weights of the average, user an item biases and the latent factors of users in an incremental fashion as new user-item information arrives to the model. The training rule for updating the parameters is modified in order to remove the sequence dependency that the update rule of the gradient descent method imposes on the update rules. The update rule is expressed as an increment over the initial value of the parameter at each iteration, and at the end of the iteration the state of the parameters is updated and saved for future use. When a new user-item information arrives, a reconstruction of the past iterations for the user and item latent vector is made based on the saved values along the batch training process. A shortcoming of this approach is that it needs an space complexity t times bigger than the non-incremental approach where t is the number of iterations needed for the batch-trained model to converge. For the biased model the space complexity is of $O(t * n * k + t * m * k)$ where n is the number of users, m is the number of items and k the size of the latent factors.

2.3 Conclusions

As seen in Table 2.4, computational complexity of neighborhood systems training task is fixed to the number of users (m) and items (n) in the system when a precomputation of similarities between users or items in the system is calculated. This precalculation process has the cuadratic complexity of comparing every user or item of the system with each other, times the complexity of the similarity metric used. Similarity metrics such as the Pearson correlation (Equation 2.5) are linear on the maximum number of shared ratings between users (p) or between items (q). Assuming that the similarities between items or users are stored in an ordered fashion, the complexity of the prediction phase in memory based systems is dominated by the process of neighborhood formation (selecting from the complete profile information which which users or items have that rating in common) that has a complexity of O(n) or O(m) and the calculation of the weighted average on the N selected users or items has a complexity of O(N).

The complexity of model based systems training phase using the SGD algorithm is dominated mainly by the number of iterations needed to achieve a low expected error of the model (t). As seen in the previous section, the number of iterations doesn't depend on scaling factors such as the ammount of user-item data, therefore when taking into account the scaling factors of the recommender system, this algorithm scales well. Other important factor in the complexity of the training

	training	prediction
User based (Eq. 2.4)	$O(m^2p)$	$O(m)$+$O(N)$
Item based (Eq. 2.6)	$O(n^2q)$	$O(n)$+$O(N)$
Bias model (Eq. 2.8)	$O(k * t)$	$O(k)$
SVD++ (Eq. 2.10)	$O((k + m) * t)$	$O(k + m)$
ORDRec (Eq. 2.12)	$O(\|\mathcal{O}\| * t) + base$	$O(\|\mathcal{O}\|)+ base$
BPR (Eq. 2.13)	$O(2k * t)$	$O(2k * m)$
CLIMF (Eq. 2.14)	$O((\tilde{n}2m + m) * k * t)$	$O((\tilde{n}2m + m) * k)$

Table 2.4: Time complexity of CF algorithms for prediction of relevance on all available items

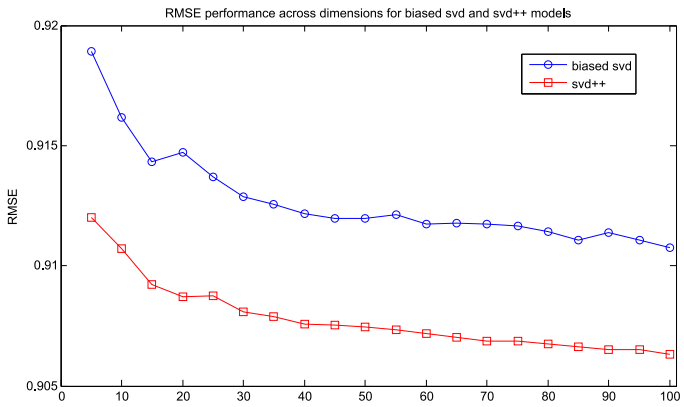


Figure 2.4: Predictive performance of svd models accross dimensions

is the calculation of the gradient of the loss function used for the optimization procedure: For the biased model the calculation of the gradient of the optimization criterion with respect to each of the parameters is $O(1)$, therefore for an iteration its complexity is $O(k)$. The SVD++ criterion has an added complexity of adding the extra vectors z_j of the items the user has interacted to the user profile, thus the complexity of going through this information is $O(m)$.

In order to illustrate the tradeoff between these factors, the biased SVD (Equation 2.8) and the SVD++ (Equation 2.10) models are compared using the Movielens-100k dataset from GroupLens research group[1] by measuring the generalization error of the model with the RMSE predictive accuracy metric (Equation 2.16). The SVD++ model has a bigger hyphotesis class than the SVD model since it has to adjust the extra parameters to account for the implicit information, therefore it has better predictive performance than a biased model when completely trained (Figure 2.4). However, as seen in the bounds of the iteration steps needed to achieve a low generalization error (Equation 2.27) it takes more iterations to adjust its parameters to a low generalization error (Figure 2.5).

[1] http://www.grouplens.org

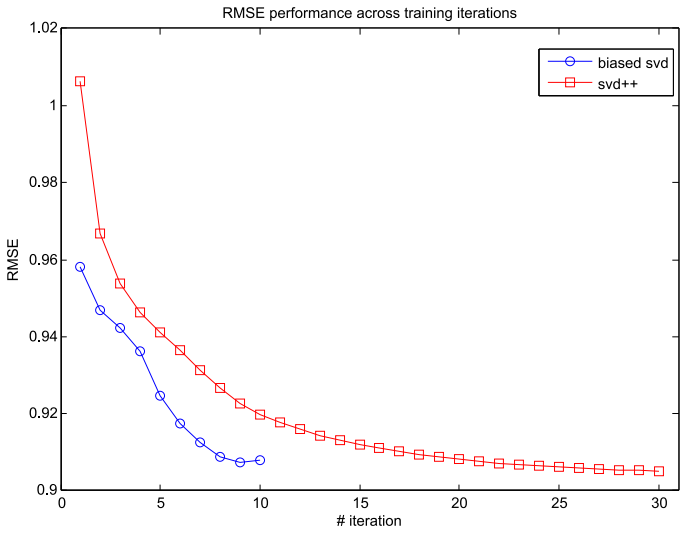


Figure 2.5: Predictive performance of svd models accross iterations

Dataset	Users	Items	Sparsity	Train	Iters	RMSE Test
Movielens-100k	943	1680	6,31%	15 (2.20)	300	0,92952
Movielens-1M	6040	3698	4,47%	152,71 (18.20)	142	0,88598
Movielens-10M	69878	10676	1,34%	467,8 (64.84)	28	0,87028

Table 2.5: Running time of the SVD++ algorithm accross different datasets

Model	Complexity of update rule
User based (Eq. 2.4)	$O(mp)$
SVD model (Eq. 2.7)	$O(k^3)$
Bias model (Eq. 2.8)	$O((\lvert R_u\rvert + \lvert R_i\rvert) * t * k)$

Table 2.6: Time complexity of single update rule in incremental CF

Also for illustrative purposes, the running time of training for different sized datasets is shown in Table 2.5. The SVD++ algorithm was trained with $k = 10$, $\gamma = 0.001$ and $\lambda = 0.005$ for differently sized versions of the Movielens dataset (100k, 1M and 10M ratings) for maximum 300 iterations or until convergence. The SVD++ implementation was taken from the open source Mahout implementation [2] that ran on a machine with 16GB of RAM and an Intel Core i7 with 8 processors. The times reported in the table are in minutes, and the time in parenthesis is the time that the system spend calculating the error on the dataset while training. As expected as the number of users an items present in the system, the running time increases proportionally to the number of users(m) in the system.

As the ranking task is more difficult than rating prediction, the BPR and CLIMF models have more complex models than the rating prediction ones. The BPR model uses a function of the parameters that predicts the preference of item i over item j for a user u. Letting this function to be described as a matrix factorization problem (e.g $\hat{X}_{uij}(\Theta) := x_u^T y_i - x_u^T y_j$) the gradient calculation complexity of this function is twice the one of the base model. On the other hand the complexity of the CLIMF gradient calculation is dominated by the ammount of observed positive interactions $\hat{Y}_{ui} = 1$ between users and items, letting $\tilde{n}$ be the average number of positive interactions for each user, the complexity of calculating the gradient with respect to each parameter of the model is $\mathrm{O}(\tilde{n}2m + m)$.

Finally, the complexity of updating the data model for a single new example r_{ui} without reconstructing the whole model is presented in Table 2.6. The user based model of [Papagelis 2005] has to re-balance for all users the pre-computed similarity if they share a common rating, this has a complexity of $O(mp)$ where p is the size of the rated elements of the active user, usually $p \ll n$. For the SVD model [Sarwar 2002] the multiplication of the folding-in strategy has a complexity of $O(k^3)$. Finally for the update strategy of the biased model [Luo 2012] the system has to iterate over the known ratings of the user and recreate for each iteration of the batch process how the user latent feature would have changed, including how the item latent factor would have changed. The complexity of iterating t times over the known ratings $R(u)$ and updating each coordinate of the latent factors is $O(|R(u)| * t * k)$. Since the same process is done for the active item iterating over the known ratings of the item ($R(i)$), the final complexity of a single update is $O((|R(u)| + |R(i)|)*t*k)$.

[2] https://mahout.apache.org/

Synopsis and first design choices

In this chapter the recommendation problem was explained. A classification of approaches was presented and two factors for evaluating recommendation systems were explained: Predictive accuracy and scalability. Finally, an explanation on how the tradeoff between the scalability of the system and its predictive performace was presented. In the light of this explanations, some design considerations can already be made about the proposed model of this thesis.

In this thesis, an hybrid predictive system will be developed by *weighting* the predictions made by a content based and a collaborative filtering system. Rather than using a memory based model, both systems will use a model based learning strategy. Despite their higher training complexity, the advantage of model based systems is twofold: By having a formal objective of minimizing an expected predictive accuracy measure, they perform better than memory based ones when compared on these measures[a], moreover, as user-item information scales the predictive performance of the model increases as well. Finally, comparing the complexity of the model on the prediction phase, collaborative filtering model based systems have a better scalability since the dimensionality of the models k is usually much less than the ammount of neighbors N for the rating prediction task. This is convenient in systems where the requests for predictions scale. A clear disadvantage of model based CF is the complexity of their update process when new user-item information arrives, this concern will also be adressed in Chapter 4.

[a] Validation of offline training and adjustment should be validated also on real life settings, understanding the relationship between offline and online measurement of the predictive accuracy of recommender systems is still an open issue (see [Amatriain 2013]).

In the next chapter, privacy will be explained as a factor for evaluating recommender systems. While the predictive performance and scalability of a system are usually in mind when designing recommender systems, the privacy of users is not. Considering privacy among predictive performance and scalability will introduce new requirements to the model based learning system that will be proposed and will be discused further in Chapter 4.

Chapter 3

Privacy: a factor for evaluating recommender systems

Contents

Privacy is a factor that is rarely considered in the design of recommendation systems. In this chapter, it will be shown that privacy-enabled personalization systems impose restrictions on the architecture of the system, the amount of information gathered about the user and the algorithms available for training the prediction model and calculating relevance predictions in order to bring some level of privacy to their users. Privacy impacts two desirable qualities of recommender systems: their predictive accuracy (Section 2.2.1) and their scalability (Section 2.2.2). To present how these three factors are related, this section will be focused on the restrictions that privacy-enabled system impose on the predictive performance of recommender systems. First, the risks to user privacy present in recommender systems will be introduced (Section 3.1), second, a classification of techniques used to address this concerns (Section 3.2), and third, a review of the works made so far based on the classification (Section 3.4).

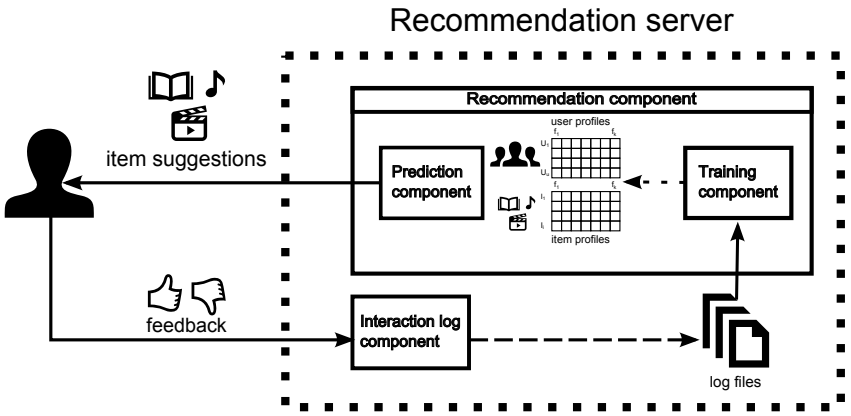


Figure 3.1: Traditional recommender systems

3.1 Privacy and recommendation

As seen in section 2.1, recommender systems have been used successfully in numerous scenarios ranging from movies [Netflix 2013], electronic commerce [Linden 2003], and more dynamic environments such as news [Das 2007] and videos [Davidson 2010], where hybrid approaches have shown to be least vulnerable to the perennial problems identified in literature.

Despite their success, all these approaches have one thing in common: Recommender systems gather information about users and store it in a centralized entity, then they apply heuristics or data mining techniques to learn the users' interests with the purpose of detecting which elements are relevant for the user. Figure 3.1 gives an overview of the components of a traditional recommendation system: The *interaction log component* is in charge of gathering feedback information about the interaction between the users and items in the system; The *recommendation component* is in charge of two tasks delegated onto two components: (1) The *training component* learns the parameters of a predictive model by going through the user-item historic interaction kept in the system logs, and (2) the *prediction component* that actively searches among the database of available items the most relevant ones for the user based on the learned parameters (user and item profiles). Both user-item feedback information and the profiles of users and items in the system are kept under the direct administration of the organization.

Users trust that the information submitted or registered about them will be used for filtering purposes, however their information can be used for purposes different than filtering, which is considered by [Lam 2006] as an *exposure* risk. According to [Foner 1999], keeping user profile information on a centralized entity can lead to *exposure* risks configured in five ways:

- **Deception by the recipient (misleading service):** The system can lie about its privacy policies and trick users to reveal personal information, using it later for a different purpose from profiling for advertisement display. For example selling the information or sharing it with other organizations.

- **Mission creep:** Initially the policy of usage of personal information is defined clearly by the system, but later the systems expands its goals in a previously unforeseen manner, changing the use of personal information for other purposes related to the new goals of the organization.
- **Accidental disclosure:** Information about users can be made available accidentally, for example leaving private information on a server that can be accessed by a search engine over the Internet.
- **Disclosure by malicious intent:** Storage servers' security can be breached and users' personal information can be stolen.
- **Forced disclosure:** Systems must disclose the information for legal reasons.

For traditional recommender systems, the logs the recommendation system keeps reveal the tastes of the user by the choices the user has made (which items the user has seen and how she has rated them). At first sight, it can be argued that if this information is usually revealed by users, trend supported by social network sites, however in these cases the user is in control of what information is being revealed. The aim of privacy-enabled recommendation systems is to give users tools to protect their privacy and keep the choice to themselves if they want to reveal their information.

Privacy allows users to decouple themselves from their actions. On an electronic commerce website, users trust that their opinions are used by the organization for helping them, however the exposure of their opinions on items can bring various potential harms [King 2010]. For example (1) Users can be targets of unwanted commercial solicitations (spam); (2) Users can be victims of identity theft and fraud; and (3) Exposure of personal information increases the user's risk to be subject of unfair commercial practices.

In some cases users want to keep their actions or opinions private since they can be contrary to their reputation. Opinions on items might reveal political inclination, sexual orientation, physical or mental treatments the user is taking or religious inclination of a particular user. Rather than being a nuisance, the breach of this information can be prejudicial to users and even bring them physical trouble.

3.2 Designing privacy-enabled recommender systems

A *privacy-enabled recommender system* is a recommender system that offers changes in its architecture and algorithmic methods in order to protect the privacy of users. Privacy engineering [Spiekermann 2009] provides a guideline on the factors that must be considered when building a privacy-enabled systems. Two approaches are considered (1) privacy by policy and (2) privacy by architecture. The former approach manages user privacy by adhering to fair information practices defined by regulatory entities. The latter enforces a change in the architecture of traditional system in order to address the privacy concerns of the stakeholders of the system, particularly

introduces changes in the mechanisms in charge of (a) user data collection, (b) user data storage, and (c) user data processing.

In [Toch 2012] and [Jeckmans 2013] a series of strategies for enabling privacy for personalization and recommender systems are described according to their approach. For policy-based strategies, the following tools are considered:

- **Privacy through law and regulation:** Government and governing bodies regulate the operation of electronic systems that process personal data. This includes personalization systems such as recommender systems. On their regulation they have enacted acts for protecting user privacy, for example the European Commission has proposed a modification on their guidelines to reinforce user data protection for unauthorized usages [European Commission 2012] .

- **Privacy through awareness and user control:** Users are given tools for managing their privacy, enabling them to easily realize the conditions and policies of their information usage. For example the W3C Platform for Privacy Preferences [Cranor 2002] recommendation facilitates information systems to inform their users about the privacy policies implemented on their data use. These practices allow users to define their privacy preferences, enabling them to restrict the use of their information and hide or obfuscate the information registered about them.

Policy-based approaches have the advantage of being easily integrated to existing business models, in terms on their efficiency protecting the users against the exposure risks, some of them can be avoided since the adherence to the policies of the system be verified thanks to technical audit mechanisms. However, this is not sufficient to avoid the considered exposure scenarios: mission creep and disclosure of information by malicious intent or by forced disclosure.

Architectural-based strategies are placed on top of policy-based ones, changing the architecture of the system in order to increase the privacy guarantees of the users. [Kobsa 2007] [Toch 2012] and [Jeckmans 2013] have identified the following architectural-based strategies for enhancing user privacy in personalization and recommendation systems.

- **Pseudonymous personalization:** The simplest strategy that has been adopted for privacy reasons is to allow users to use a pseudonym instead of their real information for accessing personalization services. By using this strategy systems attempt to reduce the exposure risk since the attacker can't be sure about the real identity of the user.

- **Anonymization:** These techniques appeared motivated by the problem of protecting individual privacy when operating data mining techniques on a database of user information. In order to obscure the relationship between user information and their real identity, perturbation techniques are used to remove or obfuscate personal information gathered by the personalization

system without damaging the structure of the underlying data. These techniques give the users a "plausible deniability" that the information found in their profiles is really from them. There are many options for perturbation or obfuscation in literature (see [Bakken 2004]) We divide these techniques on four groups : (1) *Heuristic transformations* (2) *k-anonymity* [Sweeney 2002] (3) *Random perturbation techniques* [Agrawal 2000], and (4) *differential privacy* [Dwork 2008]:

Heuristic transformations modify the information about users based on heuristics designed to keep the structure of the data for the learning algorithm and making it difficult to attackers to discover the original data. For example by substituting some information of users' profile to fixed values or swapping data from similar profiles.

k-anonymity modifies the information of users avoiding to disclose in the data with *quasi-identifiers*. Quasi-identifiers are a combination of attributes that if known can easily be correlated with other sources to obtain the real identity of the user. In order to hide the relationship between the real user and the information from the quasi-identifiers, a guarantee is introduced into the data so at least k-users must share the same information of quasi-identifiers present in the data.

Random perturbation techniques distort the values registered in the profile by adding some noise to them with values from some distribution, making it difficult for an attacker to correlate the values present in a user profile with information from other sources. In [Agrawal 2000] noise from a Uniform and a Gaussian distribution are added to user information. Assuming each feature or dimension of the user profile information as a random variable the authors show that the original distribution can be reconstructed using the perturbed values, thus being able to apply the data mining learning algorithms on the learned information

Differential privacy builds on the work from random perturbation strategies providing formal guarantees to the privacy offered to users. Differential privacy [Dwork 2008] introduces the required amount of noise taken from a laplacian distribution to a function of the private information to guarantee that the presence or absence of a single user in the complete database of users doesn't change significatively the answer of the function, so no attacker that accesses the database can be certain that a particular user participates in the database. Differential privacy parametrizes the amount of noise that should be added to the user profile based on a privacy budget ε and the sensitivity of the function used over the user data Δ_f

- **Client-side personalization:** Motivated by the fact that the exposure risks are a consequence of a centralized entity managing the information about users, client-based personalization systems move the information gathering, processing and storage from a centralized entity to each user device. Generally speaking existing CB filtering techniques have no problem operating with this paradigm [Kobsa 2007] even with elaborate models. However given the limitations of CB filtering, CF or HS are preferred over CB. Recent user studies [Kobsa 2014] have shown that using client-side personalization increases the perceived privacy of users.

- **p2p architectures:** One of the ways in which CF or HS can operate under client-side personalization is to allow the exchange of user profile information between client-side agents. One of the ways this can be achieved is by connecting users to each other by a network across the client devices. The information from the community of users allows the client-side agents to use modified versions of CF or HS algorithms. One of the disadvantages of this approach in terms of exposure risks is that the gathering of user information is replicated over many users again privacy risks into the system. Initially these architectures were introduced motivated by scalability concerns [Tveit 2001], and later mechanisms for keeping privacy were added. For the sake of respecting the privacy of the users when collaborating between agents, user profile information is often protected before being exchanged by : (1) The use of pseudonymous and anonymization techniques and (2) cryptographic tools such as homomorphic encryption.

- **Aggregation on server:** Other way in which CF or HS systems can operate under client side personalization is by introducing an aggregation server that makes available community information that is used by client-side agents to get information about the preferences of other users. The aggregated information is either already public information an aggregation calculated collaboratively between the client-side agents or information that is sanitized and trusted to the aggregation server. As well as in distributed architectures, user profile information can be altered before leaving the user agent by anonymization and cryptographic strategies.

In the next section, the attacks on user privacy on architectural enabled recommender systems are presented.

3.3 Identified attacks on privacy-enabled recommender systems

A *privacy breach* in a recommendation system is configured when an attacker learns something about the user that was previously unknown to her. In the traditional centralized case (Section 3.1) the risks of a privacy breach are high since the whole

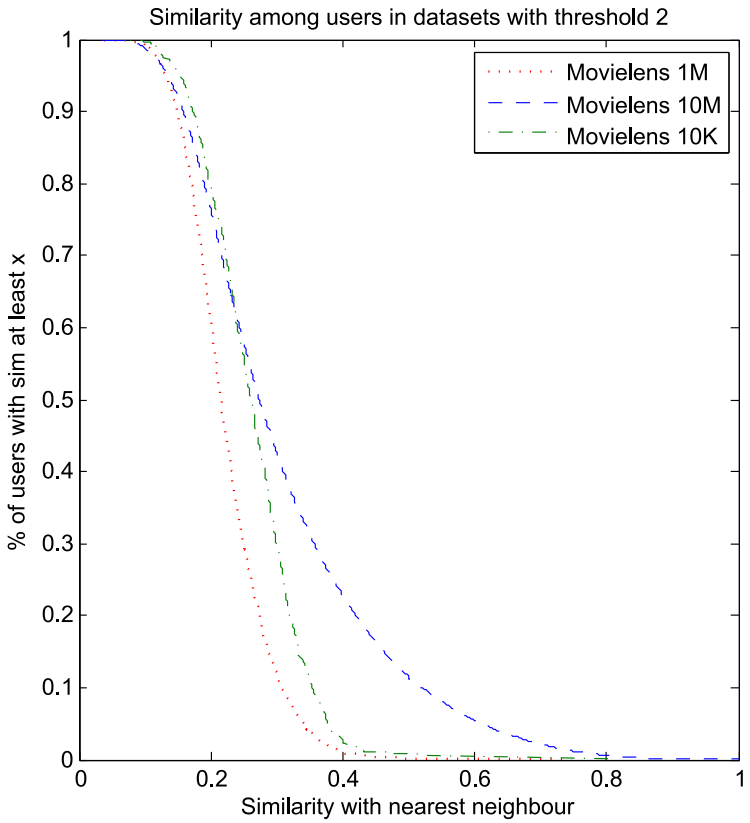


Figure 3.2: Distribution of similarities with nearest neighbor using sim function

log of user-item information can be exposed by means of a configuration of an exposure risk.

Architectural-based strategies to preserve the user's anonymity have their shortcomings as well. The use of pseudonyms and anonymization techniques have been studied as a way of publicly disclosing information about users without exposing *"personal identifiable information"*. In this way if a exposure risk is presented, users can claim "plausible deniability" about the information present in their exposed profiles.

Unfortunately, these strategies fail on *micro-data*, that is the log that registers the user-item interaction. Because data information of users is sparse, users are statistically far from each other as evidenced by [Narayanan 2008]:

Let $supp\,(u)$ be the set of elements user u has rated and $sim\,(u,v)$ a similarity measure between users u and v defined as:

$$sim\,(U_u \times U_v) = \frac{\sum_{i \in I} sim\,(V_{ui}, V_{vi})}{|supp\,(u) \bigcup supp\,(v)|} \tag{3.1}$$

Where $sim\,(V_{ui}, V_{vi})$ is the similarity of opinions of users u and v about item i which is usually an identity function that is one if the opinions are close within a certain threshold.

As seen in Figure 3.2, when calculating the similarity across all users in different datasets using the *sim* function, the distribution of nearest-neighbor similarities shows that the majority of users are far from each other. For a threshold of 2 in the Movielens-100k and Movielens-1M less than 1% of the users have a similarity of 0.5 (Modify with threshold 1 on all datasets). This means that if an attacker knows some *auxiliary information* about the user (a subset of preferences previously known, for example finding out some of their preferences on items from a social network) there is a high probability that by correlating the auxiliary information with the anonymized information with the similarity function, that is calculating $sim\left(U_u, aux\left(U_u\right)\right)$, the attacker can de-anonymize the masked or perturbed record. Narayanan et Al. [Narayanan 2008] show that by using a modified version of the *sim* function, a user can be de-anonymized with little auxiliary information and even incorrect data. Attacks can go even further when in control of the user-item iteractions as well, Bhagat et al. [Bhagat 2014] show an attack were the attacker actively presents items to users and learns their private attributes by taking advantage of rarely rated items as well.

Since there is no algorithm that can guarantee that user will be undetectable with the help of an arbitrary amount of auxiliary information [Dwork 2006] and introducing increasingly higher amounts noise or perturbation into the user profile can damage the utility of the information being calculated from the data, the objective of privacy-enabled recommender systems is to bound the necessary auxiliary information about the user in order to make it hard for an attacker to link their auxiliary information with the observed information about the user. One direct work of trying to reduce the risk of user exposure is not to protect individual privacy but to release a statistical function of the private data that doesn't change much if a user takes part or not in the database. [McSherry 2009] publishes a public item covariance matrix using aggregation functions calculated from the original item profiles. This noise is proportional to the sensitivity (maximum variation of the range of the function) of the aggregation functions, the noise is used to bring differential privacy guarantees to the published information.

Even though the differential privacy guarantee protects the privacy of a user, one shortcoming of anonymization strategies is that the masking of the user profile is usually designed for a one-shot perturbation. This means that a trusted curator is in charge of gathering user information and then submitting it for anonymization. In recommender systems the user profile is constantly changing and one-shot anonymization has to be adapted since adding multiple times perturbation to the same data leads to a great degradation of the stored information about the users, affecting the accuracy of the personalization system. If the information is not perturbed continuously, the continuous observation of publicly aggregated information can be used as an attack vector for user privacy as shown by [Calandrino 2011].

Recommender systems usually disclose aggregated information to users in order to help them find new items. For example the IMDB website [1] discloses for each

[1] www.imdb.com

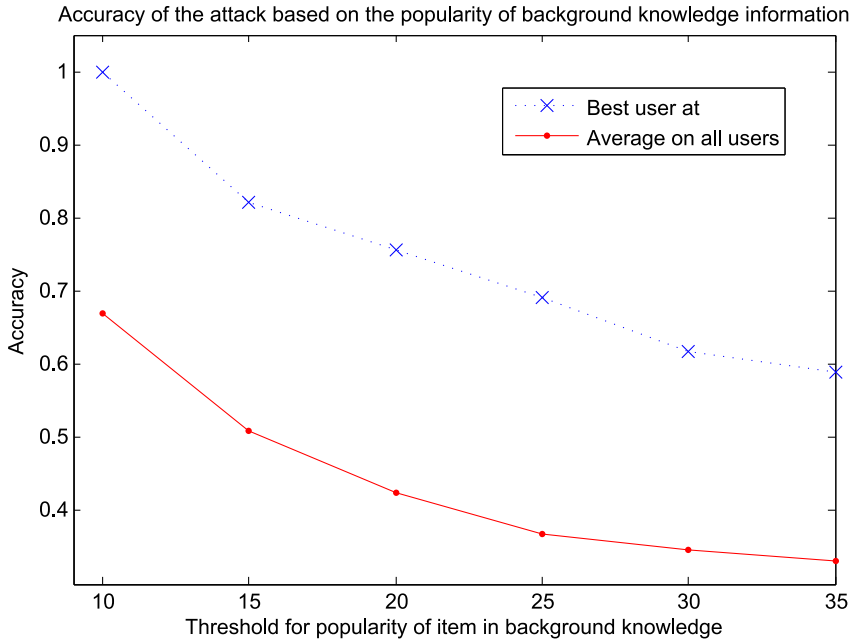


Figure 3.3: Accuracy of the similarity-list attack using background knowledge information

movie a list with similar items, Amazon [2] reveals related products when displaying the page of an item. Generally speaking, it is common for these systems to reveal *item-similarity lists* for each item in the system. By continuously observing the item-similarity list of the items that belong to the auxiliary information of an user, an attacker is able to correctly infer that a target item is in the user profile if the target item appears or moves up in the item-similarity lists for the items in the auxiliary information of the user.

In order to illustrate how this attack works, the MovieTweetings dataset [Dooms 2013] was used to simulate the attack. This dataset collects information from ratings given to movies and series given their IMDB[3] link on twitter. By the time of retrieving the dataset[4] it had 248749 ratings of 28032 users on 16458 items. The dataset was ordered chronologically and split into two parts: The first part took the first 360 days of the dataset and from this part 12 users that had at least 100 positive preferences were selected at random. Once the users were selected, 50 positive preferences for each user were selected from the whole dataset as auxiliary information. A similarity-item list was created for each item in the auxiliary information of the users for different periods of time: one for the first part of the dataset and then one for each period of 7 days. The similarity item list was created using the 20 most similar items to the active one using the cosine similarity of their profiles. If between two adjacent periods an item increases its position in

[2] www.amazon.com
[3] www.imdb.com
[4] As accessed on 15 May 2014

at least one list of the auxiliary information of an user, the item is considered to be marked as inferred by the attack considering the popularity of the background information item. In Figure 3.3 the results of the average accuracy of the attack on the selected uses is shown for different thresholds of the auxiliary item popularity. As expected, the attack has high accuracy when observing the similarity-lists of unpopular items in the auxiliary information.

Synopsis and privacy design choices

In summary, the use of pseudonymous and anonymization in centralized approaches help to reduce some of the exposure risks but are no sufficient. Even if users use pseudonymous instead of their real identity, an attacker that obtains user background information can correlate it with auxiliary information and expose the true user information. Anonymization techniques have shown results for one-shot schemes, but has to be carefully integrated in continual approaches since too much perturbation of the data can turn it unusable, other disadvantage is that it doesn't work well on high dimensionality information and anonymization techniques via heuristic approaches offer no formal guarantees in their capability to stop attackers from correlating information with other sources.

Client-side profiling is a good strategy for avoiding exposure risks but on its own can only be used for CB filtering, to solve this problem two strategies are used: p2p architectures and server aggregation. Sharing unmodified profile information with peers or with an aggregation server reintroduces (although at a lesser scale) exposure risks since each peer or aggregation server is vulnerable of the same risks of centralized entities. For this reason client-side methods protect the user privacy by (1) applying anonymization (perturbation, randomization or substitution of profile information) techniques used in centralized approaches before exchanging it with other entities, and (2) cryptographical measures to calculate *information-secure* operations without revealing the information of the input.

For the first strategy, it was shown in this chapter that a perturbed version of the profile can be easily correlated with auxiliary information, creating a privacy breach. It is also necessary to take into account that perturbation should be consistent in time in order to protect the user from a continuous supervision of its interactions with their peers. Solutions for protecting statistical functions of data under continual observation have been developed [Dwork 2010] but there are no works yet for protecting individual privacy.

For the second strategy no attacks have been identified, however as it will be discussed in Section 3.5 the scalability requirements of these types of solutions are not trivial.

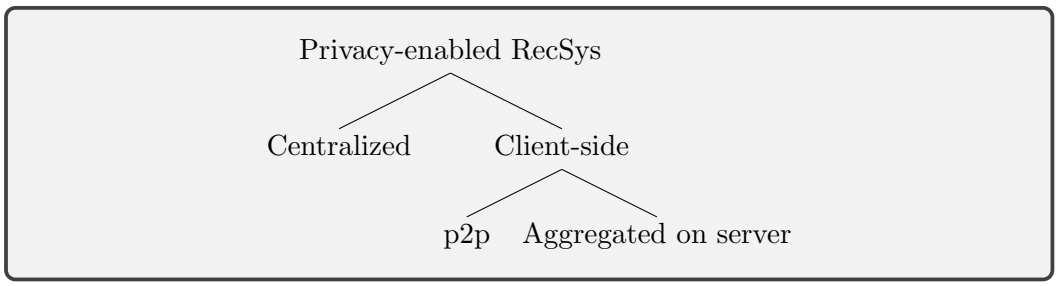


Figure 3.4: Privacy-enabled recommender systems classification

Work	Anonymization technique	Possible RS algorithms
[Parameswaran 2007]	Heuristic transformations	User based similarity recommender
[Chen 2012]	k-anonymity	SVD based recommender
[Zhao 2011]	Heuristic transformation	Item to item similarity
[McSherry 2009]	Differential privacy	Item to item similarity

Table 3.1: Centralized sanitation works for RS

3.4 State of the art on privacy-enabled recommender systems

In this section a survey of privacy-enabled recommender systems is presented. The works are classified based on architectural decisions that were made to keep the privacy of the users. First the centralized approaches that exist to keep user privacy will be presented, next the works that use a client-side agent will be shown. Since client-side agents can interact with each other on a p2p with or without a central server, they will be shown in separated section. Furthermore, if the client-side agent applies any masking, modification or perturbation of the user profile we will define it as anonymized, thus the 5 categories presented in this section. In Figure 3.4 the categories in which the related works are classified is shown :

3.4.1 Centralized approaches

Centralized approaches (Table 3.1) have studied how transform a database of user profiles already collected in a centralized entity for the purpose of publishing it without breaching user privacy. Particularly in the domain of recommender systems this problem has been studied for extracting useful information from the database of user profiles for recommendation purposes.

In [Parameswaran 2007] a series of *Heuristic transformations* are applied for each individual user profile in order to obfuscate it. The user profile information is substituted with information from the nearest neighbors based on a similarity metric, then a geometric transformation is applied to the user profile information

before releasing it.

Other similar approach using a transformation based on *k-anonymity* is presented in [Chen 2012]. Here user profiles are first transformed under a smaller dimensionality using SVD, clustered using a modified version of the *k-means* algorithm and then for each cluster a user profile is published, where each weight in the user profile is the average value given by the members of the cluster.

In [Zhao 2011] a list of each item's neighborhood and the sparsity of the dataset are published, this information is used to boost the predictions of another recommendation system that uses the same items.

Finally, in [McSherry 2009] a transformation based on *differential privacy* is proposed to build a item to item covariance matrix that can be used for privacy-preserving recommendation. The covariance matrix is built based on aggregations from the original item profiles, where noise from an exponential random variable is added proportional to the sensitivity of the aggregation functions.

3.4.2 Client-side approaches with no anonymity on p2p networks

Client-side profiling for recommender systems was first introduced for scalability purposes, in [Tveit 2001] user profile vectors are broadcast over a p2p network and client-based agents cache the most similar vectors to operate a user-based collaborative filtering algorithm using the cached vectors. With the scalability concern in mind, in [Han 2004] the database of user profiles is distributed over a distributed hash table (DTH) which keeps the votes for an item in the same bucket (or peer) so its easier for each peer to publish and find the appropriate ratings for applying a CF locally.

For systems like the one described in [Tveit 2001], one of the concerns of these systems is to make it easier for peers to find similar users on the network and reduce the number of messages in the network while keeping an up to date view of relevant information. Gossip based protocols are used to build an overlay network to connect each peer to other peers that provide her a view of the needed information for recommendation. One popular gossip algorithm to build a p2p overlay network is the *T-MAN protocol* [Jelasity 2009], this protocol uses a similarity measure to keep similar users connected in the network and occasionally contacts a peer in order to exchange peer information and re-configure the peer connected list with the most similar users based on the used similarity metric.

In [Kim 2008] a protocol similar to the T-MAN one is used to actively keep a list of most similar users for each peer using the euclidean similarity metric between their user profiles as the similarity to build the neighborhood. In [Castagnos 2007] users select which local information will be used to build a public profile. This public profile is used to calculate similarities between peers in a similar way to the T-MAN protocol, local predictions are made using a user-based CF system based on the peer's public profile. In [Kermarrec 2010] each user is connected initially to a set of peers found by the T-MAN algorithm using their user profiles similarity. To include information from other users different from the peers directly

connected to the agent a random walk strategy over the peers of the active user is used and prediction is weighted accordingly to the probability of visiting the user and the similarity. In [Ormándi 2010] the T-MAN algorithm is applied with some variations. The authors experiment with 4 different configurations of the T-MAN protocol: (1) Contacting a random user instead of a known peer for the exchange of peer information, (2) Contacting only already connected peers for profile exchange information at random, (3) contact a peer with a probability inverse to the load that that peer has at that moment and (4) contact only the best peer for exchanging information. **The conclusion of this work is that a random selected peer for profile information is a reasonable strategy (as well confirmed in [Bakker 2009]) and that an aggressive selection of peers should be avoided for balancing reasons.**

Not only CF user profiles are shared for building the overlay network, in [Draidi 2011] a CB user profile is propagated over the network for the purpose of informing other uses which topics or interests are preferred by the user, this information is used by an user to know which peers should be contacted for recommendations in a specific topic using their CF profile.

Client-side recommenders have been proposed for pervasive environments where the communication between agents or a peer service are limited or very occasional, this means that is not possible to build an overlay network among peers and they will have to rely on cached views or on historic information to provide recommendations. In [Schifanella 2008] an algorithm for exchanging peer information is proposed based on a modified similarity metric called *affinity*, each time two users are in proximity they exchange their identities and their user profile. The affinity metric is used to keep in their local view only the users with the most similar opinion as the active user. In order to accelerate the convergence process when users meet they exchange also their neighborhood information and could exchange as well a list of previously discarded profiles to help refine each others local view. In [Del Prete 2010] profiles are exchanged in a similar fashion, however the algorithm for predicting relevance is slightly changed: The local user is compared with the community of gathered profiles, if her preferences are considered to be similar to the average of the gathered information then the average opinion of the users is used as the relevance prediction function, on the contrary if its considered different a memory user-based CF algorithm is used with a similarity metric based on how much similar are the deviations of the profile when compared to the gathered profile information. Item based systems can operate under the same principle, in [Miller 2004] when profiles are exchanged they are not cached, instead an item to item similarity matrix is incrementally updated based on the information present in the new profile.

An orthogonal concern on bringing information from other users to client-side personalization systems is to how trustworthy the information coming from other peers is [Massa 2009]. Trust-aware recommender systems use the trust degree expressed by a user (or inferred by the system) to build recommendations. In [Ziegler 2005] a decentralized trust-based recommender system is proposed where each user shares her ratings and expresses her trust belief in other peers exposing a

FOAF RDF document[5]. Propagation measures are used to infer trust in new peers and recommendation predictions are made by using a similarity measure based on the trust on the peer. In [Magureanu 2012] the T-MAN algorithm is used to build an overlay network with peers using a common similarity metric, once an stable neighborhood is found for a peer, the trust for each peer is calculated based on the correlation between the user active ratings minus her average and the peers rating minus the average rating.

The works from this section have been established for memory based approaches which are known to have a smaller predictive power than model based ones, some model based approaches have been created: In [Tomozei 2011] a lower dimensional representation of each user profile is calculated using a gossip protocol on a p2p network. Users that are connected calculate a similarity value on binary ratings, once they have built a similarity with every other agent connected in the network, they start a stochastic gradient descent process using the similarity matrix computed by each peer. At each iteration the users share with each other auxiliary information about their local state. Once they have the local low-dimensional representation of their profile the opinion of the k most similar users for an item is asked, and then an average is calculated as the relevance prediction. In [Isaacman 2011] peers are producers and consumers of content, each producer and each consumer has a profile that is adjusted using an stochastic gradient descent approach at each time the user consumes an item from a producer by adjusting the producers profile with the opinion given by the user as well as the user's profile.

In terms of the risk of user exposure, all the works presented in this section (Table 3.2) leak private profile information to peers. A malicious user connected to the network can gather user-profile information an replicate the problems presented in section 3.1. In the distributed architecture of producers and consumers provided in [Isaacman 2011] the user profile information is only shared with the producer of content, however if the algorithm was applied directly as traditional recommender systems work (where there is only one producer) has the same exposure problems as centralized approaches.

3.4.3 Client-side approaches with no anonymity with aggregation on server

Client-side agents can use third party servers where they upload sanitized information about them in order to receive recommendations from a recommender system that only views the aggregated information (Table 3.3). In [Ali 2004] each local client uploads to a server their local interaction history with scrapped timestamps, the server using the view of all profiles calculates item to item similarity that is later downloaded to the client side agents to generate local recommendations. In [Bilenko 2011] tools for enabling users to sanitize and edit their profile are created before sending the local profile to the aggregation server, similarly in [Aimeur 2008]

[5]FOAF Is an RDF ontology for expressing personal information, as well as connections between people, the project website is http://www.foaf-project.org/

Work	RS algorithm	Contribution
[Tveit 2001]	User based CF	p2p with random peers
[Han 2004]	User-Item memory based CF	p2p DTH rating distribution
[Miller 2004]	Item memory based CF	p2p calculation of partial item to item similarity based on seen exchanged profiles so far
[Castagnos 2007]	User memory based CF	p2p exchange of public partial profiles with information directly chosen by the user
[Kim 2008]	User memory based CF	p2p with overlay using TMAN protocol
[Kermarrec 2010]	User memory based CF	p2p with overlay using TMAN protocol + random walk
[Ormándi 2010] [Bakker 2009]	User memory based CF	Partial calculation of item correlation table with profiles seen so far
[Draidi 2011]	User memory based CF	p2p with overlay using CB user profile
[Schifanella 2008]	User memory based CF	Pervasive exchange with peers in local proximity
[Del Prete 2010]	(Hybrid) Switch User memory based CF or average	Hybrid recommender with pervasive exchange with peers with local proximity
[Ziegler 2005]	User-Item based CF	Trust used to reduce peers for neighborhood formation
[Magureanu 2012]	User-Item based CF	Trust + T-MAN protocol for peers for neighborhood formation
[Tomozei 2011]	Model based lower factor	User factor modeling with only local information exchange between peers
[Isaacman 2011]	Model based lower factor	Online one pass consumer-producer factor modeling

Table 3.2: Client-side approaches with no anonymity on p2p networks

Work	RS algorithm	Contribution
[Ali 2004]	Item memory based CF	Item memory based CF on whole information with pseudonymous
[Vallet 2014]	Model based CF	Server receives ratings of users to update the item profiles at server side
[Bilenko 2011]	Any	Framework for sanitize and edit profile before submitting it to aggregation server
[Aimeur 2008]	Any	Third party framework for recommendation based on partial aggregated profile information
[Lathia 2007]	User memory based CF	User memory based CF with concordance measures over random profiles
[Amatriain 2009] [Ahn 2010]	User memory based CF	Expert-CF with profiles from compilation of public information

Table 3.3: Client-side approaches with no anonymity with aggregation server

a third party framework for recommendation based on partial aggregated profile information is proposed. These proposals place all their trust in central server for profile management and present similar privacy problems that traditional centralized CF recommender systems. An online learning algorithm is proposed in [Vallet 2014] where the system receives at each interaction the past ratings of the user and updates the item profile that is stored at served side.

Client profiling with privacy can be attained with the help of public information. In [Lathia 2007] a set of random CF user-based profiles is created and made available in the server. Each user keeps a local version of her profile and calculates the concordance of the user with each profile in the public dataset, this information can be used to calculate a similarity measure between users based on how many concordant, discordant or tied opinions they share with the random profile set. Despite not sharing private profile information for calculating similarities, a leakage of private information is present since each user has to share to each others the difference between a rating for an item and the user's mean rating in order to calculate the predicted relevance.

Following the strategy of publishing public profiles on a server for calculating similarity measures, Expert-CF was proposed as a client-side profiling strategy to protect user's privacy [Amatriain 2009] [Ahn 2010]: Some CF user-profiles are published on a server reach client applies a memory based algorithm using only the public profiles as neighbors. The public profiles are mined from already public trustable information such as public critics. In terms of exposure risks, this work has the advantage of not leaking user profile information from the client-side since no information leaves the user's device. It is worth noting that these proposals depend heavily on the availability of public information and can be more susceptible to the new item problem.

Work	RS algorithm	Anonymization strategy	Contribution
[Berkovsky 2007]	User based CF	Random perturbation and heuristic rating replacement	p2p network, users that want to help the user send back a modified profile to the active user
[Kaleli 2010]	Model based CF	Random perturbation of groups of ratings	Naive Bayes recommender with groups of preferences perturbed
[Zhan 2010] [Hsieh 2011]	-	Homomorphic encryption and secure dot product	Privacy preserving similarity calculation
[Hoens 2010]	User memory based CF	Homomorphic encryption	Relevance prediction as weighted average of opinion, modified to be able to do division with homomorphic encryption
[Alaggan 2011]	User memory based CF	Homomorphic encryption and random perturbation	Homomorphic secure dot product, reports modified similarity measure based on random perturbation from differential privacy

Table 3.4: Client-side approaches with anonymity on p2p networks

3.4.4 Client-side approaches with anonymity on p2p networks

In order to solve the problems of private information leakage present in p2p approaches, anonymization strategies are used for masking the user profile when communicating between peers (Table 3.4). In [Berkovsky 2007] a user-based CF algorithm like the one in [Tveit 2001] is applied, users make a request for recommendation to other peers and the ones that want to help the user send back a modified profile with random perturbation and some ratings changed, the active user gathers the profiles and applies a local version of CF. In [Kaleli 2010] a naive Bayes recommender is used for recommendation with profiles with binary preferences. First, a user requesting recommendations creates for each peer a masked profile changing randomly the values of groups of preferences on her profile vector; next, each peer receives the altered profile and a number indicating the groups of preferences. For each possible permutation the peer calculates the probability of agreeing with the active user for both classes (liked or disliked). Finally a probability for each possible permutation of the user profile is reported back to the active user who makes a prediction using the real profile values.

Trying to solve the leakage of private information in p2p approaches, some works have been created to establish similarity between pairs of users without revealing private information, two approaches are found in [Zhan 2010] [Hsieh 2011]: Homomorphic encryption and secure dot product between profiles. Homomorphic encryption allows a system to operate on encrypted data and the result of the operation can be decrypted matching the operation carried on the cyphered message, for example if *Enc* is the encryption function and *m1* and *m2* are the arguments of an

operation then $Enc(m1) \times Enc(m2) = Enc(m1 + m2)$. This property allows a pair of peers to calculate their Pearson similarity without disclosing private information (Equation 2.5). On the other hand a secure dot product relies on a third party random generator that creates 2 random vectors $Rand_u$ and $Rand_v$ and two random values $rand_u$ and $rand_v$ such that the dot product between the random vectors is the sum of the random numbers ($\langle Rand_u, Rand_v \rangle = rand_u + rand_v$). Thanks to these values users can mask their information and find out the dot product without knowing directly the weights of the peer's user profile.

Homomorphic encryption is used in [Hoens 2010] to keep privacy between peers where prediction the is the weighted average opinion of the peers. Keys are generated in a multi-threshold encryption scheme [Pedersen 1991] where the active user and her peers create a public key and share the private key, so that in order to decrypt the message all peers must participate in the process. Each peer submits its encrypted rating and a weight using the public key, or can forward the request to its peers to get more information about the item, then answers with a weighted average of her local information and the reported ratings from her peers. The active user multiplies encrypted responses to adds up the weighted averages reported by each user. Decryption is made by collaborating among the peers and the user learns the final value for preference. In this work a modification of the protocol is done to handle division with homomorphic encryption inputs in a multi-threshold scheme. [Alaggan 2011] presents a work of similarity calculation between users preserving privacy also using the multi-threshold scheme. The work calculates the dot product between the encrypted profiles of the user and a peer and adding a random perturbation into the similarity result in order to protect user individual privacy since two users with identical ratings or with no shared ratings will give a similarity of 1 or 0. The noise is introduced to avoid these scenarios.

3.4.5 Client-side approaches with anonymity with server aggregation

Two strategies exist for aggregation servers: (1) They can use the masked profile information to provide recommendations as centralized systems (Table 3.5) or (2) the aggregation server can participate in a multiparty calculation of aggregated information that is useful for each client-side agent to calculate recommendations (Table 3.6).

One of the first approaches to keep privacy in client-side approaches was to submit a modified version of the profile to the centralized entity, in [Polat 2005] each client-side profile weight is normalized and random perturbation is added before being sent to the aggregation server that executes a SVD algorithm with the masked profiles. Extensions of this work have been proposed: [Dokoohaki 2010] follows this strategy but uses trust weights for relevance prediction, and [Renckes 2012] uses a clustering algorithm in the server based on user similarities.

The amount of perturbation applied to the user profile is studied as well by [Halkidi 2011]. In this work the problem of how much perturbation should be

Work	RS algorithm	Anonymization strategy	Contribution
[Polat 2005]	SVD CF	Random perturbation and heuristic rating replacement	Perturbed ratings submitted to centralized entity for prediction
[Xin 2014]	SVD CF	Privacy mechanism	SVD decomposition from public profiles, updated with information from private profiles protected by a privacy mechanism
[Dokoohaki 2010]	Trust user-based memory CF	Random perturbation and heuristic rating replacement	Trust calculation with perturbed user profiles
[Renckes 2012]	Model based clustering	Random perturbation and heuristic rating replacement	Cluster calculation with perturbed user profiles
[Halkidi 2011]	Any - Item to item correlations are pre-defined and known	Perturbation based on maximum error allowed by recommendation	Perturbation of local profiles expressed as a Nash equilibrium problem
[Parra-Arnau 2012]	Any	Change or faking of reported opinions to the centralized entity	Privacy quantified as Shannon entropy, CB profile changes alert the user to report a changed or a faked interaction
[Shokri 2009]	Any	Merging of local information with vectors seen so far in p2p network	Privacy and low error on prediction power merging with most similar users seen so far
[Elmisery 2011]	Any	Heuristic transformation	Two level masking and obfuscation using algorithms that preserve locality

Table 3.5: Client-side approaches with anonymity with aggregation server operating on masked profiles

Work	RS algorithm	Anonymization strategy	Contribution
[Canny 2002]	SVD CF	Homomorphic encryption	Gradient descent to calculate SVD of rating matrix expressed as addition of contribution of each user, addition of the contributions is done using homomorphic encryption
[Duan 2010]	SVD CF	Private information affected by operation with random vector (differential privacy)	Framework for privacy with formal guarantees
[Erkin 2012]	User based CF	Homomorphic encryption	Two step encryption of user profile, one for neighborhood formation and other one for specific values of the profile used for relevance prediction.

Table 3.6: Client-side approaches with anonymity with aggregation server for infrastructure collaboration

applied for each profile is considered as a game theory game where each user distorts her declared profile to the centralized entity trying to achieve a Nash equilibrium between the accuracy of the system and the amount of distorted information. Other strategy for modifying the user profile is considered by [Parra-Arnau 2012] where a local based CB profile is used to alert user that a privacy breach might occur if an opinion for an item is expressed, and it is suggested to the user that she should change the reported opinion of an item or give a fake rating to an item to help he preserve her privacy.

Other approach to perturb the local user profile reported to the centralized entity is proposed by [Shokri 2009]. Users are in a p2p setting and exchange their profile information in the clear, but instead of producing recommendations with the gathered information, they use this information to merge their information with the most similar vectors seen so far before submitting their profile to the aggregation server. This work protects the user privacy from the aggregation server but not from a malicious peer.

In [Elmisery 2011], Elmisery et Al. applies a heuristic local transformation of the profiles before sharing them to the centralized entity. The client-side agent generates smaller clusters of preferences in the user profile vector and applies a transformation that masks it while keeping the distances in the preference cluster. Then the user profile is sent to an aggregation server that transforms the clustered representation of the profiles to a one dimension vector using Hilbert curves and swaps profile information between users.

Finally, in [Xin 2014], make use of already public information to calculate the

SVD decomposition of the known entries. The decomposition is then used by each client-side agent to calculate the recommendations for each user. In order to improve the accuracy of the decomposition, private agents can send a vector that is protected by a privacy mechanism whose output preserves a fixed distribution of ratings (for example the output preserves a normal distribution of ratings), therefore protecting the privacy of the user. The output of the privacy mechanism also guarantees that the SVD estimate that the server will calculate using the reported vector will improve the SVD decomposition accuracy.

Under the scope of user exposure risks, all these works trust a centralized entity vulnerable to the exposure risks presented in section 3.1. Other problem is that either the masked approaches presented so far in this section don't offer formal guarantees against a background knowledge attack if an attacker uses the information gathered in the centralized entity, or they don't provide the same level of privacy to all their users.

Other approach that leaks no profile information gathering private profile information on a server are possible: In [Canny 2002] users collaborate to build a matrix that represents the items in a lower dimensional state as in the SVD decomposition. The approximation to this matrix representation can be calculated as a gradient descent problem where the gradient function can be expressed as a sum of the contributions of each peer local information. The peers collaborate using a multi-party server calculation based on a multi-threshold encryption scheme [Pedersen 1991] to generate private shared key and public key between peers. At each iteration, users transmit to an aggregation server their part of the calculation of the gradient and in order to avoid profile exposure this is transmitted using homomorphic encryption on the generated public key. The server adds up the encrypted contributions of the users and they collaborate to decrypt the new value of the matrix. The incremental model of [Miller 2004] rate between(in the p2p with no anonymization section) also considers a homomorphic encryption scheme to obtain the correlations between items operating on the encrypted representations of their profiles.

In [Duan 2010] Presents a multi-party framework for learning using the statistical query model and illustrates its use calculating SVD of a matrix motivated by the recommendation problem. The authors use a similar approach as Canny where an aggregation server adds up each user contribution to the problem, but don't use homomorphic encryption. Instead they rely on at least two servers: Each user before sharing information creates a random vector of equal size of the shared information, then she calculates the difference between her information and the random vector and submits the difference mod ϕ to the first server, next she sends the random vector to the second server. The server that receives all the random vectors adds them up and submits the aggregation to the first server, where it adds up the received information from all users with the random vectors to obtain the sum mod ϕ of the original data vectors. The sums of profiles are the only information exposed and its result is shown to be protected of a background knowledge attack by *differential privacy* guarantees.

Finally, [Erkin 2012] introduces a third party entity to provide privacy to users

and interact with the recommendation system. Users use two profiles, one with the ratings assigned to the most popular items in the system and one with the rest of the ratings. Users encrypt their profiles with the public key of the third party privacy provider and send their encrypted profiles to the recommender system. Next the privacy provider and the recommender system interact to build a user neighborhood for each user using the encrypted versions of the users profiles operating a similarity measure with homomorphic encryption using the smaller profile. After the neighborhood for each user is found, the average opinion is calculated by the recommender system using homomorphic encryption. The relevance prediction for each item present in the neighborhood is transmitted to each user encrypted and a secure decryption algorithm is used between the third party and the client to decrypt the relevance values without revealing the recommendation values.

These proposals are secure in terms of leak of private information since the encryptions are semantically secure, which means that no information can be inferred about the original message by viewing the encrypted version. However, as it will be described in Section 3.5 these approaches need a solid infrastructure behind them in order to manage the high synchronization protocol designed for the interaction between peers and server in order to collaborate. Therefore are suitable for architectures where transmission of data is reliable and each peer has a high processing capability, for example in a cloud computing architecture.

Synopsis and privacy design choices

A classification of privacy-enabled strategies for recommender systems was presented in this section and evaluated in terms of the exposure of the user's profile information. Centralized approaches rely on a trusted server that receives the whole user profile database and publishes a sanitized version of them, usually in an aggregated format. However in terms of the exposure risks a centralized entity is still vulnerable to the privacy risks presented in Section 3.1, therefore the proposed architecture that will be used in this thesis consists on a client-side agent that keeps a local user profile.
On the client-side classification, agents organize themselfs in p2p networks to exchange profile information, or rely on an aggregation server to exchange information useful for recommendation. Both strategies can be further classified by telling if the agent applies an strategy to mask the profile before leaving the client-side or not. Since a deceptive peer or a deceptive server can receive unmasked information and replicate the attacks on privacy of a centralized entity presented in Section 3.1, the chosen architecture for the recommender system presented in this thesis will apply a masking strategy in order to keep the attackers from learning the user preferences.
Masking techniques can be classified into three general approaches: Random noise generation, cryptographical approaches and heuristic masking. In order to choose which one of the approaches will be used, their scalability against the scaling factors on the recommendation scenario will be analysed in the next section.

3.5 Privacy and scalability

After reviewing the existing technologies for privacy-preserving recommendation, its time to look into the scalability of the chosen anonymization approaches used by the works presented on the previous section. Centralized approaches are vulnerable to the mission creep scenario and client-side architectures that share un-modified profile information are vulnerable to the attacks presented on Section 3.3, this section will focus on client-side systems that modify or mask the user profile before sharing it with peers or with an aggregation server.

3.5.1 Scalability of random noise generation

Extending the work of [Agrawal 2000], random noise generation is a common strategy for masking profile information before leaving the client-side agent. The computational complexity of this type of profile masking depends on the complexity of the random number generation process involved and the number of times during the recommendation process that this masking is applied.

Pseudo-random number generators are algorithms that based on an initial

state (or *seed*) generate a sequence of numbers that simulate to come from a distribution, usually imitating a uniform-distribution random variable between 0 and 1 [L'Ecuyer 2007]. If the numbers need to come from another distribution a transformation based on the *inversion method* is applied. Since the number of states of the pseudo-random number generator is finite, the sequence of numbers generated is eventually periodic. The computational complexity of a pseudo-random number generator depends on the size of the state space it has to generate the sequence; smaller spaces may cause a low period for the generator, thus making it invalid for the application in practice.

The Mersene twister algorithm [Matsumoto 1998] is an uniform pseudo-random number generator known for its low state to period relation: It has a period of $2^{219937} - 1$ with a state of $N = 623$. The algorithm to generate the sequence first iterates through each one of the states, initializing them with an $O(1)$ operation. Once initialized, it can generate a number with each one of the states as an input until all states are used and a new initialization is needed, therefore the process of a single random number generation with the Mersene twister algorithm has an amortized complexity of $O(1)$.

The simplest strategy used for adding noise to protect user privacy in client-side recommender systems was used by [Polat 2005] where noise from a Gaussian distribution was added to the known ratings of the user profile before sending it to the centralized server. [Berkovsky 2007] also induces a change of known ratings with either a default value, a random value taken from an uniform distribution or by a Gaussian distribution reflecting the rating distribution in the dataset, both of this works mask their profile before sharing it with an aggregation server. Since a lot of information is revealed just by letting know the attacker which items the user has interacted with, this strategy was later extended to mask if the user has interacted with an item in the past. In [Dokoohaki 2010] and [Renckes 2012] ratings of the user are normalized using a z-score (number of standard deviation that each rating deviates from the mean) and then a random number of unrated items is rated with noise from an uniform distribution. While [Dokoohaki 2010] shares the masked vector with trusted peers, [Renckes 2012] shares it with an aggregation server. All these works propose a one-time perturbation of the profile before sharing it with their peers or the aggregation server.

Finally, [Duan 2010] proposes a framework using at least two servers (S_1 and S_2) for aggregating a summation of many private values and uses this framework for the calculation of the SVD decomposition of the rating matrix. Taking the whole rating matrix V, the symmetrical matrix $V^T V$ is constructed. By taking the SVD decomposition model (Equation 2.7) and the fact that matrices U and V^* are orthogonal, the symmetrical matrix can be factorized as $V^T V = V^* \Sigma V^{*T}$. The author proposes to use the framework to calculate de SVD decomposition of matrix $V^T V V^* = \Sigma V^{*T}$, where each row in matrix V^* is the low-dimension representation of each one of the items present in the system. Since each user knows its own ratings (V_u) matrix, the matrix $V^T V V^*$ can be seen as the addition of an operation of the

local information of the user ($\sum_{u \in U} V_u^T V_u v$).

First, the vector v is submitted to each one of the peers on the system, then each user calculates its own part of the addition $d_u = V_u^T V_u v$ and generates a uniform random vector to hide the result of this private calculation rd_u. Each user calculates $d_u - rd_u \mod \phi$ and submits the result to server S_2 and rd_u to server S_1. S_1 computes the aggregation of the noise $\mod \phi$ and S_2 computes the aggregation of the d_u's of each user, the sum of the results of the aggregations is indeed the sum of the private information of all users. The result is passed to a high-performance eigensolver that returns the k largest eigenvalues of the submitted matrix, that are used to build the vector v and repeat the process until it converges. Once it converges the SVD of the resulting matrix is calculated and the matrices Σ and V^* are published. By using the ratings of the user and the item representation matrix V^*, a local algorithm can be applied in order to discover the relevance of an unseen item, for example by using a item memory based system.

Recapitulating, in Table 3.7 the complexities of the process of masking the user profile are shown. Random number generation is a cheap operation and for most of the works is only applied once. In [Duan 2010] it is applied multiple times but by itself doesn't affect the scalability of the proposed algorithm.

Work	Complexity of masking profile process
[Polat 2005]	$O(\|R(u)\|)$
[Berkovsky 2007]	$O(\|R(u)\|)$
[Dokoohaki 2010]	$O(m - \|R(u)\|)$
[Renckes 2012]	$O(m - \|R(u)\|)$
[Duan 2010]	$O(m * t)$

Table 3.7: Random noise masking process complexity

3.5.2 Scalability of homomorphic cryptosystems

As explained briefly in Section 3.4.4, *Homomorphic encryption* is used by privacy enabled recommender systems as a cryptography based form of masking the user profile information when sharing it with other peers. Homomorphic encryption allows an agent to make computations on cyphertexts of messages, where the decryption of the result of the computation on the cyphertexts matches an operation on the messages without encryption. In order to adequately protect user information it is crucial that the chosen schemes are *semantically secure*. Semantically secure schemes don't provide any useful information about the plaintext that it encrypts. This means that a peer or an agent can make operations on encrypted data without learning anything from the plaintexts (in this case the user profile information).

An *additively homomorphic* encryption scheme allows a system to make an operation on two cyphered messages such that the decryption of the cyphertext result is the sum of the original messages , let pk be the public key of the cryptosystem and sk the secret key of a user, an additively homomorphic encryption scheme is

defined as follows:

$$\mathcal{D}_{sk}\left(\mathcal{E}_{pk}(m_1) \odot \mathcal{E}_{pk}(m_2)\right) = m_1 + m_2 \tag{3.2}$$

Additively homomorphic encryption schemes with the multiplication operation allow that by exponentiating a cyphertext by a constant a the decryption of the cyphertext result is the multiplication of the plaintext by the constant.

$$\mathcal{D}_{sk}\left(\mathcal{E}_{pk}(m_1)^a\right) = m_1 \times a \tag{3.3}$$

The Paillier homomorphic cryptosystem [Paillier 1999] is an widely used additive homomorphic cryptosystem with the multiplication as operation. The encryption of a message $m \in \mathbb{Z}_n$ under the Paillier cryptosystem with public key $pk = (n, g)$ and private key $sk = (p, q)$ where p and q are two large primes such that $n = p \cdot q$ results in a cyphertext message defined as:

$$\mathcal{E}_{pk(n,g)}(m, r) = g^m \cdot r^n \mod n^2 \tag{3.4}$$

Where r is a random number (that allows the semantical secure property) and $g \in \mathbb{Z}^*_{n^2}$[6] is a semi-random number which generates a subgroup of order n. To confirm if g is compatible with the encryption scheme, the *least common multiplier* of the numbers $p-1$ and $q-1$ is calculated, $\lambda = lcm(p-1, q-1)$ and g is used if the *greatest common divisor* of $(g^\lambda \mod n^2 - 1)/n$ is 1.

The decryption protocol takes as input the private key $sk(p, q)$ and g has three steps: First k is calculated as $k = (g^\lambda \mod n^2 - 1)/n$, then the inverse of k under the modulo of n is calculated as $\mu = (k^{-1} \mod n)$, finally the decryption of the cyphered message c is calculated as $m = (((c^\lambda \mod n^2 - 1)/n)\mu \mod n)$.

The Paillier cryptosystem scheme was adapted to work in a *threshold cryptosystem* in [Damgård 2001]. A (t, n) cryptosystem uses as well a public key $pk(n, g)$ but shares the private key into n shares, and it needs at least t of the shares to decrypt a message cyphered with the public key. By using this cryptosystem peers are able to collaborate with a centralized agent that carries out the operations on the cyphered domain and then users collaborate to decrypt the result, or on p2p networks where a public and threshold private key is generated between peers to allow both users to calculate a similarity value between them.

# op	encryption	plaintext sum	encrypted sum	plaintext mult	encrypted mult
100	1055367947	1785	4899948	2231	4537618
1000	10734646890	14726	49551013	31681	42297697
10000	1,06354E+11	144576	238901601	254345	213789645

Table 3.8: Time in nanoseconds of different number of operations using plaintext and encrypted data on the Paillier scheme

While the complexity of key generation, encryption and decryption processes of this scheme has to be taken into account, the most limiting aspect on the use of

[6] $\mathbb{Z}^*_{n^2}$ are the invertible elements of set $\mathbb{Z}_{n^2}$

homomorphic encryption is that the encrypted operations are carried out in a much larger space than the original one. Typically the message space is 1024-bits since this is the size of the key that offers an acceptable space to protect the cypher from an attack. While a sum or a multiplication of 32-bit or 64-bit doubles is highly optimized in current hardware, carrying over a sum or a multiplication on two cyphertexts involves operations of numbers represented over 1024 bits. This has an important impact on the scalability of algorithms since similarity measures between users rely heavily on multiplications over the user or item profile (Equation 2.5), as well as the training algorithm for adjusting the weights in latent factor systems (Equation 2.8 and Algorithm 1). Operations on cyphertexts take an order of 10^3 more time than on plaintext, in order to illustrate this effect a benchmarking algorithm was executed comparing the same operations on the plaintext and encrypted versions of the numbers using the thep[7] libray's implementation of the Paillier scheme (Table 3.8).

The first work to propose the use of homomorphic cryptography to hide the user profile in recommender systems was [Canny 2002]. In this work a low-dimension approximation matrix is calculated as a row-orthonormal matrix that approximates in a least-squares sense the original matrix V. In order to find this matrix and given the error $e = tr(VV^T) - tr(VA^TAV^T)$, where tr is the trace of a matrix, the author proposes to maximize $tr(VA^TAV^T)$ by using a iterative conjugate gradient method. The author shows that the gradient of the solution at each step can be calculated as the sum of the gradients that depend on private information of each one of the users (Row V_i) and the past state of the aggregate A as $G_i = AP_i^TP_i(IA^TA)$. The aggregation server receives the encrypted gradients of all users, multiplies their encrypted versions in order to sum them and then peers collaborate to decrypt the aggregated gradient and make a new estimation of the matrix A based on the gradients. Each G_i has a size of $k \times n$ where k is the reduce dimension and n is the number of items.

Another aggregation on the server is made by [Alaggan 2011]. In this user-based CF work there are two different servers, one that provides the recommendations called the recommendation service provider and the other in charge of carrying out some computations to keep the privacy of the protocol called the privacy service provider. Users send to the recommendation server two parts of their profile encrypted with the public key of the privacy service provider: One with a set of R of the most common global items in the system (v_u^d) which is expected to be dense and the other with the rest of their preferences (v_u^p) which is sparse. The recommendation service provider calculates the similarity between all users on the encrypted versions of the dense profiles (v_u^d) and by comparing the encrypted similarity with a public threshold δ with the help of the privacy service provider creates an binary encrypted variable γ_{uv} that is congruent to 1 if user v is similar to user u, or 0 otherwise. Once the list of neighbors for a user has been established a multiplication of the binary encrypted values to each one of the other users profiles is done to remove the vectors

[7] https://code.google.com/p/thep/

that are not going to be used for the computation, finally a multiplication of the element-wise vectors is applied over the encrypted space to sum up the ratings of the users in the neighborhood. The aggregated vector and the number of users on the similarity list is then returned to the user that requests a recommendation. The user interacts with the privacy service provider in order to decrypt each one of the coordinates of the aggregated vector, then it divides each decrypted coordinate by the number of profiles aggregated to obtain the relevance predictions.

On the p2p setting [Hoens 2010] proposes a system where a user asks for her peers to collaborate with her in order to obtain a weighted average on their opinion on items. The user and their peers collaborate under a threshold cryptosystem, when the user requests for an opinion on a rating for its peers, each peer responds two encrypted values under the public key generated for the scheme: A rating (s_{ui}) and a weight (w_u). A user has three options to collaborate with a peer: Respond with an empty tuple $(0, 0)$, respond with its real information $(V_{ui}, 1)$ or to respond with its own information aggregated with the information from its peers (s_{ui}, w_u). In order to include the information from its peers and its own into the response, a weighted average is calculated where active user gives weights $\hat{w}_v$ to each one of her peers and a weight $\hat{w}_u$ to her own contribution: The reported weight is calculated as $w_u = \hat{w}_u + \sum_v w_{vi}$ and the reported rating is calculated as $s_{ui} = V_{ui} \cdot \hat{w}_u + \sum_v s_{vi} \cdot \hat{w}_v / w_{vi}$. When the active user receives all the tuples from her peers, she aggregates the values from each peer applying the same process and predicts with s_{ui}/w_{ui}. All the aggregations are carried out on the cyphered information, however the secure division protocol is a process done in coordination with the peers, this process is done as a bit-wise long division calculation: First a transformation of the encrypted denominator to a bitwise encrypted representation must be applied, this transformation is linear in the number of bits (ℓ) selected to be compared and must be calculated by the group that shares the private key. Then the current remainder (initially the numerator) is transformed to a bit-wise representation, then peers collaborate calculating the comparison of the bit-wise encryptions of the remainder and the numerator and transmit their result to the active user, finally the active user aggregates the information from the peers to construct the final bitwise representation of the quotient and the remainder. The process of homomorphic division drives the complexity of the algorithm since the comparison process is called ℓ times at &ell rounds.

Since the addition and multiplication on encrypted profiles drive the complexity of operations drive the process, the complexities of the works are presented for the number of multiplications and exponentiations made on encrypted data (Table 3.9). In [Canny 2002] the server aggregates the $k \times n$ sized gradient contributions of each one of the m users for each one of the iterations t needed until convergence, the gradient step and the factorization of the matrix is done in plaintext data. For [Erkin 2012], the most-similar user search is what drives the complexity of the algorithm, the multiplication of user profiles to obtain a similarity value needs $O((n-1)R)$ exponentiations and the multiplication of binary variables γ_{uv} to the corresponding sparse vectors of the user profiles to cancel out the addition of users

that are not similar to the active one for the computation needs $O((n-1)(m-R))$ exponentiations per user. These complexities are reduced by designing a profile packing strategy that packs into a single encryption representation numerous values instead of one. For the similarity calculation by packing into a vector for each item in R all the opinions of all the users for the item j called v_j^c. The similarity calculation between users can be expressed as the dot product of v_j^c and v_i^d, which requires only R exponentiations. For the screening out the users that are not similar and for facilitating the aggregation process each opinion of neighboring users, the sparse vector v_u^p is not encrypted element-wise but condensed on a vector that condenses the opinion of the user under a single vector, so that when multiplying the dense representation by the binary variables only $O(n)$ exponentiations per user are needed. Since not all the opinions of an item might fit into only one vector given the key space, this number is determined to be S_2, analogously the number of packed vectors that represent the sparse opinions of each of the user is determined by number S_1. Complexities are multiplied by m since it a process that has to be done to each one of the m users of the system.

Finally for the p2p system in [Hoens 2010], let $\hat{m} \ll m$ be the number of peers that report a rating and a weight to an active user. The first step to obtain the relevance value is to divide each (s_{ui}, w_u) tuple, job delegated to the $\hat{m}$ peers that handle the divisions of the encrypted rating by the encrypted weight, each peer needs to execute at least $O(\ell)$ exponentiations in order to calculate the secure quotient, each peer also communicates if the division was invalid or not by returning a bit b_v (division by a weight of value 0). Then the active user chooses weights $\hat{w}_v$ to re-weight each one of the reported ratings, they are brought to 0 by doing an exponentiation of each weight w_u by b_v using $O(\hat{m})$ exponentiations and a final encrypted weight aggregated using $O(\hat{m})$ multiplications. A similar strategy is applied to calculate an encrypted aggregation of the quotients by doing an exponentiation of each quotient by b_v and aggregating the values by $O(\hat{m})$ multiplications. A final secure division between the aggregated quotients and weights is the final recommendation for the user.

Work	Location of calculation	Multiplications	Exponentiations
[Canny 2002]	Server	$O(t*k*n)$	-
[Erkin 2012]	Server	$O(m(n*\hat{s}_1 + R*\hat{s}_2))$	$O(m*n*R)$
[Hoens 2010]	Each peer	$O(m*n(2\hat{n}))$	$O(m*n(\ell+2\hat{n}))$

Table 3.9: Computational complexity of training process with homomorphic profile masking for a single user

As shown in Table 3.9, the number of multiplications and exponentiations in current systems based on homomorphic encryption depend linearly with factors that scale such as the number of users and items in the system. However due to the high cost of each operation (Table 3.8) these systems are only available for medium-sized systems. Extrapolating from the results obtained in (Table 3.8) a system that does only $(n*m)$ operations on a medium to large dataset $Movielens10M(69878, 10676)$

would take around 265 minutes to finish the calculations.

3.5.3 Scalability of heuristic-based perturbation

Other ideas for perturbing the user profile before leaving the client-side agent are based on heuristics that come from different disciplines.

A way of defining privacy is to "melt-in the crowd". In [Shokri 2009] peers share their ratings and aggregate their reported profile with information from their peers using different policies, this strategy is linear on the number of items in the system. Unfortunately, while hiding from the server in the crowd, this scheme doesn't protect the user from curious peers.

A similar idea was proposed by [Halkidi 2011]. Here users trust a centralized server to distribute a public profile that is built as a perturbation of the ratings on the original sparse profile. Each user receives a set of public profiles from other users and then calculates how much different the predictions on the items would have been by contrasting the difference between the predictions on the true ratings of the user and the predictions on the perturbed ratings of the user. By bounding the difference of the predictions the user maximizes the ammount of perturbation on each rating to maximize the privacy of each user until the difference is acceptable. The authors show that if each user follows this protocol for a number of rounds a Nash equilibrium between the privacy and the quality of the predictions is achieved. In terms of scalability can be seen as an iterative random noise generation to protect the user profile, however it is unclear how many iterations are needed to reach an agreement on real life sized examples.

Finally, in [Kaleli 2010] a well known strategy to preserve privacy for users in surveys is adapted to recommender systems. The work proposes a p2p system where users have an implicit profile (A 1 is assigned in the user-item interaction if the user has liked the item, 0 if not). The masking algorithm assigns a 1 for a random percentage of the unrated items of the user profile. Once the whole profile is masked, the system reports to each peer a modified version of the profile by using the *Randomized Response Technique* [Warner 1965]. This technique was initially designed for asking individuals in surveys yes or no answers for sensitive issues without compromising the privacy of the subject. Users must respond positively to a question with probability p regardless if the answer is true or not, providing to each subject plausible deniability of their answers. The user divides her profile in groups of minimum 3 or maximum 5 adjacent ratings and for each one of the profile subdivisions the user reverses the preferences of the subdivision with probability p. Each peer receives from the active user a modified profile, the number of groups the profile was divided into and a target item for which the user wants a relevance prediction, the peer returns the probability that the active item is relevant based on each one of the possible states of the original profile. If adj is the number of adjacent ratings masked, the active peer must make a probability estimation for each one of the $2^{n/adj}$ possibilities of true profiles, therefore the scalability of the system as the number of items increases is affected since its execution time depends

exponentially on the number of items present in the system.

Synopsis and privacy design choices

After explaining the prohibitive impact on the scalability of the system that cryptographical tools have when protecting the user profile, the options left open to mask the user profile are random noise generation and heuristic-based perturbation. The approach that will be presented in this thesis will be based on both a heuristic-based perturbation of the profile that will reveal only a part of the user profile (Chapter 4), and in order to refrain an attacker to make attacks using the revealed information, the revealed information will be masked by a random noise perturbation strategy (Chapter 6).

3.6 Conclusions

This chapter presents the strategies used to bring privacy to users in recommender systems under the perspective of avoiding exposure risks. Centralized approaches extract a sanitized version of the complete database of user profile information that serves for recommendation purposes without disclosing personal information. In the light of risks of user exposure, the centralized gathering and processing of user profile information can be susceptible to all the risks exposed in Section 3.1.

Client-side profiling is a common solution to avoid the disclosure of personal information by transferring the collection, gathering and storage of profile information to the user device. CB filtering doesn't present a problem when adapted to client-side personalization since it relies only on local history, but for CF and hybrid approaches mechanisms for guarding the user profile information must be taken into account since other clients need information about what other users have done. It is argued that client-side profiling is not enough to avoid exposure risks since most of the works presented here leak private profile information to their peers, therefore a malicious user can deviate from the expected protocol and gather information from their peers, replicating the exposure risks present in central entities. From these works, only the ones based on *Expert CF* present no risks in term of exposure since they don't leak private information, however these systems are subjected to the availability of public data, which aggravates the new item problem if the system is deployed on a domain where items appear frequently.

In order to control the leak of private information on client-based CF systems, clients apply masking to protect their privacy before sharing it with other peers (Table 3.4) or with the aggregation server (Tables 3.5 and 3.6). Three strategies are applied to mask the user profile on client-based systems: Random vector perturbation, homomorphic encryption and heuristic-based perturbation. In terms of privacy guarantees only homomorphic encryption offers formal guarantees on the inferences an attacker can make with background knowledge, however, as seen in Section 3.5 its computational cost makes its application impractical on systems where the number

of users, items and user-item interactions are expect to scale.

Random noise generation on the other hand doesn't affect the scalability of the system, contrary to the checked heuristic approaches that strain the scalability of recommendation algorithms although a carefully chosen heuristic-based perturbation of profiles could be used if chosen carefully. The biggest problem of random noise generation for client-side anonymization is that users share sparse vector of user profile information. As seen in Section 3.3, users in recommender systems are far from each other statistically by looking at them by their choices because the user profile information is sparse (no user has a significant number of opinions compared to the available items in the system). Too much noise can transform the observed distribution of user profiles to a fictitious one and if noise is added in a way that makes user more alike the utility of the information will be affected as well, affecting the predictive performance of the system. The learned lesson is that random noise generation can easily be adapted as a mechanism for providing privacy and scalability as a masking strategy to information leaving the client, however it shouldn't be applied on sparse user preferences; instead CF algorithms should be modified so that information leaving from the client are not sparse representations of the user profile that can be easily correlated with background information.

Synopsis and final remarks

As seen in this chapter, privacy enabled recommender systems impose architectural restrictions to the traditional architecture of recommender systems. After reviewing the different configurations used for privacy enabled recommendation under the light of user exposure risks, client-side agent architectures must be used in order to avoid user exposure risks, and anonymization techniques should be applied in order to mask the user information that leaves the user profile.

Since the researh objective of this thesis is to achieve an scalable privacy enabled system, one must be careful when appliying a masking strategy. In this chapter it was explained that the cryptographical tools do not scale well with the scaling factors of recommender systems and thus should be avoided. In the next chapter a predictive model that uses a heuristic-based strategy for protecting the client-side agent privacy will be explained and its performance in terms of compuational complexity an predictive performance will be explored. Further analysis on how to improve the privacy provided by this model will be analysed on Chapter 6.

Part II

Model architecture and performance

Chapter 4

A CF client-side recommender system

Contents

As seen in the previous chapter, the centralized gathering and processing of user information made by traditional recommender systems can lead to user information exposure, violating her privacy. Client-side personalization systems are a privacy by architecture solution to the problem since this architecture removes the risks associated with a centralized entity managing user profile information. On Chapter 3 there were identified problems related to the privacy and scalability of current client-based solutions. Motivated by these findings, and keeping into account the predictive accuracy-scalability tradeoff explained on Chapter 2, in this chapter a privacy-enabled scalable CF system is presented.

4.1 A client-side agent for privacy-enabled recommender systems

In order to remove the centralization of user-item interaction present in most recommender systems, a client-side agent is needed in order to limit the exposure risks of user profile information. The client-side agent is in charge of keeping up to date the user profile, as well as giving the necessary information to the recommendation server to keep the item profiles up to date without revealing the opinions the user has expressed on the items. One of the most limiting factor of using client-side agents for personalization systems is that in order to apply prediction

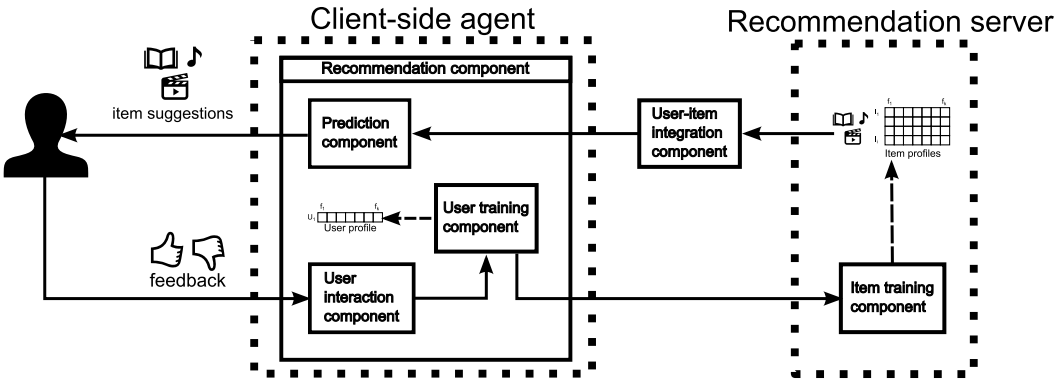


Figure 4.1: Proposed architecture for recommender system

models that operate using the whole user-item interaction database a redesign must be applied since each client only has its own user-item interaction log [Kobsa 2007]. The proposed model in order to bring a CF predictive model is **a client-side agent approach with aggregation on the server**.

In Figure 4.1 an overview of the proposed architecture is presented: The *user interaction component* is in charge of receiving feedback information about the interaction between the user and the items. When a user u interacts with an item i, she will assign a rating $r_{ui} \in \mathcal{O}$. The set $\mathcal{O}$ is the set of possible ratings the user can assign to an item. (e.g $\mathcal{O} = \{1, 2, 3, 4, 5\}$ or $\mathcal{O} = \{+, -\}$). When a user interacts with an item, this information is given to the *user training component*. This component updates the user profile based on the item's profile and the rating given to the item by the user. After updating the user profile, this component sends back to the *item training component* on the recommendation server information from the user profile that is used to keep an up to date version of the item profile, without disclosing the action the user made on the item.

In order to bring relevant items to the user, the *user-item integration component* is in charge of actively going through the item database to offer the user items she might be interested in. This component can be installed either on the server or the client side. Finally, the *prediction component* on the client side filters out or ranks the items sent to it by the user-item integration by calculating the relevance prediction function with the local user profile and the item profile. In the next section the prediction models and algorithms will be reviewed.

4.2 Collaborative Filtering model

Following the work of Isaacman et al. [Isaacman 2011], the predictive model predicts a user rating for an item as an estimation of a probability distribution of the item ratings over the user's information using a matrix factorization technique.

Let $\tilde{\pi}^o_{ui}$ be the probability that user u will give a rating $o \in \mathcal{O}$ to item i, the goal of the system is to estimate each of the coordinates of each of the matrices

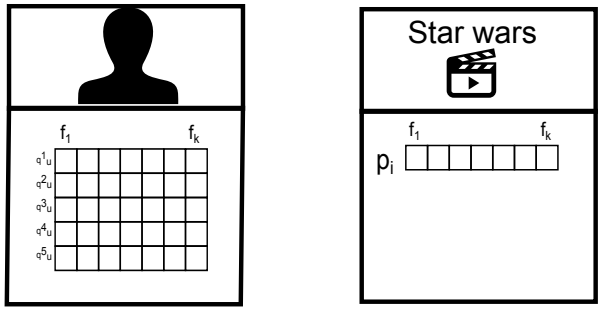


Figure 4.2: Probability user and item profile in CF system

$\tilde{\prod}^{o} = [\tilde{\pi}^{o}_{ui}]$. Assuming these matrices are low-rank, they can be reconstructed from the multiplication of two lower rank matrices of rank k: $\tilde{Q}^{o}$ of size $m \times k$ and $\tilde{P}$ of size $n \times k$ where m is the number of users of the system and n the number of items in the system.

In order to make an approximation of the ideal matrices $\tilde{Q}^{o}$ and $\tilde{P}$, a latent profile structure is defined to describe both items and users as seen in Figure 4.2. For items, a vector p_i that approximates the i-th row of matrix $\tilde{P}$ is defined. Analogously, a vector q^{o} that approximates the u-th row of each matrix $\tilde{Q}^{o}$ for $o \in \mathcal{O}$ is defined as a representation of each user.

Each p_i vector represents a probability distribution of items across the latent factors, therefore is restricted to $p_{i,k} \geq 0$ and $\sum_{k \in K} p_{i,k} = 1$, and each of the $|\mathcal{O}|$ vectors of the user represent a probability distribution of the preferences of the user across the latent factors, and is as well restricted to $q^{o}_{u,k} \geq 0$ and $\sum_{o \in \mathcal{O}} q^{o}_{u,k} = 1$.

Given these definitions , the estimation of the probability that user u has given rating o to item i is:

$$\pi^{o}_{ui} = \sum_{k \in K} q^{o}_{u,k} \times p_{i,k} = \langle p_i, q^{o}_{u} \rangle \tag{4.1}$$

In order to maintain user privacy, the matrix of item profiles P is kept by the recommendation server and the matrix Q is distributed among the users since each user has her own user profile. For the rating prediction task, the predicted rating is calculated using the local profile information and the public item profile as:

$$\hat{r}_{ui} = \sum_{o \in \mathbb{O}} \pi^{o}_{ui} \times o = \sum_{o \in \mathbb{O}} \langle p_i, q^{o}_{u} \rangle \times o \tag{4.2}$$

4.2.1 Training and prediction on the online learning framework

Recalling from Chapter 2, the objective of a learning task is to find the adequate parameters that reduce the *expected risk function* (Equation 2.22). Since the distribution of the true user's choices is unknown, an approximation of the expectation of the error is calculated as the average loss across the known information about users. Let the result of a relevance prediction parametrized by Θ be $\hat{r}_{ui}$, ℓ a function that scores the prediction against the real value of the user-item interaction r_{ui} and

$L = |r_{ui} \neq null|$ be the number of known user-item interactions, the *empirical risk function* is defined as follows:

$$\hat{e}\,(\Theta) = \frac{1}{L} \sum_{r_{ui} \neq null} \ell\,(\hat{r}_{ui}, r_{ui}) \tag{4.3}$$

The batch gradient descent algorithm minimizes the empirical risk by updating the parameters in the opposite direction of the gradient of the average loss function over all the known user-item interactions. After going through all the data (completing an iteration), parameters Θ are updated as follows:

$$\theta_t \leftarrow \theta_{t-1} - \gamma \frac{1}{L} \sum_{r_{ui} \neq null} \nabla\theta\ell\,(\hat{r}_{ui}, r_{ui}, \lambda) \tag{4.4}$$

The batch gradient descent is an *offline* learning algorithm since an intensive computation process is needed to calculate the loss for each one of the elements on the user-item interaction log for the iterations needed until the convergence of the algorithm. Other offline learning algorithms, (as seen in Chapter 2 Section 2.2.2), don't update the parameters based on the whole user-item interaction average losses but rather approximate it by the instant gradient of a sample user-item interaction picked at random, repeating the process as many times as needed until convergence (Algorithm 1).

Following the stochastic gradient descent approach, in the *online* learning model a direct update of the parameters of the prediction model is done continuously as new user-item interaction instances arrive to the system instead of drawing them from the user-item interaction log [Bottou 1998]. A recommender system with an online learning model is updated one instance at a time as follows:

1. The user interacts with an item, creating an user-item interaction (u, i).
2. Based on the profile of the item p_i and the profiles of the user, the user-agent predicts a rating for this interaction $\hat{r}_{ui} = \sum_{o \in \mathcal{O}} \langle p_i, q_u^o \rangle \times o$.
3. The recommender system learns the true rating for this interaction r_{ui}.
4. User and item update their local profile representations, based on the squared loss function over the predicted value vs the real one $\ell(r_{ui}, \hat{r}_{ui})$

Placing the recommendation problem in an online setting brings advantages in terms of the desirable objectives of scalability and privacy of the system. First of all, the prediction rule adapts intermediately the model parameters after seeing an example from reality without going through a computational intensive training process; and since user-item interaction doesn't have to saved to a log, the risks of a privacy breach are reduced.

The formal objective of the system is to minimize the error between the predictions $\prod^o$ and the ideal matrix $\tilde{\prod}^o$:

$$\min_{p_\star, q_\star} \sum_{u \in U, i \in I, o \in \mathcal{O}} (\langle \tilde{p}_i, \tilde{q}_u^o \rangle - \langle p_i, q_u^o \rangle)^2 \tag{4.5}$$

Subjected to the following restrictions:

$$D_{item}(p_i): \quad p_{i,k} \geq 0 \wedge \textstyle\sum_{k \in K} p_{i,k} = 1 \tag{4.6}$$

$$D_{user}(q_u): \quad q_{u,k}^o \geq 0 \wedge \textstyle\sum_{o \in \mathcal{O}} q_{u,k}^o = 1 \tag{4.7}$$

Applying the instant gradient descent rule, update rules for each parameter are defined as follows:

When a user assigns a rating for an item r_{ui}, the item profile of the item is available to her and adjusts the weights of the q^o profile vectors as follows:

$$\begin{aligned} q_u^o &\leftarrow q_u^o + \gamma_{t_u}(\mathbb{1}_{r_{ui}=o} - (\langle p_i, q_u^o \rangle))p_i \\ q_u &\leftarrow \textstyle\prod_{D_{user}}(q_u) \end{aligned} \tag{4.8}$$

Where γ_t is a function that calculates the learning rate. In order to achieve convergence on the online setting, the learning rate has to satisfy the following properties: $\gamma_t \geq 0$, $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$ and $\sum_{t=1}^{\infty} \gamma_t = \infty$ [Bottou 1998]. The function $\gamma_t = \gamma_0(1 + \alpha\gamma_0 t)^{-c}$ [Xu 2011] was used.

After applying the gradient step, $\prod_{D_{user}}$ projects the rows of q_u into a probability distribution defined by the restriction D_{user}.

Differing from [Isaacman 2011], in order to update the item profile p_i, the q_u profile and the rating r_{ui} the user assigned are not reported back to the centralized entity that updates the item profile since this would violate the purpose of decentralization. Instead the only information sent back to the recommendation server is the vector q_u^o where $r_{ui} = o$, the update is as follows:

$$\begin{aligned} p_i &\leftarrow p_i + \gamma_{t_i}(1 - (\langle p_i, q_u^o \rangle))q_u^o \\ p_i &\leftarrow \textstyle\prod_{D_{item}}(p_i) \end{aligned} \tag{4.9}$$

Where t_i is the number of times the item has been rated, and $\prod_{D_{item}}$ projects p_i into a probability distribution defined by the restriction D_{item}.

The computational complexity of projecting the vectors into the probability is linear on the dimensions of the vectors and the number of ratings [Chen 2011]. The complexity of calculating the user profile projection $\prod_{D_{user}}$ is $O(f * |\mathcal{O}|)$ and for calculating the item profile projection is $O(f)$.

Finally, the rating prediction under this model is the expected value of the probability distribution of the model, calculated as follows:

$$\hat{r}_{ui} = \sum_{o \in \mathcal{O}} \langle p_i, q_u^o \rangle \times o \tag{4.10}$$

In terms of the predictive performance of the model, the proposed model adjusts the parameters based on each one of the predicted probability values $\langle p_i, q_u^o \rangle$ rather

than the final prediction given by the expected value of the probability distribution $\hat{r}_{ui}$ (Equation 4.10, the expected error of the model is reduced as well since the final prediction is the summation of the predictions for each probability. However, one key feature that is included in CF models that accounts for the item bias (b_i in Equation 2.8) cannot be included into the predictive model due to the architectural restrictions placed in order to keep the recommender system from learning the ratings of the user. Since this key element for rating prediction is missing and due to the smaller training phase, the predictive performance of the model should be worse than the explored traditional model based CF predictive models seen in Section 2.1.2.

Now that the model has been presented after taking care of the design choices contemplated in the first part of the document, its predictive performance will be evaluated by using real life published datasets used for evaluating recommender system's algorithms.

4.2.2 Model validation datasets

The model validation is done on different datasets, each partitioned into three parts: The *training set* that is used to train the model, the *cross validation set* that is used to adjust the hyper parameters of the model (e.g γ_0 and K) and the*test set* that is used to report the results.

The Movielens 10M dataset contains 10000054 ratings of 10681 movies by 71567 users. Ratings that a user can assign to an item were restricted to the set $\mathcal{O} = \{1, 2, 3, 4, 5\}$. The `ratings` file has 4 fields per line: an user id, an item id, the rating she gave to the item and a timestamp. It was divided into two different datasets first: The training set has 9301274 ratings and partial test set with 698780 ratings. The partial test file contains exactly 10 ratings per user, while the rest of the ratings of each user went to the train file. The partial test set was further divided in two equal parts randomly to generate the cross validation and test sets. The training dataset is sorted by the timestamp field in order to simulate what would happen in an online setting.

The DBbook dataset contains 75558 ratings of 6166 items by 6181 users. Ratings are restricted to the set $\mathcal{O} = \{0, 1, 2, 3, 4, 5\}$. Each user has between 5 and 25 ratings. The same partition was introduced but since the dataset doesn't have the timestamps, a randomization of the user-item interactions in the train set. Models are trained with 63479 ratings, cross validation has 6039 ratings and the test set has 6038 ratings.

Finally , in order to test the model on a large dataset, the R2-Yahoo music dataset[1] was used. This dataset contains 717872016 ratings that 1823179 users gave to 136736 items. The dataset is divided into training and testing datasets. 699640226 ratings are used for the training set where each user has at least 10 ratings, 10000000 were used as the test set and 8231790 were used for cross validation. The

[1] R2 - Yahoo! Music User Ratings of Songs with Artist, Album, and Genre Meta Information, v. 1.0, http://webscope.sandbox.yahoo.com/

Dataset	Users	Items	Sparsity
DBBook	6181	6166	0,19%
Movielens-10M	69878	10676	1,34%
R2-Yahoo music dataset	1823179	136736	0,28%

Table 4.1: Validation datasets

train set was also randomized since the dataset doesn't have the timestamps of the interactions. Table 4.1 resumes the information about the validation datasets.

4.3 Model Validation

The most similar approach to the one presented in this thesis is the one from [Isaacman 2011], the main difference of this approach and their work is the item update rule, that is performed keeping into account the loss on all the user profile vectors, **this needs the original r_ui value assigned by the item to the user**, as follows:

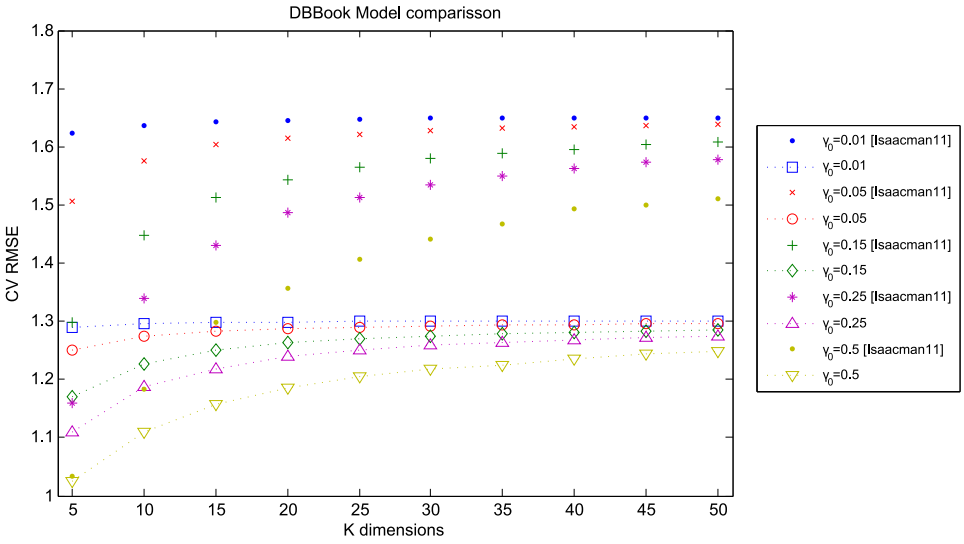
$$\begin{aligned} p_i &\leftarrow p_i + \gamma(t_i) \sum_{o \in \mathcal{O}} (\mathbb{1}_{r_{ui}=o} - (\langle p_i, q_u^o \rangle)) q_u^o \\ p_i &\leftarrow \prod_{D_{item}} (p_i) \end{aligned} \tag{4.11}$$

In order to compare the proposed model and the one from [Isaacman 2011], the predictive performance of the system is compared under different initial learning rates γ_0 and different dimension size K on the DBbook dataset (Figure 4.3a) and the Movielens-10M (Figure 4.3b) measuring the RMSE (Equation 2.16) on the cross validation set using the prediction value $\hat{r}_{ui}$ from Equation 4.10.

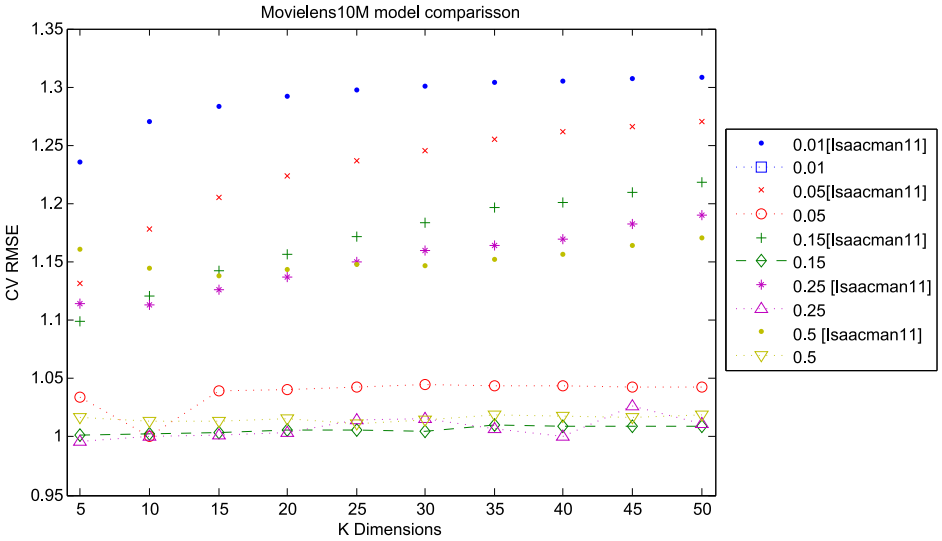
$$\mathsf{RMSE} := \sqrt{\frac{1}{|T_{ui} \neq null|} \sum_{T_{ui} \neq null} (\hat{r}_{ui} - T_{ui})^2}$$

The first observed property of the proposed update rule is that the predictive performance of the model (in unfilled markers) increases when compared to the one implemented in [Isaacman 2011] (filled markers) that shares the rating with the recommendation system. This can be explained by the increased step size that is taken when updating the item. While the weights of the losses of the original model can be averaged out causing a smaller error and thus a smaller step fixing the weights of the item profile (Equation 4.11), the proposed model (Equation 4.9) uses only the observed loss and fixes directly the weights of the item profile, causing a faster convergence.

Another observed property of the training model is that as the number of dimensions increase, the predictive performance of the model decreases, particularly for the [Isaacman 2011] model. This can be understood by looking into the tradeoff explained in Section 2.2.2: As the hypothesis size of the model increases, the number of iterations of the gradient descent algorithm over the all known data of the system



(a) DBbook



(b) Movielens

Figure 4.3: RMSE on cross validations sets across different dimensions K and γ_0 comparing performance of proposed model and [Isaacman 2011] for DBBook and Movielens

must increase as well; since only one-pass through the dataset is done, it is not enough to adequately train the extra dimensions of the model.

Finally, as the number of user-item information starts to increase in the training set, the best performing learning rate decreases. Again by checking the tradeoffs of statistical learning theory it is expected that without sufficient information the best performing algorithm tries to learn as fast as possible, setting a high learning rate. As more information becomes available, this high learning rate makes enough mistakes over the profiles making them invalid and preferable an smaller step size. This effect is observable with a small training set such as the one of the DBBook dataset, however with a medium to large dataset such as the Movielens-10M dataset the effect is not present, moreover, increasing the dimensions of the model doesn't affect as much the predictive performance of the model. In order to keep the model improving its predictive performance as new user-item information appears, a *regularization technique* for updating the item profile will be introduced in the next section.

4.4 Adding regularization to the predictive model

A predictive model *overfits* when its parameters are adjusted too closely to the trained data and stop generalizing well on new examples, a technique used to keep the model from overfiting is to minimize the regularized error defined as:

$$\min_{p_\star, q_\star} \sum_{u \in U, i \in I, o \in \mathcal{O}} (\langle \tilde{p}_i, \tilde{q}_u^o \rangle - \langle p_i, q_u^o \rangle)^2 + \lambda \left(\|p_i\|^2 + \|q_u^o\|^2 \right) \tag{4.12}$$

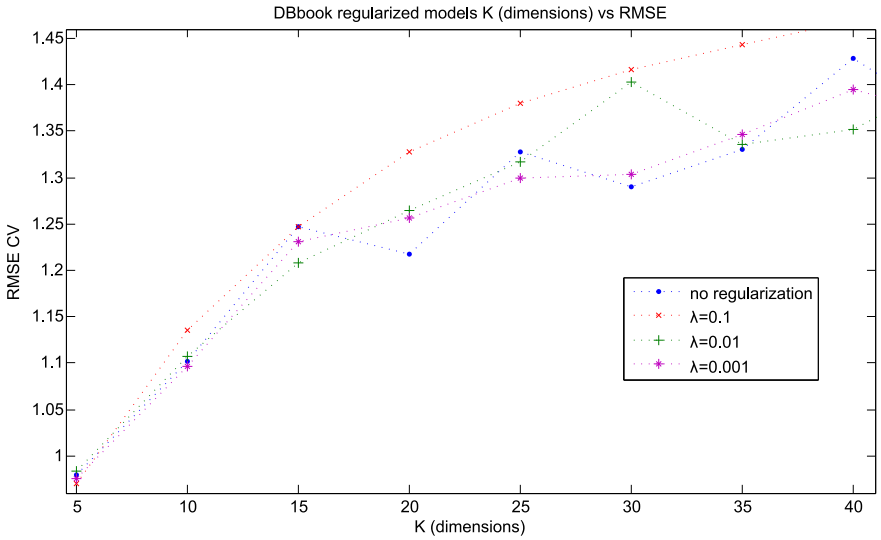
When using this minimization, the item update rule is transformed into:

$$\begin{aligned} p_i &\leftarrow p_i + \gamma_{t_i}(1 - (\langle p_i, q_u^o \rangle))q_u^o - \lambda p_i \\ p_i &\leftarrow \textstyle\prod_{D_{item}}(p_i) \end{aligned} \tag{4.13}$$

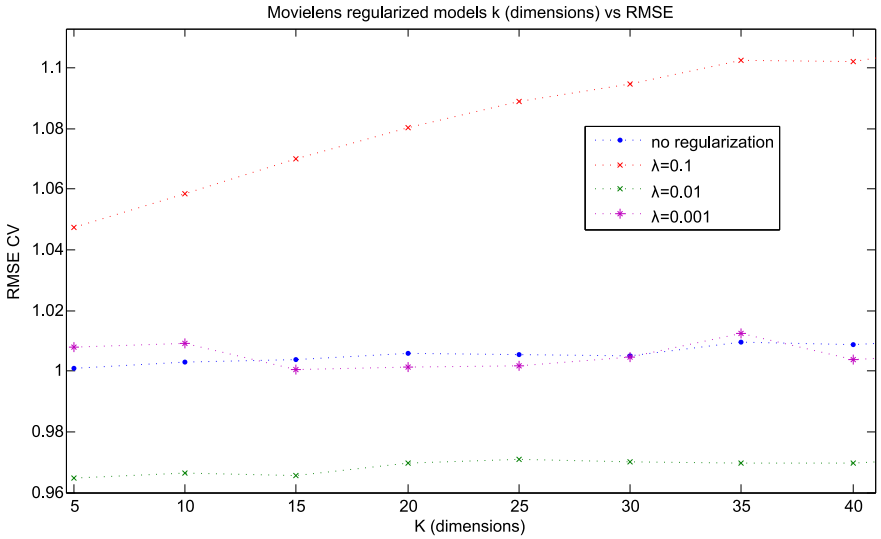
The same experiment is tested with the modified update rule for the DBBook and Movielens-10M datasets for different regularization constants (λ) using the best initial learning rate found in the previous section ($\gamma_0 = 0.5$ for the DBBook dataset, and $\gamma_0 = 0.15$ for the movielens dataset) with $k = 5$ for all models and its results are presented in Figure 4.4.

As seen by the results on both experiments of the datasets in Figure 4.4, regularization of the update rules helps the system to achieve a better predictive performance: For the DBBook dataset the RMSE on the cross validation dataset for the unregularized model was 0.97998 while the best result was 0.96959 for the regularized model with $\lambda = 0.1$ (Figure 4.4a). For the Movielens-10M dataset the RMSE of the cross validation error improves from 1.00098 for the unregularized model to 0.96463 for the regularized model with $\lambda = 0.01$ (Figure 4.4b).

Finally, results are presented on a dataset with millions of users such as the Yahoo Music dataset to illustrate the effects of regularization with a bigger dataset. In Figure 4.5 a comparison between different regularized models and the unregularized



(a) DBbook



(b) Movielens

Figure 4.4: RMSE on test sets across different dimensions K and γ_0 comparing performance of the proposed model and the improved biased model for DBBook and Movielens datasets

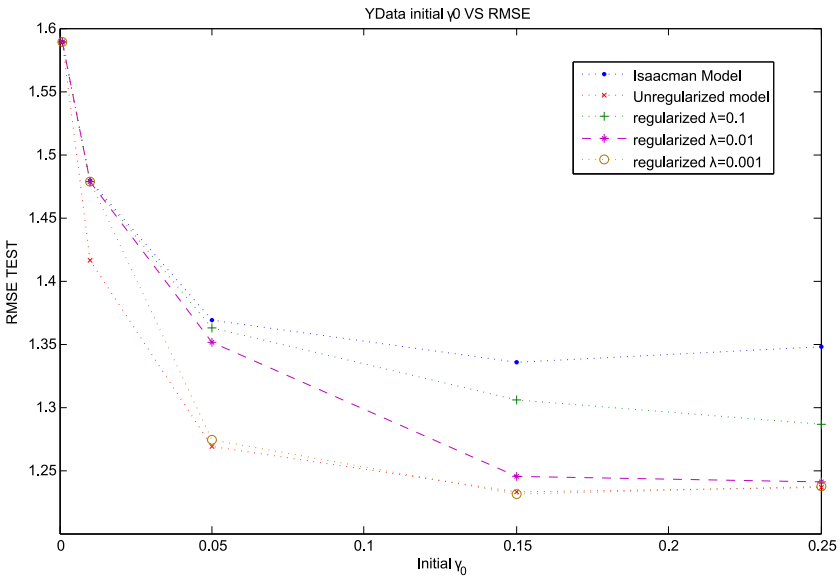


Figure 4.5: Comparisson of unregularized and regularized models for Yahoo Music dataset

one along different initial learning rates are presented with dimensionality $k = 5$. The best RMSE for the unregularized model is 1.2328 and was improved by a smaller amount using regularization to 1.2307 when setting the initial learning rate at $\gamma_0 = 0.15$ and the regularization constant to $\lambda = 0.001$.

4.5 Adding user bias to the predictive model

One known way to improve the predictive performance of rating prediction is to include into the hypothesis of the predictive model the global average, user and item bias such as the biased model seen in Equation 2.8. Due to the proposed architecture and interactions, the global average and item bias can't be calculated from the user information that is sent to the server. Therefore the only information that could be used to adapt the model is the local user bias.

In order to model the bias in terms of a probability distribution compatible with the proposed CF model, $\pi^o_{b_u}$ is defined as the bias probability that user u assigns a rating $o \in \mathcal{O}$ by modeling it as probability sampled from a *Beta distribution*. The $Beta(\alpha, \beta)$ distribution is commonly used to model the probability of a success after x successes on n trials. This distribution has two parameters: α that is related to the number of seen successes of the event and β that is related to the number of failures. The beta probability is used with bayesian inference models since its convenient for outputting a posterior distribution and compatible with the online model proposed:

Let $\pi^o_{b_u} \sim Beta\,(\alpha = 1, \beta = 1)$ be the prior probability of the rating, after seeing x successes on n trials the inferred probability distribution of the rating is

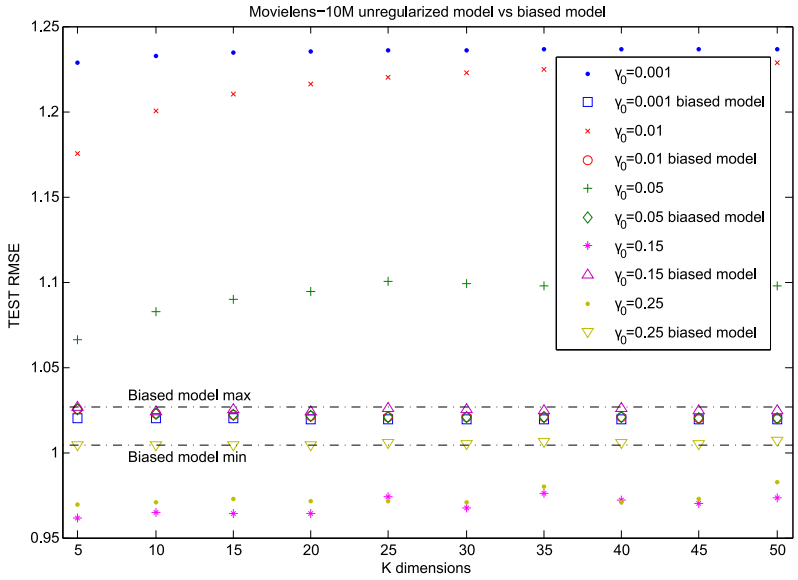


Figure 4.6: Comparisson of biased and regularized model for $\lambda = 0.1$ with Movielens-10M dataset

$Beta\,(\alpha = 1 + x, \beta = n - x + 1)$. When a user interacts with an item, she locally updates each one of the probabilities of the ratings $\pi^o_{b_u}$ by increasing the parameter α^o by one where $r_{ui} = o$ and increases the β^o parameters for the distributions where $r_{ui} \neq o$. For scalability purposes, each one of the probabilities can be calculated using the numerical mean of the distribution: $\alpha/(\alpha + \beta)$. A normalization is done to constitute a probability distribution over the user probability biases $\sum_{o \in \mathcal{O}} \pi^o_{b_u} = 1$.

To combine the output of the CF model ($\pi^o_{f_{ui}}$) and the user bias estimation ($\pi^o_{b_u}$) , a logarithmic pool [Clemen 1999] is used to generate the combined probability distribution (π^o_{ui}):

$$\pi^o_{ui} = c \times \pi^o_{f_{ui}} \times \pi^o_{b_u} \tag{4.14}$$

Where c is a normalizing constant to combine both probabilities into a distribution.

In Figure 4.6 the biased model is compared to different regularized versions of the predictive models across many initial learning rates and different dimensions K. As seen by the results, adding this signal doesn't improve the best predictive performance that can be achieved with a regularized model. The biased model seems to adjust too closely to each estimated user average as each model tends to stay in the area delimited by the minimum and maximum lines. The stagnation of the biased model around the user average doesn't allow to generalize its predictions well enough to future examples.

Dataset	Train/Predict (mins)	RMSE	Iterations
DBBook	1.13 (0,34)	0.88536	202
Movielens-10M	467.8 (64,84)	0.87028	28
R2-Yahoo music dataset	-	1.49746[2]	

Table 4.2: Running time of the training of the SVD++ model across different datasets

Dataset	Train/Predict (mins)	RMSE(Test)
DBBook	0,044 (0,012)	0.96959
Movielens-10M	1.66 (0,255)	0.96463
R2-Yahoo music dataset	245,15 (43,12)	1.23078

Table 4.3: Running time and Test error of the regularized model across different datasets

4.6 Predictive performance and scalability considerations

For each one of the seen user-item interactions, the complexity of updating the item and user profile is linear with the number of possible ratings and dimensions k in the system. **For the prediction task the complexity is $O(\mathcal{O} * f)$ and for the update task is $O(f)$ in the server and $O(\mathcal{O} * f)$ in the client**. In order to illustrate the tradeoff between scalability and predictive performance of the proposed system, it will be compared to a state of the art collaborative filtering algorithm (SVD ++, on Equation 2.10).

The testing implementation of the biased model is a multi-threaded application that goes through the dataset and simulates events as they are ordered in the dataset. The implementation is locked on a user-basis and item-basis, expected changes of order in the queue of item updates do not affect significatively the final predictive performance of the model as shown in [Isaacman 2011]. The SVD++ implementation was taken from the open source Mahout implementation[3](Equation 2.10). Both systems ran on a machine with 16GB of RAM and an Intel Core i7 with 8 processors.

Table 4.2 presents the running time of an experiment using the SVD++ model with parameters $k = 10$,$\gamma = 0.001$ and $\lambda = 0.005$. An experiment consists on training the predictive model with the training set, and predicting on the test, cross validation and test set at the end of each iteration. On the other hand Table 4.3 shows the running time and predictive performance of the biased model presented in this chapter. Since the Yahoo movie dataset is too big to fit in memory (tested on 16 GB RAM) to be processed by the single-machine implementation of Mahout, no

[2]Estimation of item average, not the SVD++ model
[3]https://mahout.apache.org/

	baseline	Model		SVD++ model	
Dataset	RMSE Item Average	Train/Predict (mins)	RMSE(Test)	Train/Predict/Iterations SVD++ (mins)	RMSE SVD++
DBBook	0.9847	0.044 (0.012)	0.96959	1.13 (0.34) 202	0.88536
Movielens-10M	0.9812	1.66 (0.255)	0.96463	467.8 (64.84) 28	0.87028
R2-Yahoo music dataset	1.4974	245.15 (43.12)	1.23078	-	-

Table 4.4: Running time and RMSE summary of the model with different datasets

results are given for the SVD++ model, in order to compare to an RMSE value, a non personalized item average prediction is considered for reference. On the other hand by handling the recommendation problem as an online learning algorithm the model is able to build a predictive model for datasets that currently are only processed by specialized cluster parallel tasks such as *MapReduce* [Schelter 2012] [Schelter 2013]. As expected, the centralized algorithm gives a better predictive performance, but the proposed biased system protects the privacy of users and scales up to millions of users while protecting their privacy by not revealing the true ratings of the user. Moreover, as the number of user-item interactions scale, the predictive model attains better performance when compared to centralized methods, as explained by the tradeoffs of large scale learning presented in Chapter 2.

4.7 Conclusions

In this chapter a client-based approach for CF recommendation under the **online learning setting**, this setting allows a system with low computational complexity operations when updating either the user or item representations at both training and prediction phases, scaling up to the number of items present in the system and the number of predictions the agent must make over time.

This comes with the cost of a significant lower predictive performance when compared to other client-based systems that either share the complete user profile releasing the local ratings to peers or a trusted entity, or by systems that protect the user's privacy with non-scalable encryption methods for the chosen architecture. However, as the number of user-item information scales, this difference decreases; this effect can be explained by the tradeoffs of large scale learning explained in Chapter 2 where as the number of user-item information scales, the hypothesis of the model can be better adjusted to generalize on future examples. The tradeoff that the proposed model assumes between its predictive performance and its scalability can be illustrated using Table 4.4. The proposed model error is comparable with memory based approaches whose RMSE is similar to the item average baseline, taking into account that the proposed model can't without considering the item average as a signal, the scalability of the proposed model allows the system to scale up to thousands of items and millions of users easily.

In the next chapter the predictive performance of the model will be further improved under certain conditions by the use of a client-side content based system that operates in parallel with the model presented in this chapter.

In terms of the exposure concerns of recommender systems presented in Chapter

3, **the client-side agent doesn't share explicitly its ratings with the aggregation server or with other peers, protecting the user's privacy**. However under a curious but honest behavior the heuristic-based strategy used by the client-side agent still discloses to the recommendation server information that can be used by an attacker to infer the ratings of the user. A system with these privacy considerations in hand will be further disused on Chapter 6.

Chapter 5

An Hybrid client-side recommender system

Contents

Motivated by improving the predictive performance of the client-based recommender system proposed in the previous chapter, this chapter explores the hybridization of the online recommender system. The system's hybrid prediction model is based on an ensemble that blends the online matrix factorization CF model and a logistic regression model trained on item metadata with a probabilistic feature inclusion strategy.

5.1 Introduction

Since content based recommender systems can run locally without compromising user privacy, an hybridization technique can be applied using the privacy-enabled recommender system explained in the last chapter and a content based system at the client-side. In order to be compatible with the scalability requirements and the chosen architecture of the system, the hybrid system must operate as well on the online learning setting.

Among the different hybridization techniques explained in Section 2.1.3, the *weighted* [Burke 2002] strategy mixes the output of different recommender systems running in parallel in order to bring one final prediction as an aggregation of single recommenders. In order to calculate the weight assigned to each predictive model, a weight based on the historical regret of the models will be applied as will be explained in Section 5.3.

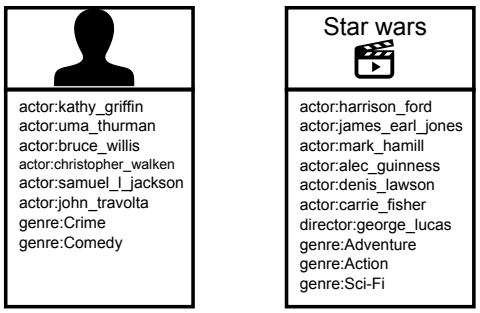


Figure 5.1: Keyword user and item profile in CF system

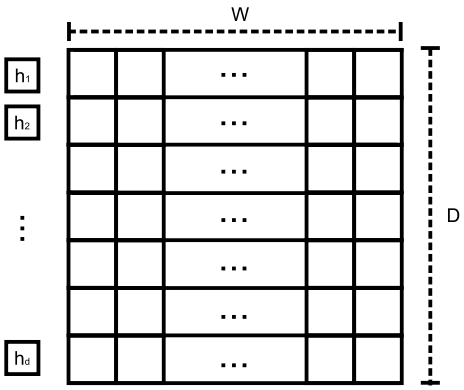


Figure 5.2: The count-min sketch structure

5.2 Content Based model

Initially, a *keyword based profile* is used to describe items and users in the system. Each item i is described by a set of concepts C_i and a user u has a profile with a list of non duplicate concepts C_u.

For example in Figure 5.1 an user and item are represented by a list of concepts. In the case of the movie domain, each item can be described by the actors, writers, directors and genres of the movie. The user is also described by a list of concepts that have frequently appeared in the history of user-item interaction. In order to represent the affinity the user has for each concept in her list, a set of $|\mathcal{O}|$ vectors $w^o \in \mathbb{R}^{|C_u|}, o \in \mathcal{O}$ is kept as the user profile as well.

As each user interacts with the items present in the system, each one of the concepts that are in the item profile (C_i) are considered for addition or deletion from the user's list C_u. Based on the work developed in [McMahan 2013], all concepts seen by the user at least N times are present in the user's list, and the size of the vectors w^o is updated. Since **keeping a list of all concepts the user has interacted in the past and how many times they have appeared in the past is not scalable** as the number of user-item interactions scale, a data structure to keep an approximate count on how many times the user has seen a concept is adopted.

The *sliding window min-count sketch structure* [Dimitropoulos 2008] is a structure that keeps a queue of *min-count sketches* [Cormode 2005]. The min-count sketch (Figure 5.2) is a bi-dimensional array T of a fixed-size (D, W). Each position or *bucket* of the array represents a counter and is initially set to 0. Additionally for each row $d \in D$ there is an independent hash function h_d that maps a concept into a column $w \in W$. When a user interacts with an item, each one of the concepts on the item's profile is hashed through each one of the h_d hash functions. The result of the hash function gives a column value for the bucket that must be incremented as explained in Algorithm 2.

Algorithm 2: Algorithm to update the count-min sketch structure

```
Data: CountMinSketch (T(D,W), h_*), C_i
Result: T(D,W)
foreach c ∈ C_i do
    foreach d ∈ D do
        hashResult ← h_d(c);
        T[d][hashResult] ← T[d][hashResult] + 1;
    end
end
return T(d,w)
```

The estimation of the number of times a concept has appeared in a user history can be estimated as the minimum across the buckets indexed by the hash of each row in the structure:

$$countSketch(c) = \min_{d \in D} T[d][h_d(c)] \tag{5.1}$$

The min-count sketch structure gives formal guarantees about the probability and the accuracy of the count estimation of the sketch. Increasing the width of the structure reduces the number of expected collisions of the hash functions, therefore the width is related with the error rate of the sketch: For an error rate of $\mathcal{E}$, the width that must be chosen is $W = \frac{e}{\mathcal{E}}$. Likewise, increasing the number of hash functions reduces the probability that two hashes index the same bucket for the same concept, therefore to achieve a probability of failure of δ a depth of $D = \lceil \ln\left(\frac{1}{\delta}\right) \rceil$ must be chosen. For example for an error rate of $\mathcal{E} = 0.1$ and a probability failure of $\delta = 0.1$, $W = 28$ and $D = 3$ can be set.

The sliding window sketch structure keeps an estimation of the latest L elements presented to the sketch, known as the window length. This window is divided into M segments, and for each segment the sliding window sketch structure keeps a queue of M identical min-count sketch structures (same dimensions and same hash functions). When an element is presented to the sliding window sketch, the sliding window sketch updates only the sketch at the head of the queue, but after $\lfloor L/M \rfloor$ updates the sketch pops the oldest sketch and pushes a new empty min-count sketch into the queue. The estimation of the number of times a concept has appeared in a user history in the sliding window min-count sketch is the sum of the estimated counts of the sketches that compose the structure, that gives the estimation of the

count during the observed window.

$$totalCount(c) = \sum_{m \in M} countSketch_m(c) \tag{5.2}$$

Eliminating old bucket values is beneficial for two purposes: (1) gradually forget older views in order to avoid the saturation of the sketch, which causes bigger estimation errors as the number of elements processed by the sketch increases, and (2) to be able to account for the *interest drift* in users. Users change their tastes over periods of time, therefore by giving more importance to recent concepts in the user profile the predictive performance of the system is expected to increase [Koychev 2000].

Once the list of concepts C_u and the w^o vectors length are updated when a user-item interaction is registered, the weights of the vector are adjusted using an online logistic regression strategy. Let $r_{ui} \in \mathcal{O}$ be the rating user u gives to item i, t_u be the number of items the user has rated and $m_{ui}(C_i \times C_u) \rightarrow \mathbb{R}^{|C_u|}$ a function that takes the concept set of an item and converts it into a binary vector where each coordinate is 1 if the user's concept belong to the items list ($m_{ui}[f] = \mathbb{1}_{C_u[f] \in C_i}$). For each vector the prediction w^o, $\sigma(\langle w^o, m_{ui} \rangle)$ is calculated and each vectors is updated as follows:

$$w_u^o \leftarrow w_u^o - \gamma(t_u)(\sigma(\langle w^o, m_{ui} \rangle) - \mathbb{1}_{r_{ui}=o}) m_{ui} \tag{5.3}$$

Where $\langle w^o, m_{ui} \rangle$ is the dot product between vectors w^o and m_{ui}, $\sigma(c) = 1/(1 + \exp(-c))$ is the sigmoid function and $\gamma(t)$ is a function of the learning rate that decreases as the number of trainings of the user increases, e.g $\gamma_t = \gamma_0(1 + \alpha\gamma_0 t)^{-c}$ [Xu 2011].

The rating prediction under this model is calculated as follows:

$$\hat{r}_{ui} = \frac{\sum_{o \in \mathcal{O}} \sigma(\langle w^o, m_{ui} \rangle) \times o}{\sum_{o \in \mathcal{O}} \sigma(\langle w^o, m_{ui} \rangle)} \tag{5.4}$$

The CB algorithm can be divided into three steps in order to analyze its computational complexity. The first step is to present to the sketch the concepts related to the item, this has a complexity linear on the number of concepts the item has $O(D \times |C_i|)$. Once all the concepts are presented, the second step is to add or remove the concepts from the user list based on the sketch estimated count, this has a complexity of $O(D \times |C_i \bigcup C_u|)$. Finally, the third step is to update each one of the vectors of the user: the creation of the binary vector m is linear with the number of items left in the user's concept list and both the final update and prediction are $O(\mathcal{O} * |C_u|)$. By this analysis is beneficial to use the sketch and the threshold criterion to keep a manageable size of the concept list of each user since the scalability of the CB system is related to its size.

In order to validate the model, the Movielens dataset is used for training and testing the model as described in Section 4.2.2. In order to define the set of concepts C_i that describe each item the mapping information released in the 2001 HetRec

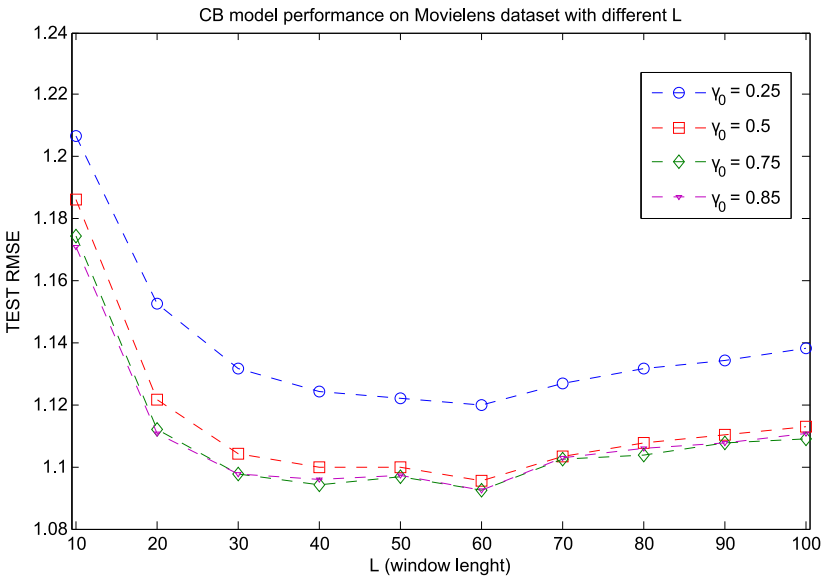


Figure 5.3: CB filtering tested on the Movielens 10M dataset

workshop [Cantador 2011] is used. This mapping uses the dataset information from the IMDb website[1] and the Rotten Tomatoes website[2] to describe each movie present in the Movielens dataset. The following concepts were used to describe a movie: actors, directors, writers and genres. The feature space size is 131407 concepts.

Parameters for each user sketch are configured as follows: $W = 450$ and $D = 3$ for an accuracy of $\varepsilon = 0.006$ with probability of $0.9(\delta = 0.1)$. The list C_u of concepts is estimated by the sliding window sketch, where the estimated count has to be at least $N = 5$. CB models with different initial learning rates and window lengths L are plotted against the error and presented in Figure 5.3. The figure shows that keeping an sliding window sketch to calculate the concepts each user has is beneficial for the predictive performance of the model: A small window size is not enough for the model to have enough features to learn, on the other hand when keeping a window too large the model includes more features and the size of the user-item information is not enough to adjust the weights of these extra features properly. Therefore, having an sliding window sketch for each user is both beneficial for the scalability and the predictive performance of the CB model.

5.3 Hybrid Model

As seen in Chapter 2, single paradigms for recommendation have their own problems: CB approaches are known to be vulnerable to the overspecialization problem since they only can detect the relevance of items that are similar to the ones the user has seen before, on the other hand CF approaches are known to be vulnerable to

[1] http://www.imdb.com
[2] http://www.rottentomatoes.com

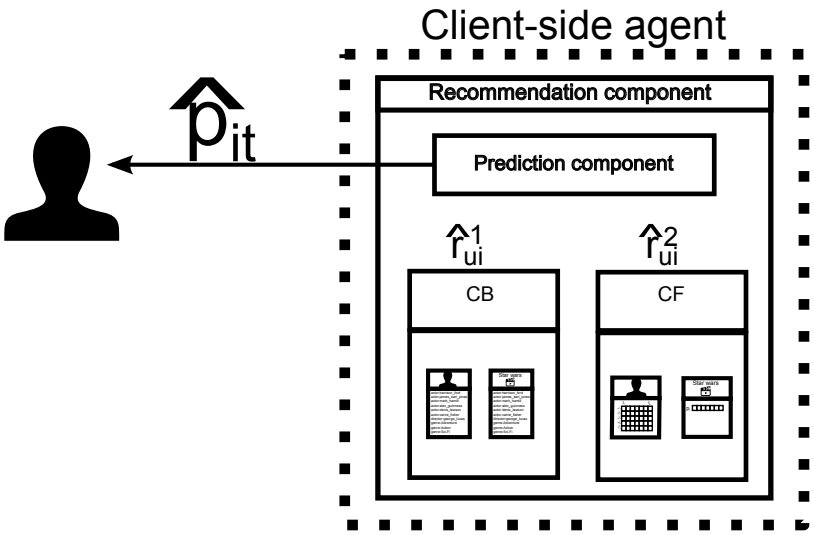


Figure 5.4: The hybrid client-side model

sparsity and cold-start problems. By placing the system under the framework of prediction with expert advice [Bianchi 2006], the final relevance prediction of the client-side recommendation agent is calculated as a *exponentially weighted average forecaster* of two experts at the client-side: The CB model and the CF model (Figure 5.4), the weighted approach generates a final relevance prediction based on the outputs of both models.

In this section the u from the notation is dropped for clarity. Let $\hat{p}_{i,t}$ be the final prediction of the forecaster for item i at turn t after taking into account the predictions of the experts, $\mathcal{E} = \{1, 2\}$ is set of expert indexes, $\hat{r}^E_{i,t}$ is the prediction of expert E at time t for item i and $\ell(\mathbb{R} \times \mathcal{O}) \to \mathbb{R}$ is a non-negative loss function that scores a prediction (either from the final forecaster or from an expert) against the true rating that the user gave to the item.

In the prediction with expert advice model, when a user rates an item i at time t, the item profile of i is presented to the experts and they make a prediction $\hat{r}^E_{i,t}$. The final forecaster accesses these predictions and makes a final prediction $\hat{p}_{i,t}$, the real rating of the item $r_{i,t}$ is revealed to the experts and each one incurs on a loss $\ell(\hat{r}^E_{i,t}, r_{i,t})$. The forecaster incurs on a loss $\ell(\hat{p}_{i,t}, r_{i,t})$.

The *cumulative regret* is defined as the difference between the cumulative losses of the final predictor and an expert. The regret of the forecaster with respect with expert E after n trains is defined as:

$$R_{E,n} = \sum_{t=1}^{n} \left(\ell(\hat{p}_{i,t}, r_{i,t}) - \ell(\hat{r}^E_{i,t}, r_{i,t}) \right) \tag{5.5}$$

Each expert prediction has a weight that is used by the forecaster to compute its prediction, the expert's weight is computed as follows:

$$W_{E,t-1} = \frac{\exp\left(\eta_t R_{E,t-1}\right)}{\sum_{e \in \mathcal{E}} \exp\left(\eta_t R_{e,t-1}\right)} \tag{5.6}$$

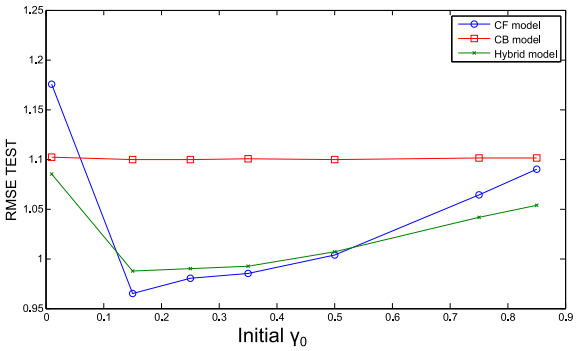


Figure 5.5: Hybrid filtering tested on the Movielens 10M dataset

The forecaster prediction after turn t is:

$$\hat{p}_{i,t} = \frac{\sum_{E \in \mathcal{E}} W_{E,t-1} \hat{r}^E_{i,t}}{\sum_{E \in \mathcal{E}} W_{E,t-1}} \tag{5.7}$$

The extra complexity of using this framework is linear on the number of experts used for the recommendation.

In order to validate the model, the Movielens-10M dataset is used for training and testing the model as described in Section 4.2.2. In order to compare the models explained so far a CB, CF and the hybrid system were trained with various γ_0. The CF model had a dimensionality of $k = 5$ and the CB model was limited to a window size of $l = 60$. For the hybrid model the CB model was fixed to a model of initial learning rate $\gamma_0 = 0.75$ and the initial learning rate of the CF model was changed under the same parameters of the single CB and CF models. Results in Figure 5.5 show that the hybrid model only beats the regularized CF model when the parameters of the CF model are not tuned. Although not beating the model, the regret of the final model is linear to the one of the best predictive expert that composes the model. In Table 5.1 results from single models are compared with the hybrid strategy, for this validation experiment both models were trained using the same initial learning rate in order to show what would happen if a under performing model is combined with a better tuned one. The table allows to appreciate more clearly the impact that the prediction of the single model has on the final output prediction of the model.

5.4 Predicting under the cold-start scenario (new item problem)

One reason because the CF model outperforms the proposed hybrid is because in its validation no *cold-start* situation is present. Recapitulating from Chapter 2, a shortcoming from CF models is that they are unable to make relevant predictions for new items since no opinions are known for the item, this scenario is known as a cold-start problem. In order to show what would happen in a more realistic scenario

$\gamma_0(CFandCB)$	k	l	RMSE TEST CF	RMSE TEST CB	RMSE Hybrid
0.01	5	60	1.14824429	1.26665657	1.1720365
0.15	5	60	1.0031442	1.1484055	1.00874913
0.25	5	60	0.99940185	1.12122614	0.9946159
0.35	5	60	0.99576192	1.10876316	0.99095252
0.5	5	60	1.01698005	1.09913029	1.01353192
0.75	5	60	1.06807696	1.09600724	1.04312786
0.85	5	60	1.09030993	1.09785004	1.05429579

Table 5.1: Results hybrid recommender with Movielens dataset

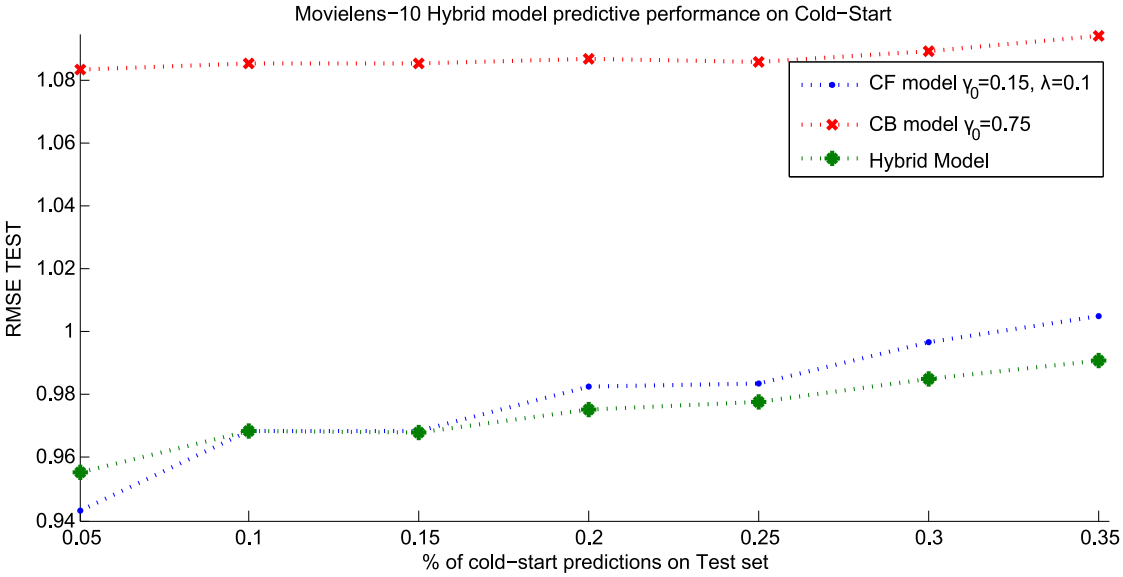


Figure 5.6: Hybrid filtering on Cold-Start scenario tested on the Movielens 10M dataset

when items arrive to the client-side agent and no information on this item are known, a partition of the whole dataset is made in order to force the test and cross validation datasets to contain ratings from items not present in the training set. An experiment was designed in which the test set was forced to include items not seen before in the train set, at increasing percentages. In Figure 5.6 the results for the Hybrid, CF and CB models are shown as the percentage of ratings with a cold-start situation is forced into the test set. As observed as the number of predictions on a cold-start situation increases, the pure CF model predictive performance deteriorates while the pure CB model is not affected, finally after 20% of the predictions being from a cold-start situation the hybrid model outperforms the CF and CB models that compose the system.

5.5 Conclusions

As a motivation to improve the predictive accuracy of the privacy-enabled recommender system, this chapter introduced a content-based filtering model that uses a keyword based approach to represent both users and items. Following the CF model presented in the previous chapter, a vector per possible rating of size $|C_u|$ is defined and trained in a online logistic regression as the user interacts with items. Since the number of concepts an item has can be considerable and the number of items an user has seen can scale, a sliding window sketch structure is used to keep an estimation of the number of times a user has interacted with an item and a threshold based criterion is used to keep into the user's concepts list the concepts seen by the user at least N times during the duration of the window. This strategy was shown to be beneficial in terms of the scalability and the predictive performance of the system.

Next, a client-based approach for recommendation using a model based hybrid approach that mixes the predictions of the collaborative model presented in Chapter 4 and the keyword-based CB model presented in Section 5.2. The online model mixes two low computational complexity models at both training and prediction phases, scaling up to the number of items present in the system and the number of predictions the agent must make over time. The expert weighting framework allows an online personalized balancing of each one of the predictions, therefore if a model is under performing the final predictive performance of the model is not affected significatively. Moreover, on cold-start situations the hybrid model helps the system to attain a better predictive performance.

On the privacy concern, although the decentralized model presented here doesn't share the whole profile of the user with the recommendation server, under a curious but honest behavior the proposed system still leaks enough information so the recommendation server can estimate the distribution of ratings for each user. In the next chapter tools for offering further privacy guarantees to the users of the system will be explained and the hybrid model presented in this section will be useful to keep the predictive performance of the system.

Chapter 6

Privacy considerations and their impact on the predictive accuracy of the system

Contents

As [Kobsa 2007] identified, in order to use collaborative filtering on client-side agents a transmission of user profile information from and to the user is needed, however if done without precautions a curious server or peer might learn information from the user profile and cancel out the benefits that client-side architectures bring to user privacy.

The decentralized model presented in Chapter 4 doesn't share the ratings the user has assigned to items, neither shares the whole profile with the recommendation server. However, under a curious but honest behavior the proposed system still lets know the recommendation server which items the user has interacted. Moreover, by continously observing the profiles sent by the users, the system could potentially identify the evolution of the user profiles sent and infer if a user is assigning the same rating to different elements. In this chapter two solutions to these problems are presented: The use of a random perturbation approach to limit the inferences an attacker can make when observing the dense probability vectors the client-based approach sends to the server in the probability CF model (Section 6.1), and a filtering criteria based on keywords in order to automatically limit the reporting of interactions to the server in case the user doesn't want to report the interaction.

6.1 Perturbation of the user profile

Continous observation of the vectors that the user sends to the recommender system can configure an attack that allows the system to learn the ratings the user has assigned to the items.

The user profile update rule formulated in Chapter 4 (Equation 4.8) is formulated as follows:

$$q_u^o \leftarrow q_u^o + \gamma_{t_u}(\mathbb{1}_{r_{ui}=o} - (\langle p_i, q_u^o \rangle))p_i$$
$$q_u \leftarrow \prod_{D_{user}}(q_u)$$

The recommendation server can simulate the next state of the user profile vector since all the parameters are public: The recommendation server knows the number of times the server has interacted with the user allowing the recommender system to calculate γ_u at interaction t and the item profile p_i is public. When a vector q_u^o arrives to the recommender server, the system can reproduce the update that the vector will be subjected to in the client-side agent, so if the updated vector is seen again in the interaction stream the server is able to count the frequency of appeareance of each vector, configuring a *frequency analysis attack*.

The attack is configured as shown in Algorithm 3, the attacker keeps a map of lists of the vectors it has seen before by each user (*knownVectorMap*) and the times it has seen each vector (*frequencyMap*). When a vector arrives the attacker compares it to the known vectors and if found increases the frequency count, adds it to the corresponding user list and then simulates the update that the client will do to its vector.

Algorithm 3: Frequency-analysis vector attack

Data: Stream of tuples (u, q_u^o, p_i)
Result: frequencyMap(u, index, count), knownVectorMap(u,*User vectors*)
frequencyMap $\leftarrow$ <>;
knownVectorMap $\leftarrow$ <>;
foreach $c \in Stream$ **do**
  if knownVectorMap.get($c.u$).contains(q_u^o) **then**
    index $\leftarrow$ knownVectorMap.get($c.u$).getIndex(q_u^o);
  end
  else
    index $\leftarrow$ knownVectorMap.get($c.u$).size()+1;
  end
  count $\leftarrow$ frequencyMap.get($c.u$, index);
  frequencyMap.put($c.u$,index,count + 1);
  oldUserVectors $\leftarrow$ knownVectorMap.get($c.u$);
  knownVectorMap.get($c.u$).put(simulateUpdate(q_u^o, p_i, oldUserVectors));
end
return frequencyMap,knownVectorMap

The simulation of the update (Algorithm 4) takes advantage of the fact that the vectors represent a probability distribution, thus if a vector is missing the residual probability vector can be easily calculated and projected along the known vectors of

the user.

Algorithm 4: Simulate update algorithm

```
Function simulateUpdate(q_u^o, p_i, γ_{t_u}, knownUserVectorList)
Result: updatedVectorList
updatedVectorList ← [];
residualVector ← [];
if ¬ knownUserVectorList.contains(q_u^o) then
    knownUserVectorList.add(q_u^o);
end
foreach known ∈ knownUserVectorList do
    residualVector ← residualVector + (1 − known);
end
foreach known ∈ knownUserVectorList do
    if known = q_u^o then
        trueValue ← 1;
    end
    else
        trueValue ← 0;
    end
    updatedVectorList.add(known + γ_{t_u}(trueValue − (⟨p_i, known⟩))p_i);
end
∏_{D_user}(updatedVectorList ∪ residualVector);
return updatedVectorList
```

Once the frequency of each vector is calculated and by using as background information the rating distribution of the users, the mapping between the frequency of the vector and its real value in the user profile can be estimated. The performance of a simple attack that assigns a rating value for each vector in the order of the known distribution of ratings on the Movielens-10M dataset is shown in Table 6.1 with users with at least min interactions ratings in the dataset. As seen in the table, as the user reveals more ratings, the attack on user privacy becomes more effective.

Min interactions	MAE
10	0,81932808
20	0,81821449
30	0,81636623
40	0,81422623
50	0,8120607
60	0,80986958

Table 6.1: Frequency-analysis attack results

A solution to avoid the attack described beforehand is to introduce a random perturbation into the reported vector in order to make it difficult for the recommendation server to discern if it has seen the vector beforehand. In order to understand how to efficiently mask the vector reported to the recommender server, notions of

differential privacy are used.

Differential privacy [Dwork 2006] is privacy preservation technique that was initially created to protect user privacy when making available sensitive information form a user database, for example user purchase records, web search records or medical records. Differential privacy was created to create algorithms that publish an outcome based on the private data that make difficult for an attacker to relate the published information with other background information about a user. Let D_1 be the database that has all users and D_2 the same database without one user, differential privacy offers a guarantee that the output of an algorithm $\mathcal{M}$ that takes as input a database will be bounded by a difference of ε with probability $1 - \delta$.

More formally, a randomized algorithm $\mathcal{M}$ that has an output in the set S ($S \subseteq range(\mathcal{M})$) satisfies (ε, δ) *differential privacy* if for two adyacent databases D_1 and D_2 (that differ only in one record), the probability of the outcome of the algorithm calculated on both datasets is bounded by:

$$Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \times Pr[\mathcal{M}(D_2) \in S]] + \delta \tag{6.1}$$

One way in which the output of an algorithm can be randomized to account for differential privacy is to add noise to the output of the algorithm [Dwork 2008], let $\mathcal{M}(X)$ be a function $f(X) : D^n \rightarrow \mathbb{R}^d$, the *sensitivity* of a function f is the maximum diference on the output of the function f operating on adyacent databases:

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1 \tag{6.2}$$

Noise from a *laplace distribution* (also known as exponential distribution) can be used to create a randomization that implements differential privacy. The laplace distribution's probability distribution function is $P(x|b) = \frac{1}{2b} \exp(-|x|/b)$ with variance $2b^2$ (centered at 0). An algorithm that adds noise to the output of f sampled from a laplace distribution with parameter $b = \Delta f / \varepsilon$ enjoys ε-differential privacy [Dwork 2008].

Differential privacy relies on a curator that releases aggregate information using differential privacy functions that take as input private information from different users, it is not designed to protect individual privacy. Due to its inherent architecture it was not used directly as presented in known literature since reintroducing a curator as a trusted peer can re-introduce the exposure risks presented on centralized entities and the scalability problems that sharing the encrpted profile present as seen in Chapter 3 Section 3.5. Instead, a random perturbation technique is used in order to restrain the attacker from identifiying similar vectors from the same user, therefore the noise added should be proportional to the sensitivity of the *dot product* function since it is a function that represents the similarity between two vectors.

The interaction between the client and the server proposed in Section 4.2 is changed. Noise from a laplace distribution is added to each of the coordinates of the vector q_u^o where $r_{ui} = o$ to build the reported the recommendation server q_u^{Noise} as follows:

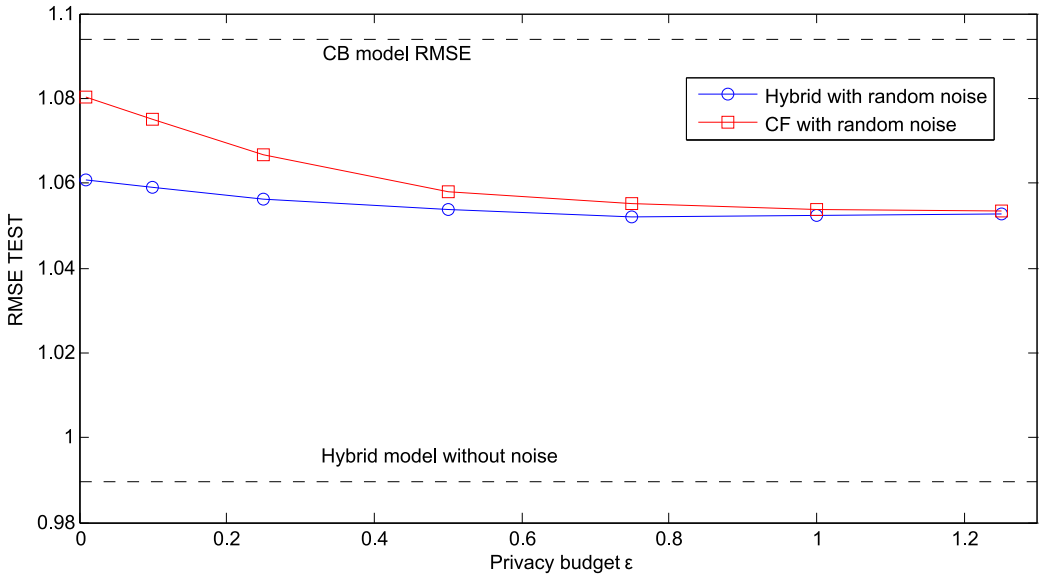


Figure 6.1: Noise on CF and Hybrid models tested on Movielens 10M dataset

$$q_u^{Noise} = q_u^o + Laplace\left(\frac{1}{\varepsilon}\right) \tag{6.3}$$

Since there is noise on the information used by the recommender system to adjust the item profiles, the predictive performance of the recommender system is expected to drop. In order to verify how setting a privacy budget ε affects the overall predictive performance of the hybrid model (Chapter 5) when reporting a noisy user profile, an hybrid model with a CF model with parameters $\gamma_0 = 0.15$ and $k = 5$ and a CB model with parameters $\gamma_0 = 0.75$ and $l = 60$ was trained with different ε privacy budgets as seen in Figure 6.1 using the Movielens-10M dataset for training and testing the model as described in Section 4.2.2.

First, by observing the results of the CF model it is clear that the privacy constraints affect the predicive performance of the model. As the privacy budget increases, the ammount of noise required to mask the reported user profile q_u^{Noise} decreases and the predictive performance of the model improves.

On the other hand, the predictive performance of the hybrid model is not as affected with the noise as the CF model since the CB model helps to mitigate the effects of the noise in the recommender system. **The use of the hybrid model is benefitial in order to improve the predictive performance of the CF model with noise**, although the predictive performance of the hybrid system is noticeable worse when compared to the performance of the hybrid model without noise.

6.2 Keyword-based filtering

As recognized by different authors [Castagnos 2007] [Draidi 2011] [Bilenko 2011], one of the advantages of client-side agents in terms of privacy is that the agents can reveal only certain interactions they have with the items. An advantage of the proposed model is that the user can completely skip the reporting of the vector q_u^{Noise} if she deems the interaction prejudicial for its privacy. A way to facilitate this omission is to create keyword-based blacklists of items.

In Table 6.2 all the genres of the movies in the Movielens-10M are presented with the number of movies marked with that genre. In order to simulate what would happen if users refused to report ratings on one category, all users were blacklisted with the genre *Horror*, which leaves out reporting the vector for around 10% of the items in the dataset.

Genre	count
Drama	5339
Comedy	3703
Thriller	1706
Romance	1685
Action	1473
Crime	1118
Adventure	1025
Horror	1013
Sci-Fi	754
Fantasy	543
Children	528
War	511
Mystery	509
Documentary	482
Musical	436
Animation	286
Western	275
Film-Noir	148
IMAX	29
Short	1

Table 6.2: Genres in Movielens-10M dataset and count

In order to verify how setting a privacy budget ε affects the overall predictive performance of the hybrid model presented on the previous section with blacklisted items, an hybrid model with a CF model with parameters $\gamma_0 = 0.15$ and $k = 5$ and a CB model with parameters $\gamma_0 = 0.75$ and $l = 60$ was trained with different ε privacy budgets as seen in Figure 6.2 using the Movielens-10M dataset for training and testing the model as described in Section 4.2.2.

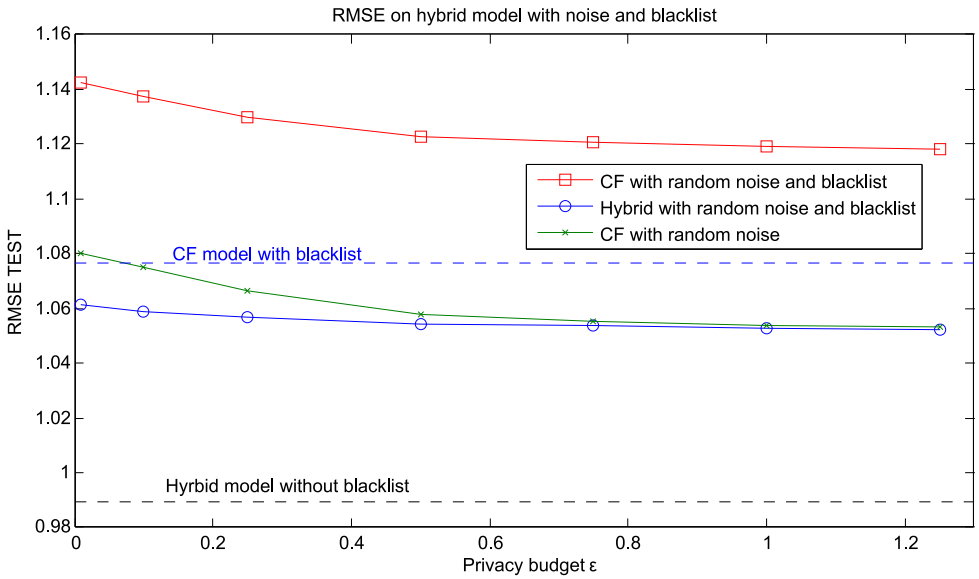


Figure 6.2: Noise and blacklist strategy on CF and Hybrid models tested on Movielens 10M dataset

The figure shows that the blacklisting of items affects the predictive performance of the CF model presented in Chapter 4. While the RMSE measured in the test set of the CF model without perturbation is 1.004964, the RMSE on the test set with only the blacklist criteria abstention is 1.07680. When operating the blacklist policy with the random noise masking presented on the previous chapter, the predicive performance of the model is furthermore affected. **However, and as shown in the previous section, the use of an hybrid model with a CB model helps to mitigate the introduction of privacy protecting policies in the CF model.**

6.3 Conclusions

In this chapter, two strategies for protecting user privacy are explained. The first strategy is a random noise perturbation that makes it difficult for the attacker to simulate the inner state of the vectors of the client. Rather than claiming complete privacy against an attack using any background information the attacker might have about the user, this work claims that correlation attacks to find out the rating the user has assigned to items are protected until a certain measure by the privacy budget parameter ε. The random noise perturbation is shown to have an impact on the predictive performance of the CF model, however when combined with a CB model that doesn't have privacy concerns, the proposed hybrid architecture mitigates the impact of using the random noise perturbation until a certain extend.

The other strategy used to protect user's privacy is to abstain from reportig the interaction with an item to the recommender system if such interaction is deemed prejudicial to the privacy of the user. This strategy also affects negatively

the predictive performance of the CF model, and again the hybrid model shows a mitigating effect on the predictive performance loss that the keyword based filtering has, even when the filtering stops from training around 10% of the available items.

Chapter 7

Conclusions

This thesis proposes a new recommendation system where privacy and scalability are major concerns. The system was designed under the premises of avoiding user exposure risks while providing high-scalability explained in Chapter 3. The first limitation that the strategy for preserving the privacy of the user imposes traditional architectures is the introduction of a client-side agent that keeps the processing and gathering of user information away from a centralized entity to avoid exposure risks. While all user information will be kept on the user profile, the items' profiles will be kept by an aggregation server that will make them public.

The proposed decentralization brings challenges in terms of the scalability of the system and in terms of the privacy of the users of the system. By avoiding centralized entities, Collaborative Filtering algorithms that scale cannot be applied since they need the whole database of users and items to adjust its parameters efficiently. On the other hand Collaborative Filtering recommender systems need information from other users to adjust the user profile, and sending information to other user's or an aggregation server without masking or protecting the information can overturn the benefits that the decentralization brings to user privacy. Strategies to send information to other peers or aggregation servers while keeping user privacy can be classified as : Heuristic, cryptographic and random noise perturbation strategies. Computational complexity analysis of the cryptographic approach was carried in Section 3.5 and in order to scale with the number of users and items of typical recommendation systems it was avoided and a heuristic-based strategy for profile masking was presented along with the general architecture of the system in Chapter 4.

In the proposed CF algorithm, user and item profiles are updated in a strategy based in the work of [Isaacman 2011], where an online gradient descent strategy is used to update user and item latent vectors. A variation is introduced since the original algorithm doesn't mask its information before sharing it with other peers. In order to adjust the users' and items' profiles, the proposed distributed system adjusts keeps at the local agent all user information and at each user-item interaction shares with the recommendation system a dense vector that is used for adjusting a public item profile. One major modification to the algorithm made by [Isaacman 2011] is that only one vector is revealed to the recommendation system without revealing the true rating of the user. By using this modification the recommender system doesn't learn the ratings of the user and the predictive performance is improved, particularly when using a regularization strategy.

Next, an hybridization approach was presented in Chapter 5. A keyword-based Content Based recommender system was presented where items are marked with keywords and the user is marked with her most frequently used concepts. Keeping the same strategy as the CF model, the user has as many vectors as ratings are available in the system, however the dimension of these vectors depends on the number of frequent concepts of each user. To keep a low dimensionality of the user profile, a sliding window sketch structure is used to keep an estimation of the number of times a user has interacted with an item. The presented strategy is shown to be beneficial for the predictive performance of the model, as well as the scalability since the complexity of the operations carried away by the system are linear with the number of dimensions. Finally an hybridization strategy was presented based on the historical regret of both models running in parallel. This low-complexity strategy is known to keep the regret of the hybrid model linear with its best performing expert. The model was tested and it was shown that in a cold-start situation, the hybrid model outperforms the models that compose the hybrid system. The results obtained until this chapter were presented in a publication at the UMAP conference [Moreno 2014b].

Finally, two strategies were presented to improve the privacy guarantees that the heuristic-based strategy presented in Chapter 4 offers. Random noise was generated to mask the vector that leaves the client-side agent to keep an attacker from simulating the internal state of the client-side vector. By using notions from differential privacy, each coordinate of the reported vector is affected by noise taken from a laplacian distribution. While perfect privacy against all types of background information is impossible to achieve, the noise added to the vector is parametrized by a privacy budget ε. Results show that the predictive performance of the CF model is affected as the privacy budget decreases, however this effect is mitigated by the use of the hybridization strategy. Another heuristic-based strategy used for protecting the user's privacy is to refrain to report some items to the recommender system when a user doesn't want to be linked with an item, the strategy was simulated by using a keyword based strategy where a keyword was chosen to be blacklisted by all users, comprising 10% of the items in the system. After simulating both strategies the hybrid approach is shown to mitigate the effect of the alterations to the CF model.

Closing remarks and future work

This thesis proposes two rarely considered design factors in the recommendation system's literature. First, the use of online learning models is not usually considered since the computational capacity of current cloud based solutions allow current recommendation systems to gather as much data as needed, and process it efficiently to train the predictive models. As seen in Chapter 2 only one iteration along the user-item information is not sufficient to train these models and new models should be adapted for this kind of systems. However, as shown in this work, one advantage

of this kind of learning strategy is that the complexity of the training and prediction tasks can be greatly reduced, and as the theory of statistical theory explains, by keeping a fixed small hypothesis for the model and increasing the number of user-item interactions, the predictive performance of the model can be improved. This is beneficial for applications that expect a great amount of user-item interactions since its predictive performance wont be as affected as the ones where user-item information is scarce. An interesting approach for future analysis is to analyze how does these kinds of models and training approaches scale for more complex models with bigger hypothesis, for example those that instead of adjusting their parameters for rating prediction adjust parameters for ranking prediction such as the models from [Rendle 2009] [Shi 2012].

Other rarely considered choice is the use of hybrid models at client-side. One reason for the lack of studies of hybrid models at client-side is again that bigger hypothesis models are needed to include extra features about the user and the item into the predictive model (i.e [Gantner 2010]). Since traditional architectures can solve efficiently the scalability problem, contemplating the performance of these models in a distributed environment where each agent has its own data is still an open problem. Other reason for this choice is that traditionally CF algorithms are the ones to present a challenge in privacy and scalability when operating in a distributed environment with a client-side agent, much of the research in these kind of systems has been devoted to solve these challenges as shown in Chapter 3. The simple hybridization approach shown in Chapter 5 can offer valuable insights for future online integration of more complex models.

Finally, a basic model for representing users and items was used in this work in the CB model by the use of a knowledge model based on keywords. The choice of using this knowledge model was guided by the scalability concern, however more complex knowledge models could be used such as the hierarchy-based models or ontology-based models presented in Section 2.1.1. The use of these knowledge models could further reduce the sparsity between the representations of users and items, improving the predictive performance of the system. For example, an extension to the CB model presented in this thesis was designed by our group in [Moreno 2014a] where the features of the items were assigned as cluster ids created from a clustering technique based on co-occurrence of features, as extracted from open web semantic data (DBPedia) for the DBBook dataset. The model was trained in an offline setting using a logistic regression strategy. As shown by the results for rating prediction in [Di Noia 2014], our hybridization approach based on model *switching* performed similarly to state of the art hybrid methods. Future work could be developed to adapt this strategy to the online setting for clustering as well as the model training.

Bibliography

[Adomavicius 2005] G. Adomavicius and A. Tuzhilin. *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions.* IEEE Trans. on Knowl. and Data Eng., vol. 17, no. 6, pages 734–749, June 2005. (Cited on pages 3, 10, 13 and 18.)

[Agrawal 2000] Rakesh Agrawal and Ramakrishnan Srikant. *Privacy-preserving data mining.* In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, SIGMOD '00, pages 439–450, New York, NY, USA, 2000. ACM. (Cited on pages 39 and 57.)

[Ahn 2010] Jae W. Ahn and Xavier Amatriain. *Towards Fully Distributed and Privacy-Preserving Recommendations via Expert Collaborative Filtering and RESTful Linked Data.* Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on, vol. 1, pages 66–73, 2010. (Cited on page 50.)

[Aimeur 2008] E. Aimeur, G. Brassard, J. M. Fernandez, F. S. M. Onana and Z. Rakowski. *Experimental Demonstration of a Hybrid Privacy-Preserving Recommender System.* In Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, pages 161–170. IEEE, March 2008. (Cited on pages 48 and 50.)

[Alaggan 2011] Mohammad Alaggan, Sébastien Gambs and Anne-Marie Kermarrec. *Private Similarity Computation in Distributed Systems: From Cryptography to Differential Privacy.* In Antonio Fernàndez Anta, Giuseppe Lipari and Matthieu Roy, editeurs, Principles of Distributed Systems, volume 7109 of *Lecture Notes in Computer Science*, pages 357–377. Springer Berlin Heidelberg, 2011. (Cited on pages 51, 52 and 61.)

[Ali 2004] Kamal Ali and Wijnand van Stam. *TiVo: making show recommendations using a distributed collaborative filtering architecture.* In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, pages 394–401, New York, NY, USA, 2004. ACM. (Cited on pages 48 and 50.)

[Amatriain 2009] Xavier Amatriain, Neal Lathia, Josep M. Pujol, Haewoon Kwak and Nuria Oliver. *The wisdom of the few: a collaborative filtering approach based on expert opinions from the web.* In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, pages 532–539, New York, NY, USA, 2009. ACM. (Cited on page 50.)

[Amatriain 2013] Xavier Amatriain. *Mining Large Streams of User Data for Personalized Recommendations.* SIGKDD Explor. Newsl., vol. 14, no. 2, pages 37–48, April 2013. (Cited on page 33.)

[Bakken 2004] D. E. Bakken, R. Rarameswaran, D. M. Blough, A. A. Franz and T. J. Palmer. *Data obfuscation: anonymity and desensitization of usable data sets.* Security & Privacy, IEEE, vol. 2, no. 6, pages 34–41, November 2004. (Cited on page 39.)

[Bakker 2009] Arno Bakker, Elth Ogston and Maarten van Steen. *Collaborative filtering using random neighbours in peer-to-peer networks.* In CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management, pages 67–75, New York, NY, USA, 2009. ACM. (Cited on pages 47 and 49.)

[Balabanović 1997] Marko Balabanović and Yoav Shoham. *Fab: content-based, collaborative recommendation.* Commun. ACM, vol. 40, pages 66–72, March 1997. (Cited on pages 11 and 20.)

[Bell 2007] Robert M. Bell and Yehuda Koren. *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights.* In Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, ICDM '07, pages 43–52, Washington, DC, USA, October 2007. IEEE Computer Society. (Cited on page 17.)

[Bellogín Kouki 2012] Alejandro Bellogín Kouki. *Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective.* PhD thesis, Universidad Autónoma de Madrid, October 2012. (Cited on page 20.)

[Bellogin 2011] Alejandro Bellogin, Pablo Castells and Ivan Cantador. *Precision-oriented evaluation of recommender systems: an algorithmic comparison.* In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM. (Cited on page 26.)

[Berkovsky 2007] Shlomo Berkovsky, Yaniv Eytani, Tsvi Kuflik and Francesco Ricci. *Enhancing privacy and preserving accuracy of a distributed collaborative filtering.* In Proceedings of the 2007 ACM conference on Recommender systems, RecSys '07, pages 9–16, New York, NY, USA, 2007. ACM. (Cited on pages 51, 58 and 59.)

[Bhagat 2014] Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis and Nina Taft. *Recommending with an Agenda: Active Learning of Private Attributes Using Matrix Factorization.* In Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, pages 65–72, New York, NY, USA, 2014. ACM. (Cited on page 42.)

[Bianchi 2006] Nicolo C. Bianchi and Gabor Lugosi. Prediction, learning, and games. Cambridge University Press, New York, NY, USA, 2006. (Cited on page 90.)

[Bilenko 2011] Mikhail Bilenko and Matthew Richardson. *Predictive client-side profiles for personalized advertising.* In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, pages 413–421, New York, NY, USA, 2011. ACM. (Cited on pages 48, 50 and 100.)

[Blanco-Fernández 2008] Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Martín López-Nores, Jorge García-Duque, Ana Fernández-Vilas and Rebeca P. Díaz-Redondo. *Exploiting synergies between semantic reasoning and personalization strategies in intelligent recommender systems: A case study.* J. Syst. Softw., vol. 81, no. 12, pages 2371–2385, 2008. (Cited on page 13.)

[Borlund 2003] Pia Borlund. *The concept of relevance in IR.* J. Am. Soc. Inf. Sci., vol. 54, no. 10, pages 913–925, August 2003. (Cited on page 9.)

[Bottou 1998] Léon Bottou. *Online Algorithms and Stochastic Approximations.* In David Saad, editeur, Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012. (Cited on pages 72 and 73.)

[Bottou 2008] Léon Bottou and Olivier Bousquet. *The Tradeoffs of Large Scale Learning.* In J. C. Platt, D. Koller, Y. Singer and S. Roweis, editeurs, Advances in Neural Information Processing Systems 20, pages 161–168, 2008. (Cited on page 27.)

[Bottou 2010] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent.* In Yves Lechevallier and Gilbert Saporta, editeurs, Proceedings of COMPSTAT'2010, pages 177–186. Physica-Verlag HD, 2010. (Cited on page 27.)

[Buckley 1995] Chris Buckley and Gerard Salton. *Optimization of relevance feedback weights.* In Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '95, pages 351–357, New York, NY, USA, 1995. ACM. (Cited on pages 11 and 13.)

[Burke 2002] Robin Burke. *Hybrid Recommender Systems: Survey and Experiments.* User Modeling and User-Adapted Interaction, vol. 12, no. 4, pages 331–370, November 2002. (Cited on pages 10, 18, 19, 20 and 85.)

[Calandrino 2011] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten and V. Shmatikov. *"You Might Also Like:" Privacy Risks of Collaborative Filtering.* In Security and Privacy (SP), 2011 IEEE Symposium on, pages 231–246. IEEE, May 2011. (Cited on page 42.)

[Canny 2002] J. Canny. *Collaborative filtering with privacy*. Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on, pages 45–57, 2002. (Cited on pages 54, 55, 61, 62 and 63.)

[Cantador 2011] Iván Cantador, Peter Brusilovsky and Tsvi Kuflik. *2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)*. In Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011, New York, NY, USA, 2011. ACM. (Cited on page 89.)

[Castagnos 2007] Sylvain Castagnos and Anne Boyer. *Modeling Preferences in a Distributed Recommender System*. In UM '07: Proceedings of the 11th international conference on User Modeling, pages 400–404, Berlin, Heidelberg, 2007. Springer-Verlag. (Cited on pages 46, 49 and 100.)

[Chen 2011] Yunmei Chen and Xiaojing Ye. *Projection Onto A Simplex*, February 2011. (Cited on page 73.)

[Chen 2012] Xiaoqiang Chen and V. Huang. *Privacy Preserving Data Publishing for Recommender System*. In Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, pages 128–133. IEEE, July 2012. (Cited on pages 45 and 46.)

[Clemen 1999] Robert T. Clemen and Robert L. Winkler. *Combining Probability Distributions From Experts in Risk Analysis*. Risk Analysis, vol. 19, no. 2, pages 187–203, April 1999. (Cited on page 80.)

[Cormode 2005] Graham Cormode and S. Muthukrishnan. *An Improved Data Stream Summary: The Count-min Sketch and Its Applications*. J. Algorithms, vol. 55, no. 1, pages 58–75, April 2005. (Cited on page 87.)

[Cranor 2002] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall and Joseph Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. `http://www.w3.org/TR/P3P/`, April 2002. (Cited on page 38.)

[Crestani 1997] F. Crestani. *Application of Spreading Activation Techniques in Information Retrieval*. Artif. Intell. Rev., vol. 11, no. 6, pages 453–482, December 1997. (Cited on page 13.)

[Damgård 2001] Ivan Damgård and Mads Jurik. *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*. In Kwangjo Kim, editeur, Public Key Cryptography, volume 1992 of *Lecture Notes in Computer Science*, chapitre 9, pages 119–136. Springer Berlin Heidelberg, Berlin, Heidelberg, June 2001. (Cited on page 60.)

[Das 2007] Abhinandan S. Das, Mayur Datar, Ashutosh Garg and Shyam Rajaram. *Google news personalization: scalable online collaborative filtering*. In Proceedings of the 16th international conference on World Wide Web, WWW

'07, pages 271–280, New York, NY, USA, 2007. ACM. (Cited on pages 20 and 36.)

[Dasiopoulou 2011] Stamatia Dasiopoulou, Eirini Giannakidou, Georgios Litos, Polyxeni Malasioti and Yiannis Kompatsiaris. *A Survey of Semantic Image and Video Annotation Tools*. In Georgios Paliouras, Constantine Spyropoulos and George Tsatsaronis, editeurs, Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, volume 6050 of *Lecture Notes in Computer Science*, chapitre 8, pages 196–239. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2011. (Cited on page 12.)

[Davidson 2010] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston and Dasarathi Sampath. *The YouTube video recommendation system*. In Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM. (Cited on page 36.)

[Del Prete 2010] L. Del Prete and L. Capra. *diffeRS: A Mobile Recommender Service*. In Mobile Data Management (MDM), 2010 Eleventh International Conference on, pages 21–26. IEEE, May 2010. (Cited on pages 47 and 49.)

[Di Noia 2014] Tommaso Di Noia, Iván Cantador and VitoClaudio Ostuni. *Linked Open Data-Enabled Recommender Systems: ESWC 2014 Challenge on Book Recommendation*. In Valentina Presutti, Milan Stankovic, Erik Cambria, Iván Cantador, Angelo Di Iorio, Tommaso Di Noia, Christoph Lange, Diego Reforgiato Recupero and Anna Tordai, editeurs, Semantic Web Evaluation Challenge, volume 475 of *Communications in Computer and Information Science*, pages 129–143. Springer International Publishing, 2014. (Cited on page 105.)

[Dimitropoulos 2008] Xenofontas Dimitropoulos, Marc Stoecklin, Paul Hurley and Andreas Kind. *The Eternal Sunshine of the Sketch Data Structure*. Comput. Netw., vol. 52, no. 17, pages 3248–3257, December 2008. (Cited on page 87.)

[Dokoohaki 2010] Nima Dokoohaki, Cihan Kaleli, Huseyin Polat and Mihhail Matskin. *Achieving Optimal Privacy in Trust-Aware Social Recommender Systems*. In Leonard Bolc, Marek Makowski and Adam Wierzbicki, editeurs, Social Informatics, volume 6430 of *Lecture Notes in Computer Science*, chapitre 5, pages 62–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. (Cited on pages 52, 53, 58 and 59.)

[Dooms 2013] Simon Dooms, Toon De Pessemier and Luc Martens. *MovieTweetings: a Movie Rating Dataset Collected From Twitter*. In Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013, 2013. (Cited on page 43.)

[Draidi 2011] Fady Draidi, Esther Pacitti and Bettina Kemme. *P2Prec: A P2P Recommendation System for Large-Scale Data Sharing Transactions on Large-Scale Data- and Knowledge-Centered Systems III.* In Abdelkader Hameurlain, Josef Küng and Roland Wagner, editeurs, Transactions on Large-Scale Data- and Knowledge-Centered Systems III, volume 6790 of *Lecture Notes in Computer Science*, chapitre 4, pages 87–116. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2011. (Cited on pages 47, 49 and 100.)

[Duan 2010] Yitao Duan, John Canny and Justin Zhan. *P4P: practical large-scale privacy-preserving distributed computation robust against malicious users.* In Proceedings of the 19th USENIX conference on Security, USENIX Security'10, page 14, Berkeley, CA, USA, 2010. USENIX Association. (Cited on pages 54, 55, 58 and 59.)

[Duboc 2007] Leticia Duboc, David Rosenblum and Tony Wicks. *A framework for characterization and analysis of software system scalability.* In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ESEC-FSE '07, pages 375–384, New York, NY, USA, 2007. ACM. (Cited on page 25.)

[Dwork 2006] Cynthia Dwork. *Differential Privacy.* In Michele Bugliesi, Bart Preneel, Vladimiro Sassone and Ingo Wegener, editeurs, Automata, Languages and Programming, volume 4052 of *Lecture Notes in Computer Science*, chapitre 1, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. (Cited on pages 6, 42 and 98.)

[Dwork 2008] Cynthia Dwork. *Differential privacy: a survey of results.* In Proceedings of the 5th international conference on Theory and applications of models of computation, TAMC'08, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on pages 39 and 98.)

[Dwork 2010] Cynthia Dwork, Moni Naor, Toniann Pitassi and Guy N. Rothblum. *Differential privacy under continual observation.* In Proceedings of the 42nd ACM symposium on Theory of computing, STOC '10, pages 715–724, New York, NY, USA, 2010. ACM. (Cited on page 44.)

[Ehrig 2004] Marc Ehrig, Peter Haase, Mark Hefke and Nenad Stojanovic. *Similarity for Ontologies - a Comprehensive Framework.* In In Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004, 2004. (Cited on pages 12 and 13.)

[Elmisery 2011] Ahmed M. Elmisery and Dmitri Botvich. *Privacy Aware Recommender Service for IPTV Networks.* In Proceedings of the 2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering, MUE '11, pages 160–166, Washington, DC, USA, 2011. IEEE Computer Society. (Cited on pages 53 and 54.)

[Erkin 2012] Z. Erkin, T. Veugen, T. Toft and R. L. Lagendijk. *Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing*. Information Forensics and Security, IEEE Transactions on, vol. 7, no. 3, pages 1053–1066, June 2012. (Cited on pages 54, 55, 62 and 63.)

[European Commission 2012] European Commission. *COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE COMMITTEE OF THE REGIONS Safeguarding Privacy in a Connected World A European Data Protection Framework for the 21st Century*. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52012DC0009:en:NOT, January 2012. (Cited on page 38.)

[Foner 1999] Leonard N. Foner. *Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System*. PhD thesis, Program in Media Arts and Sciences, School of Architecture and Planning, Massachusetts Institute of Technology, June 1999. (Cited on pages 4 and 36.)

[Funk 2006] Simon Funk. Netflix update: Try this at home. Accesed online March 2013 [http://sifter.org/˜simon/journal/20061211.html], 2006. (Cited on page 17.)

[Gantner 2010] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle and L. Schmidt-Thieme. *Learning Attribute-to-Feature Mappings for Cold-Start Recommendations*. In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pages 176–185. IEEE, December 2010. (Cited on pages 20 and 105.)

[Guha 2009] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi and Paul Francis. *Serving Ads from localhost for Performance, Privacy, and Profit*. In Proceedings of the 8th Workshop on Hot Topics in Networks (HotNets), New York, NY, Oct 2009. (Cited on page 19.)

[Gunawardana 2009] Asela Gunawardana and Christopher Meek. *A unified approach to building hybrid recommender systems*. In Proceedings of the third ACM conference on Recommender systems, RecSys '09, pages 117–124, New York, NY, USA, 2009. ACM. (Cited on page 20.)

[Halkidi 2011] Maria Halkidi and Iordanis Koutsopoulos. *A Game Theoretic Framework for Data Privacy Preservation in Recommender Systems*. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba and Michalis Vazirgiannis, editeurs, Machine Learning and Knowledge Discovery in Databases, volume 6911 of *Lecture Notes in Computer Science*, pages 629–644. Springer Berlin Heidelberg, 2011. (Cited on pages 52, 53 and 64.)

[Han 2004] P. Han. *A scalable P2P recommender system based on distributed collaborative filtering*. Expert Systems with Applications, vol. 27, no. 2, pages 203–210, August 2004. (Cited on pages 46 and 49.)

[Hanani 2001] Uri Hanani, Bracha Shapira and Peretz Shoval. *Information Filtering: Overview of Issues, Research and Systems.* User Modeling and User-Adapted Interaction, vol. 11, pages 203–259, August 2001. (Cited on page 10.)

[Hoens 2010] T. Ryan Hoens, Marina Blanton and Nitesh V. Chawla. *A Private and Reliable Recommendation System for Social Networks.* In Proceedings of the 2010 IEEE Second International Conference on Social Computing, SOCIALCOM '10, pages 816–825, Washington, DC, USA, 2010. IEEE Computer Society. (Cited on pages 51, 52, 62 and 63.)

[Hsieh 2011] Chia-Lung Hsieh. *Toward Better Recommender System by Collaborative Computation with Privacy Preserved.* In 2011 IEEE/IPSJ International Symposium on Applications and the Internet, pages 246–249. IEEE, July 2011. (Cited on page 51.)

[Hsu 2007] Shang Hsu, Ming-Hui Wen, Hsin-Chieh Lin, Chun-Chia Lee and Chia-Hoang Lee. *AIMED- A Personalized TV Recommendation System.* In Pablo Cesar, Konstantinos Chorianopoulos and Jens Jensen, editeurs, Interactive TV: a Shared Experience, volume 4471 of *Lecture Notes in Computer Science*, chapitre 18, pages 166–174. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2007. (Cited on page 12.)

[Isaacman 2011] Sibren Isaacman, Stratis Ioannidis, Augustin Chaintreau and Margaret Martonosi. *Distributed rating prediction in user generated content streams.* In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 69–76, New York, NY, USA, 2011. ACM. (Cited on pages v, 48, 49, 70, 73, 75, 76, 81 and 103.)

[Jeckmans 2013] ArjanJ Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, ReginaldL Lagendijk and Qiang Tang. *Privacy in Recommender Systems.* In Naeem Ramzan, Roelof Zwol, Jong-Seok Lee, Kai Clüver and Xian-Sheng Hua, editeurs, Social Media Retrieval, Computer Communications and Networks, pages 263–281. Springer London, 2013. (Cited on page 38.)

[Jelasity 2009] Márk Jelasity, Alberto Montresor and Ozalp Babaoglu. *T-Man: Gossip-based fast overlay topology construction.* Computer Networks, vol. 53, no. 13, pages 2321–2339, August 2009. (Cited on page 46.)

[Jiang 2009] Xing Jiang and Ah-Hwee Tan. *Learning and inferencing in user ontology for personalized Semantic Web search.* Information Sciences, vol. 179, no. 16, pages 2794–2808, July 2009. (Cited on page 13.)

[Kaleli 2010] Cihan Kaleli and Hüseyin Polat. *P2P collaborative filtering with privacy.* Turkish Journal of Electric Electrical Engineering and Computer Sciences, vol. 8, no. 1, pages 101–116, 2010. (Cited on pages 51 and 64.)

[Kermarrec 2010] Anne-Marie Kermarrec, Vincent Leroy, Afshin Moin and Christopher Thraves. *Application of Random Walks to Decentralized Recommender*

Systems. In Chenyang Lu, Toshimitsu Masuzawa and Mohamed Mosbah, editeurs, Principles of Distributed Systems, volume 6490 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin Heidelberg, 2010. (Cited on pages 46 and 49.)

[Kim 2008] Jae K. Kim, Hyea K. Kim and Yoon H. Cho. *A user-oriented contents recommendation system in peer-to-peer architecture*. Expert Systems with Applications, vol. 34, no. 1, pages 300–312, January 2008. (Cited on pages 46 and 49.)

[King 2010] Nancy J. King and Pernille W. Jessen. *Profiling the mobile customer - Privacy concerns when behavioural advertisers target mobile phones - Part I*. Computer Law & Security Review, vol. 26, no. 5, pages 455–478, September 2010. (Cited on page 37.)

[Kobsa 2007] Alfred Kobsa. *Privacy-enhanced personalization*. Commun. ACM, vol. 50, no. 8, pages 24–33, August 2007. (Cited on pages 38, 40, 70 and 95.)

[Kobsa 2014] Alfred Kobsa, Bart P. Knijnenburg and Benjamin Livshits. *Let's Do It at My Place Instead?: Attitudinal and Behavioral Study of Privacy in Client-side Personalization*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '14, pages 81–90, New York, NY, USA, 2014. ACM. (Cited on page 40.)

[Kohavi 2009] Ron Kohavi, Roger Longbotham, Dan Sommerfield and RandalM Henne. *Controlled experiments on the web: survey and practical guide*. Data Mining and Knowledge Discovery, vol. 18, no. 1, pages 140–181, February 2009. (Cited on page 21.)

[Koren 2008] Yehuda Koren. *Factorization meets the neighborhood: a multifaceted collaborative filtering model*. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM. (Cited on page 17.)

[Koren 2011] Yehuda Koren and Joe Sill. *OrdRec: an ordinal model for predicting personalized item rating distributions*. In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 117–124, New York, NY, USA, 2011. ACM. (Cited on page 17.)

[Koychev 2000] Ivan Koychev and Ingo Schwab. *Adaptation to Drifting User's Interests*. In In Proceedings of ECML2000 Workshop: Machine Learning in New Information Age, pages 39–46, 2000. (Cited on page 88.)

[Lam 2006] Shyong Lam, Dan Frankowski and John Riedl. *Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems*. In Günter Müller, editeur, Emerging Trends in Information and Communication Security, volume 3995 of *Lecture Notes in Computer*

Science, chapitre 2, pages 14–29. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006. (Cited on page 36.)

[Lang 1995] Ken Lang. *NewsWeeder: learning to filter netnews*. In Proceedings of the 12th International Conference on Machine Learning, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995. (Cited on page 11.)

[Lathia 2007] Neal Lathia, Stephen Hailes and Licia Capra. *Private distributed collaborative filtering using estimated concordance measures*. In Proceedings of the 2007 ACM conference on Recommender systems, RecSys '07, pages 1–8, New York, NY, USA, 2007. ACM. (Cited on page 50.)

[L'Ecuyer 2007] Pierre L'Ecuyer. *Random Number Generation*. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, pages 93–137, 2007. (Cited on page 58.)

[Lieberman 1995] Henry Lieberman. *Letizia: an agent that assists web browsing*. In Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1, IJCAI'95, pages 924–929, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. (Cited on page 12.)

[Linden 2003] G. Linden, B. Smith and J. York. *Amazon.com recommendations: item-to-item collaborative filtering*. IEEE Internet Computing, vol. 7, no. 1, pages 76–80, January 2003. (Cited on pages 15 and 36.)

[Luo 2012] Xin Luo, Yunni Xia and Qingsheng Zhu. *Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization*. Knowledge-Based Systems, vol. 27, pages 271–280, March 2012. (Cited on pages 29 and 32.)

[Magureanu 2012] S. Magureanu, N. Dokoohaki, S. Mokarizadeh and M. Matskin. *Epidemic Trust-Based Recommender Systems*. In Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom), pages 461–470. IEEE, 2012. (Cited on pages 48 and 49.)

[Marinho 2011] Leandro B. Marinho, Alexandros Nanopoulos, Lars Schmidt-Thieme, Robert Jäschke, Andreas Hotho, Gerd Stumme and Panagiotis Symeonidis. *Social Tagging Recommender Systems*. In Francesco Ricci, Lior Rokach, Bracha Shapira and Paul B. Kantor, editeurs, Recommender Systems Handbook, chapitre 19, pages 615–644. Springer US, Boston, MA, 2011. (Cited on page 12.)

[Massa 2009] Paolo Massa and Paolo Avesani. *Trust Metrics in Recommender Systems*. In Jennifer Golbeck, editeur, Computing with Social Trust, Human-Computer Interaction Series, chapitre 10, pages 259–285. Springer London, London, 2009. (Cited on page 47.)

[Matsumoto 1998] Makoto Matsumoto and Takuji Nishimura. *Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudo-random Number Generator*. ACM Trans. Model. Comput. Simul., vol. 8, no. 1, pages 3–30, January 1998. (Cited on page 58.)

[McMahan 2013] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar M. Hrafnkelsson, Tom Boulos and Jeremy Kubica. *Ad Click Prediction: A View from the Trenches*. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, pages 1222–1230, New York, NY, USA, 2013. ACM. (Cited on page 86.)

[McSherry 2009] Frank McSherry and Ilya Mironov. *Differentially private recommender systems: building privacy into the net*. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, pages 627–636, New York, NY, USA, 2009. ACM. (Cited on pages 42, 45 and 46.)

[Middleton 2004] Stuart E. Middleton, Nigel R. Shadbolt and David C. De Roure. *Ontological user profiling in recommender systems*. ACM Trans. Inf. Syst., vol. 22, no. 1, pages 54–88, January 2004. (Cited on pages 12 and 13.)

[Miller 2004] Bradley N. Miller, Joseph A. Konstan and John Riedl. *PocketLens: Toward a personal recommender system*. ACM Transactions on Information Systems, vol. 22, no. 3, pages 437–476, July 2004. (Cited on pages 47, 49 and 55.)

[Moreno 2014a] Andrés Moreno, Christian Ariza-Porras, Paula Lago, ClaudiaLucía Jiménez-Guarín, Harold Castro and Michel Riveill. *Hybrid Model Rating Prediction with Linked Open Data for Recommender Systems*. In Valentina Presutti, Milan Stankovic, Erik Cambria, Iván Cantador, Angelo Di Iorio, Tommaso Di Noia, Christoph Lange, Diego Reforgiato Recupero and Anna Tordai, editeurs, Semantic Web Evaluation Challenge, volume 475 of *Communications in Computer and Information Science*, pages 193–198. Springer International Publishing, 2014. (Cited on page 105.)

[Moreno 2014b] Andrés Moreno, Harold Castro and Michel Riveill. *Client-Side Hybrid Rating Prediction for Recommendation*. In Vania Dimitrova, Tsvi Kuflik, David Chin, Francesco Ricci, Peter Dolog and Geert-Jan Houben, editeurs, User Modeling, Adaptation, and Personalization, volume 8538 of *Lecture Notes in Computer Science*, pages 369–380. Springer International Publishing, 2014. (Cited on page 104.)

[Narayanan 2008] A. Narayanan and V. Shmatikov. *Robust De-anonymization of Large Sparse Datasets*. In Security and Privacy, 2008. SP 2008. IEEE

Symposium on, volume 0, pages 111–125, Los Alamitos, CA, USA, May 2008. IEEE. (Cited on pages 41 and 42.)

[Netflix 2009] Netflix. *Netflix Prize*. http://www.netflixprize.com/, September 2009. (Cited on page 5.)

[Netflix 2013] Netflix. *Netflix Cinematch*. http://www.netflix.com, April 2013. (Cited on pages 4 and 36.)

[Oku 2006] K. Oku, S. Nakajima, J. Miyazaki and S. Uemura. *Context-Aware SVM for Context-Dependent Information Recommendation*. In MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06), page 109, Washington, DC, USA, 2006. IEEE Computer Society. (Cited on page 12.)

[Ormándi 2010] Róbert Ormándi, István Hegedűs and Márk Jelasity. *Overlay Management for Fully Distributed User-Based Collaborative Filtering*. In Pasqua D'Ambra, Mario Guarracino and Domenico Talia, editeurs, Euro-Par 2010 - Parallel Processing, volume 6271 of *Lecture Notes in Computer Science*, chapitre 43, pages 446–457. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010. (Cited on pages 47 and 49.)

[Paillier 1999] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. In Jacques Stern, editeur, Advances in Cryptology EUROCRYPT '99, volume 1592 of *Lecture Notes in Computer Science*, chapitre 16, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. (Cited on page 60.)

[Papadogiorgaki 2008] Maria Papadogiorgaki, Vasileios Papastathis, Evangelia Nidelkou, Simon Waddington, Ben Bratu, Myriam Ribiere and Ioannis Kompatsiaris. *Two-Level Automatic Adaptation of a Distributed User Profile for Personalized News Content Delivery*. International Journal of Digital Multimedia Broadcasting, vol. 2008, 2008. (Cited on page 13.)

[Papagelis 2005] Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis and Elias Theoharopoulos. *Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms*. In Mohand-Said Hacid, NeilV Murray, ZbigniewW Raś and Shusaku Tsumoto, editeurs, Foundations of Intelligent Systems, volume 3488 of *Lecture Notes in Computer Science*, pages 553–561. Springer Berlin Heidelberg, 2005. (Cited on pages 28 and 32.)

[Parameswaran 2007] R. Parameswaran and D. M. Blough. *Privacy Preserving Collaborative Filtering Using Data Obfuscation*. In Granular Computing, 2007. GRC 2007. IEEE International Conference on, page 380. IEEE, November 2007. (Cited on page 45.)

[Parra-Arnau 2012] Javier Parra-Arnau, David Rebollo-Monedero and Jordi Forné. *A Privacy-Protecting Architecture for Collaborative Filtering via Forgery and*

Suppression of Ratings. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, Nora Cuppens-Boulahia and Sabrina Capitani di Vimercati, editeurs, Data Privacy Management and Autonomous Spontaneus Security, volume 7122 of *Lecture Notes in Computer Science*, pages 42–57. Springer Berlin Heidelberg, 2012. (Cited on pages 53 and 54.)

[Pazzani 1997] Michael Pazzani and Daniel Billsus. *Learning and Revising User Profiles: The Identification of Interesting Web Sites*. Machine Learning, vol. 27, no. 3, pages 313–331, June 1997. (Cited on page 12.)

[Pazzani 1999] Michael J. Pazzani. *A Framework for Collaborative, Content-Based and Demographic Filtering*. Artif. Intell. Rev., vol. 13, no. 5-6, pages 393–408, December 1999. (Cited on page 20.)

[Pedersen 1991] TorbenPryds Pedersen. *A Threshold Cryptosystem without a Trusted Party*. In DonaldW Davies, editeur, Advances in Cryptology EUROCRYPT '91, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer Berlin Heidelberg, 1991. (Cited on pages 52 and 55.)

[Pentland 2001] Tony Jebara Alex Pentland. *On reversing Jensen's inequality*. In Advances in neural information processing systems 13: proceedings of the 2000 conference, volume 13, page 231. The MIT Press, 2001. (Cited on page 18.)

[Pilászy 2009] István Pilászy and Domonkos Tikk. *Recommending new movies: even a few ratings are more valuable than metadata*. In Proceedings of the third ACM conference on Recommender systems, RecSys '09, pages 93–100, New York, NY, USA, 2009. ACM. (Cited on pages 4, 19, 20 and 21.)

[Polat 2005] Huseyin Polat and Wenliang Du. *SVD-based collaborative filtering with privacy*. In Proceedings of the 2005 ACM symposium on Applied computing, SAC '05, pages 791–795, New York, NY, USA, 2005. ACM. (Cited on pages 52, 53, 58 and 59.)

[Renckes 2012] Sahin Renckes, Huseyin Polat and Yusuf Oysal. *A new hybrid recommendation algorithm with privacy*. Expert Systems, vol. 29, no. 1, pages 39–55, February 2012. (Cited on pages 52, 53, 58 and 59.)

[Rendle 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars S. Thieme. *BPR: Bayesian personalized ranking from implicit feedback*. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press. (Cited on pages 18, 21 and 105.)

[Resnick 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom and John Riedl. *GroupLens: an open architecture for collaborative filtering of netnews*. In Proceedings of the 1994 ACM conference on Computer supported

cooperative work, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM. (Cited on page 15.)

[Salton 1975] G. Salton, A. Wong and C. S. Yang. *A vector space model for automatic indexing*. Commun. ACM, vol. 18, no. 11, pages 613–620, November 1975. (Cited on page 11.)

[Salton 1988] Gerard Salton and Christopher Buckley. *Term-weighting approaches in automatic text retrieval*. Inf. Process. Manage., vol. 24, no. 5, pages 513–523, August 1988. (Cited on page 11.)

[Sarwar 2001] Badrul Sarwar, George Karypis, Joseph Konstan and John Reidl. *Item-based collaborative filtering recommendation algorithms*. In Proceedings of the 10th international conference on World Wide Web, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. (Cited on page 15.)

[Sarwar 2002] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl. *Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems*. In Fifth International Conference on Computer and Information Science, pages 27–28, 2002. (Cited on pages 16, 29 and 32.)

[Schapire 2012] Robert E. Schapire and Yoav Freund. Boosting: Foundations and algorithms (adaptive computation and machine learning series). The MIT Press, May 2012. (Cited on page 27.)

[Schelter 2012] Sebastian Schelter, Christoph Boden and Volker Markl. *Scalable Similarity-based Neighborhood Methods with MapReduce*. In Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pages 163–170, New York, NY, USA, 2012. ACM. (Cited on page 82.)

[Schelter 2013] Sebastian Schelter, Christoph Boden, Martin Schenck, Alexander Alexandrov and Volker Markl. *Distributed Matrix Factorization with Mapreduce Using a Series of Broadcast-joins*. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pages 281–284, New York, NY, USA, 2013. ACM. (Cited on pages 4 and 82.)

[Schifanella 2008] Rossano Schifanella, André Panisson, Cristina Gena and Giancarlo Ruffo. *MobHinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks*. In RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, pages 27–34, New York, NY, USA, 2008. ACM. (Cited on pages 47 and 49.)

[Shani 2011] Guy Shani and Asela Gunawardana. *Evaluating Recommendation Systems*. In Francesco Ricci, Lior Rokach, Bracha Shapira and Paul B. Kantor, editeurs, Recommender Systems Handbook, chapitre 8, pages 257–297. Springer US, Boston, MA, 2011. (Cited on pages 5, 21 and 22.)

[Shi 2012] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver and Alan Hanjalic. *CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering*. In Proceedings of the sixth ACM conference on Recommender systems, RecSys '12, pages 139–146, New York, NY, USA, 2012. ACM. (Cited on pages 18 and 105.)

[Shokri 2009] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos and Jean P. Hubaux. *Preserving privacy in collaborative filtering through distributed aggregation of offline profiles*. In Proceedings of the third ACM conference on Recommender systems, RecSys '09, pages 157–164, New York, NY, USA, 2009. ACM. (Cited on pages 53, 54 and 64.)

[Sieg 2007] Ahu Sieg, Bamshad Mobasher and Robin Burke. *Web search personalization with ontological user profiles*. In CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pages 525–534, New York, NY, USA, 2007. ACM. (Cited on page 13.)

[Singel 2009] Ryan Singel. *Netflix Spilled Your Brokeback Mountain Secret, Lawsuit Claims*. http://www.wired.com/threatlevel/2009/12/netflix-privacy-lawsuit/, December 2009. (Cited on page 5.)

[Spiekermann 2009] S. Spiekermann and L. F. Cranor. *Engineering Privacy*. Software Engineering, IEEE Transactions on, vol. 35, no. 1, pages 67–82, January 2009. (Cited on page 37.)

[Sweeney 2002] Latanya Sweeney. *k-anonymity: a model for protecting privacy*. Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol. 10, no. 5, pages 557–570, October 2002. (Cited on page 39.)

[Toch 2012] Eran Toch, Yang Wang and LorrieFaith Cranor. *Personalization and privacy: a survey of privacy risks and remedies in personalization-based systems*. User Modeling and User-Adapted Interaction, vol. 22, no. 1-2, pages 203–220, April 2012. (Cited on page 38.)

[Tomozei 2011] Dan-Cristian Tomozei and Laurent Massoulié. *Distributed User Profiling via Spectral Methods*, September 2011. (Cited on pages 48 and 49.)

[Tveit 2001] Amund Tveit. *Peer-to-peer based recommendations for mobile commerce*. In WMC '01: Proceedings of the 1st international workshop on Mobile commerce, pages 26–29, New York, NY, USA, 2001. ACM. (Cited on pages 40, 46, 49 and 51.)

[Vallet 2006] David Vallet, Iván Cantador, Miriam Fernández and Pablo Castells. *A Multi-Purpose Ontology-Based Approach for Personalized Content Filtering and Retrieval*. Semantic Media Adaptation and Personalization, International Workshop on, vol. 0, pages 19–24, 2006. (Cited on page 13.)

[Vallet 2014] David Vallet, Arik Friedman and Shlomo Berkovsky. *Matrix Factorization without User Data Retention.* In VincentS Tseng, TuBao Ho, Zhi-Hua Zhou, ArbeeL Chen and Hung-Yu Kao, editeurs, Advances in Knowledge Discovery and Data Mining, volume 8443 of *Lecture Notes in Computer Science*, pages 569–580. Springer International Publishing, 2014. (Cited on page 50.)

[Vapnik 1999] V. N. Vapnik. *An overview of statistical learning theory.* Neural Networks, IEEE Transactions on, vol. 10, no. 5, pages 988–999, September 1999. (Cited on page 26.)

[W3C 2012] W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition).* http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52012DC0009:en:NOT, January 2012. (Cited on page 12.)

[Warner 1965] Stanley L. Warner. *Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias.* Journal of the American Statistical Association, vol. 60, no. 309, pages 63+, March 1965. (Cited on page 64.)

[Xin 2014] Yu Xin and Tommi Jaakkola. *Controlling privacy in recommender systems.* In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger, editeurs, Advances in Neural Information Processing Systems 27, pages 2618–2626. Curran Associates, Inc., 2014. (Cited on pages 53 and 54.)

[Xu 2011] Wei Xu. *Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent*, December 2011. (Cited on pages 73 and 88.)

[Zhan 2010] J. Zhan, Chia-Lung Hsieh, I-Cheng Wang, Tsan-Sheng Hsu, Churn-Jung Liau and Da-wei Wang. *Privacy-Preserving Collaborative Recommender Systems.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 40, no. 4, pages 472–476, July 2010. (Cited on page 51.)

[Zhao 2011] Yu Zhao, Xinping Feng, Jianqiang Li and Bo Liu. *Shared collaborative filtering.* In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 29–36, New York, NY, USA, 2011. ACM. (Cited on pages 45 and 46.)

[Zhen 2009] Yi Zhen, Wu J. Li and Dit Y. Yeung. *TagiCoFi: tag informed collaborative filtering.* In Proceedings of the third ACM conference on Recommender systems, RecSys '09, pages 69–76, New York, NY, USA, 2009. ACM. (Cited on page 12.)

[Ziegler 2005] Cai-Nicolas Ziegler. *Towards decentralized recommender systems.* PhD thesis, University of Freiburg, 2005. http://d-nb.info/975319213. (Cited on pages 47 and 49.)

Privacy-enabled scalable recommender systems

Abstract: The main objective of this thesis is to propose a recommendation method that keeps in mind the privacy of users as well as the scalability of the system.

To achieve this goal, an hybrid technique using content-based and collaborative filtering paradigms is used in order to attain an accurate model for recommendation, under the strain of mechanisms designed to keep user privacy, particularly designed to reduce the user exposure risk.

At first, the related work on privacy-enabled recommender systems was explored by keeping in mind the privacy risks that the centralized gathering and processing of user profile information brings to the information of users. From this analysis, some design criteria for a privacy-enabled recommender system are found. Succinctly, a client-side architecture is favored for privacy reasons, and in order to keep an scalable masking strategy of user profile information random noise generation should be used.

Next a privacy-enabled collaborative filtering approach is defined. In this strategy a stochastic approximation of the item and user profile is calculated using a client-side architecture that interacts with public information about items kept on the recommender system side. Later, this model is extended into an hybrid approach for recommendation that includes a content-based strategy for content recommendation. Using a knowledge model based on keywords that describe the item domain, the hybrid approach increases the predictive performance of the models without much computational effort on the cold-start setting.

Finally, some strategies to improve the recommender system's provided privacy are introduced: Random noise generation is used to limit the possible inferences an attacker can make when continually observing the interaction between the client-side agent and the server, and a blacklisted strategy is used to refrain the server from learning interactions that the user considers violate her privacy. The use of the hybrid model mitigates the negative impact these strategies cause on the predictive performance of the recommendations.

Keywords: Privacy, Recommender systems, User profiling