From Traditional to Context-Aware Recommendations by Correlation-Based Context Model

Zahra Vahidi Ferdousi

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'université Paris-Dauphine

From Traditional to Context-Aware Recommendations by Correlation-Based Context Model

Soutenue par
Zahra Vahidi Ferdousi
Le **18 décembre 2020**

École doctorale n°543
Ecole Doctorale SDOSE

Spécialité
Informatique

Composition du jury :

Dr. Armelle Brun, HDR Maître de Conférences, Université de Lorraine	*Rapporteur*
Pr. Max Chevalier Professeur, Université de Toulouse	*Rapporteur et président du jury*
Pr. Jacky Akoka Professeur, CEDRIC-CNAM et IMT-BS	*Examinateur*
Dr. Florian Yger Maître de Conférences, Université Paris Dauphine	*Examinateur*
Pr. Dario Colazzo Professeur, Université Paris Dauphine	*Co-directeur*
Dr. Elsa Negre, HDR Maître de Conférences, Université Paris Dauphine	*Co-directrice*
M. Julien Blaize Ingénieur, Rakouten	*Représentant entreprise*

DAUPHINE UNIVERSITÉ PARIS | PSL

Acknowledgement

Many people have contributed either directly or indirectly to the success of this work, to whom I extend my deepest appreciation.

First and foremost, I would like to express my sincere gratitude to my PhD supervisors, Prof. Dario Colazzo and Dr. Elsa Negre from Université Paris-Dauphine - PSL University, for allowing me to carry out this thesis. Many thanks for your availability, your beneficial scientific pieces of advice and your helpful discussions. I would never have been able to improve my writings and presentations without your insightful comments and your endless patience.

I would also like to thank Julien Blaize, my advisor in Coheris, for his attentiveness and support throughout the years I spent in Coheris. Thank you for your precious assistance in overcoming many technical issues.
I thank Coheris company for the financial support of this thesis (CIFRE).

Likewise, I was delighted that Prof. Max Chevalier, from Université de Toulouse, and Dr. Armelle Brun, from Université de Lorraine accepted to review my thesis. I thank you both for your attentive reading of my thesis, and for your useful remarks plus your pleasant communication.

I would like to thank Prof. Jacky Akoka from CNAM and Dr. Florian Yger from Université Paris-Dauphine - PSL University, for their interest in my thesis and for accepting to be my jury members.

I thank the SPAD team members for their warm welcome and encouragement, as well as my PhD colleagues at LAMSADE for their friendship and support.

Finally, I cannot thank my family and friends enough for their unconditional support and patience in difficult times. I thank my beloved parents for their love and care, my sister Leyla, and my brother Mahdi for their kindness and encouragement. Not to mention, I am grateful to my husband, Saeed, for his patience, constant support and help during the challenges of doctoral studies and life. I am truly thankful to have you in my life. And, I embrace my little Jasmine whom I often had reluctantly to part from in order to concentrate on my work, but that today we can enjoy the result of these sacrifices.

Abstract

With the rise in volume of data from various sources, we have an increasing need of recommender systems, which provide a data filtering to help users to find appropriate information. To satisfy even more users' needs and generate more relevant recommendations, a new kind of recommender systems called Context-Aware Recommender System (CARS) integrates contextual information related to the users in their recommendation process. However there exists no unique definition for context. In this thesis we firstly identify relevant context factors for CARSs, to improve upon previous propositions, which can be used for a large spectrum of applications. Then we propose a new context representation and approach to integrate this kind of information into a recommender system. We make a relevant representation of the context, based on the influence of context on ratings, calculated using the Pearson Correlation Coefficient. We present a pre-filtering and a post-filtering context-aware recommender systems based on this representation. We propose a method to generate explanations for our context-aware recommendations. Also, we demonstrate that our approach can reduce the well-known sparsity problem of CARS and outperform state of the art approaches.

Résumé

Avec l'augmentation du volume de données produit par diverses sources, nous avons un besoin croissant de systèmes de recommandation, qui filtrent les données pour aider les utilisateurs à trouver l'information appropriée. Afin de satisfaire encore plus les besoins des utilisateurs et générer des recommandations plus pertinentes, un nouveau type de systèmes de recommandation, nommé système de recommandation contextuel (CARS), intègre les informations contextuelles des utilisateurs dans le processus de recommandation. Cependant, il n'existe toujours pas de définition unique du contexte. L'objectif de cette thèse est, dans un premier temps, d'identifier les facteurs de contexte pertinents pour les CARSs, afin d'améliorer les précédentes propositions de l'état de l'art, et pouvant être utilisés pour un large éventail d'applications. Ensuite, nous proposons une nouvelle représentation du contexte, ainsi qu'une approche pour intégrer ce type d'information dans un système de recommandation. Nous représentons le contexte en nous basant sur l'influence du contexte sur les scores donnés par les utilisateurs aux éléments, calculée à l'aide du Coefficient de Corrélation de Pearson. Ensuite nous filtrons les données à partir de ces représentations, afin de les intégrer dans le processus de recommandation. Nous présentons deux approches de recommandations contextuelles à base de pré-filtrage et post-filtrage. De plus, nous proposons une méthode pour générer des explications pour nos recommandations contextuelles. Par des expérimentations, nous démontrons que notre approche réduit la parcimonie, problématique bien connue des CARS, et peut également améliorer les performances de l'état de l'art.

Résumé Etendu

Les systèmes de recommandation traditionnels (basés sur des techniques à base de contenu et/ou de filtrage collaboratif) ont prouvé leur efficacité dans différents domaines [83], comme la musique, les films, les lieux d'intérêt, les articles de recherche, les cours en ligne, etc. Mais ils ont pour limite de ne pas tenir compte de la situation contextuelle dans laquelle se trouve l'utilisateur au moment où il veut utiliser la recommandation. En effet, ces informations peuvent influencer ses préférences pour les éléments recommandables [6]. Par exemple, pour choisir un film à regarder, l'utilisateur aura des préférences différentes s'il souhaite regarder le film avec un enfant ou avec son partenaire. Dans ce cas, un système de recommandation contextuel (CARS), intégrant les informations contextuelles de l'utilisateur dans son processus, peut fournir des recommandations plus pertinentes [4].
Dans cette thèse, nous proposons une approche de recommandation contextuelle, qui intègre les informations contextuelles des utilisateurs en les modélisant en fonction de leur influence sur les notes données par les utilisateurs aux éléments. Nous allons tout d'abord décrire notre module de *filtrage à base de corrélation (CBF)*, qui sera ensuite intégré dans le processus de recommandation sous forme d'approches pré- et post-filtrage.

Filtrage à base de corrélation

L'objectif principal de notre approche de filtrage est de transformer l'ensemble des données contextuelles initiales qui est multidimensionnel ($utilisateur \times item \times contexte \rightarrow note$) en un ensemble de données bidimensionnel ($utilisateur \times item \rightarrow note$) contenant uniquement les données relatives au contexte de l'utilisateur cible. En ce qui concerne l'état de l'art sur les CARS [40], notre approche est, dans un sens, plus centrée sur l'utilisateur, puisque nous proposons de calculer l'influence du contexte sur les notes en fonction des items ou des utilisateurs à l'aide du coefficient de corrélation de Pearson (PCC) [27] entre le contexte et les notes. Le PCC nous permet de saisir plus précisément cette influence, et donc de calculer des similarités plus précises entre les contextes, une étape importante dans notre processus de pré/post-filtrage. De plus, nous utilisons les caractéristiques statiques des items/utilisateurs (e.g. la catégorie d'un film, ou l'âge et le genre des utilisateurs) pour améliorer notre modèle.

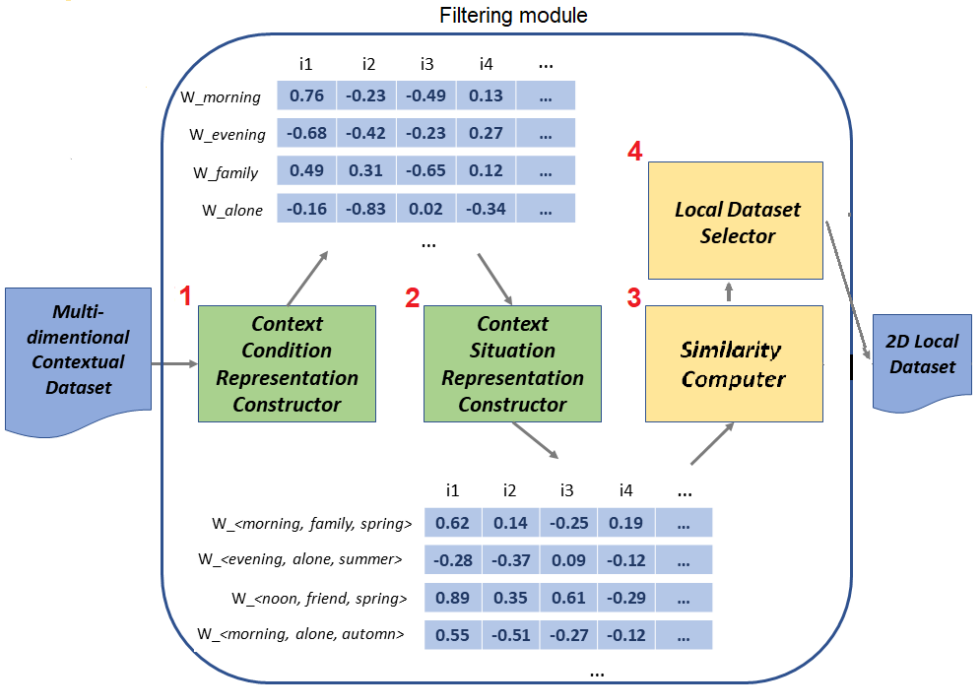


Figure 1: Chaîne de traitement de filtrage à base de corrélation

Méthodologie

L'ensemble du processus de *filtrage à base de corrélation (CBF)*, illustré dans la figure 1, peut être décomposé en quatre étapes : (1) représentation des conditions de contexte (valeur possible pour chaque facteur de contexte), (2) représentation des contextes, (3) identification des contextes similaires en calculant la similarité entre le contexte de l'utilisateur cible et les autres contextes, et (4) construction d'un ensemble de données bidimensionnel qui rassemble les notes données dans les contextes similaires au contexte de l'utilisateur cible. Notez que pour les opérations de certaines étapes, nous proposons différentes extensions :

Étape 1 : Comme dit précédemment, pour la recherche des contextes similaires, nous avons besoin d'une représentation pertinente du contexte. Le contexte de l'utilisateur est principalement exprimé par des données nominales (comme matin, printemps, heureux, etc.). Pour mesurer les similarités entre ces données nous pouvons faire appel à des ressources externes comme les ontologies. Cependant, ce type de ressources est en général spécifique à un domaine, et il est difficile de trouver une ontologie générique qui puisse être utilisée pour n'importe quel domaine d'application. Nous proposons donc une représentation numérique des contextes en fonction de leur influence sur les notes. Nous mesurons cette influence en calculant les coefficients de corrélation de Pearson (PCC) entre la variable note r, et chaque variable de condition de contexte c_j, avec $j \in [1, n]$, où n est le nombre total des variables de condition de contexte. Nous proposons la mesure de corrélation PCC (avec des valeurs comprises entre -1 et 1) car elle est beaucoup utilisée en statistique pour mesurer l'association entre deux variables, et cela correspond à ce

User	Item	Rating	Time	Companion	Season
U1	I1	3	morning	family	spring
U1	I2	1	evening	alone	summer
U2	I1	5	noon	friend	spring
U3	I6	4	morning	alone	autumn

(a) Matrice originale

User	Item	Rating	Time= Morning	Time= Noon	Time= Evening	Time= Night	Companion =Alone	Companion =Family	Companion =Friends	Season= Summer	Season= Winter	Season= Spring	Season= Autumn
U1	I1	3	1	0	0	0	0	1	0	0	0	1	0
U1	I2	1	0	0	1	0	1	0	0	1	0	0	0
U2	I1	5	0	1	0	0	0	0	1	0	0	1	0
U3	I6	4	1	0	0	0	1	0	0	0	0	0	1

(b) Matrice transformée

Table 1: Transformation de matrice de données

que nous recherchons, puisque nous souhaitons saisir l'influence des conditions de contexte sur les notes.

Dans un environnement contextuel, une observation est le croisement des variables utilisateur, item, notes et des différents facteurs de context (e.g. *type de jour, saison, lieu, social, etc*). Pour appliquer le PCC, nous transformons les facteurs de contexte en variables binaires (voir tableau 1). Donc $X_t = (u_t, i_t, r_t, c_{1t}, c_{2t}, ..., c_{nt})$ est la t-ième observation, qui représente la note r_t donnée par l'utilisateur u_t à l'item i_t dans le contexte $c_{1t}, c_{2t}, ..., c_{nt}$, où n est le nombre total des conditions de contexte, et $c_{pt} = 1$ signifie que la p-ième condition de contexte est présente dans le contexte de l'utilisateur, et $c_{pt} = 0$ signifie qu'elle ne l'est pas. Par exemple, dans l'exemple du tableau 1b, nous avons une notation de 1 à 5, où 1 signifie que l'utilisateur n'a pas aimé l'item, et 5 signifie qu'il l'a beaucoup apprécié. Donc la première ligne, l'observation X_1= *(U1, I1, 3, matin=1, midi=0, soir=0, nuit=0, seul=0, famille=1, amis=0, été=0, hiver=0, printemps=1, autumn=0)* signifie que *U1* a donné la note *3* à l'item *I1*, lorsqu'il a utilisé cette item avec sa *famille*, un *matin* de *printemps*.

- ***Extension 1.1 :*** Pour obtenir une valeur plus précise, nous calculons l'influence du contexte sur les notes en fonction de l'item ou de l'utilisateur.

  (a) basée sur les items : La raison pour laquelle l'influence basée sur les items pourrait être intéressante est que le contexte peut influencer les notes différemment, selon les items. Par exemple, dans le cas d'une recommandation de points d'intérêt, un temps neigeux aura une influence positive sur les centres de sports d'hiver, mais une influence négative sur les parcs naturels. C'est pourquoi il est important de calculer cette influence en fonction des points d'intérêt.

  Ainsi, la corrélation basée sur les items entre les notes et la condition de contexte c_j est calculée par l'équation 1.

$$w_{c_j i} = PCC_i(r, c_j) = \frac{\sum_{k \in K}(r_k - \overline{r_i})(c_{jk} - \overline{c_{ji}})}{\sqrt{\sum_{k \in K}(r_k - \overline{r_i})^2}\sqrt{\sum_{k \in K}(c_{jk} - \overline{c_{ji}})^2}} \quad (1)$$

Dans notre calcul de PCC, les sommes sont faites sur K, qui est l'ensemble des observations $X_k = (u_k, \mathbf{i}, r_k, c_{1k}, c_{2k}, ..., c_{nk})$ où l'item est égal à i. $\overline{r_i}$ représente la moyenne des notes données à l'item i, et $\overline{c_{ji}}$ représente la valeur moyenne de la condition de contexte c_j sur les observations où l'item est égal à i.

(b) basée sur les utilisateurs : L'étude de l'influence en fonction des utilisateurs est également intéressante. En effet, on peut dire que l'influence du contexte sur les notes peut également dépendre des utilisateurs, et qu'elle sera différente d'un utilisateur à un autre. Par exemple, une "personne très liée à sa famille" aimerait pratiquer des activités avec sa famille, alors qu'une autre personne préférerait pratiquer des activités entre amis. Le contexte social influence donc différemment ces deux personnes.

Dans ce cas, la corrélation basée sur les utilisateurs entre les notes r et la condition de contexte c_j est calculée par l'équation 2.

$$w_{c_j u} = PCC_u(r, c_j) = \frac{\sum_{k \in K}(r_k - \overline{r_u})(c_{jk} - \overline{c_{ju}})}{\sqrt{\sum_{k \in K}(r_k - \overline{r_u})^2}\sqrt{\sum_{k \in K}(c_{jk} - \overline{c_{ju}})^2}} \tag{2}$$

où K est l'ensemble des observations $X_k = (\mathbf{u}, i_k, r_k, c_{1k}, c_{2k}, ..., c_{nk})$ de l'utilisateur u. $\overline{r_u}$ est la moyenne des notes données par l'utilisateur u, tandis que $\overline{c_{ju}}$ est la valeur moyenne de la condition de contexte c_j sur les observations de l'utilisateur u.

- ***Extension 1.2 :*** Sur la base des explications ci-dessus, nous pouvons construire une représentation vectorielle pour chaque condition de contexte, par l'une des deux méthodes suivantes :

  (a) non-groupé ("not-clustered") : La taille du vecteur de représentation des conditions de contexte est le nombre total d'items ou d'utilisateurs, et les valeurs de ce vecteur (entre -1 et 1) sont égales aux valeurs de PCC basées sur les items/utilisateurs entre le vecteur note et le vecteur condition du contexte.

  (b) groupé ("clustered") : Dans les applications de recommandation, le nombre total d'items/utilisateurs est souvent très élevé, et le calcul de la corrélation demande donc beaucoup de ressources. Afin de limiter ce coût de calcul, nous proposons de regrouper les items/utilisateurs en un nombre limité de groupes, et de calculer l'influence en fonction des groupes d'items/utilisateurs. De plus, comme le nombre de notes connues est souvent très faible, la stratégie de regroupement pourrait aider le PCC à saisir plus précisément la corrélation entre les conditions du contexte et les notes. En effet, plus les données disponibles pour deux variables sont importantes, meilleures sont les corrélations obtenues à l'aide du PCC. En regroupant les items/utilisateurs, nous regroupons les notes de plusieurs items/utilisateurs en une seule variable, et le PCC sera donc calculé sur des variables plus riches.

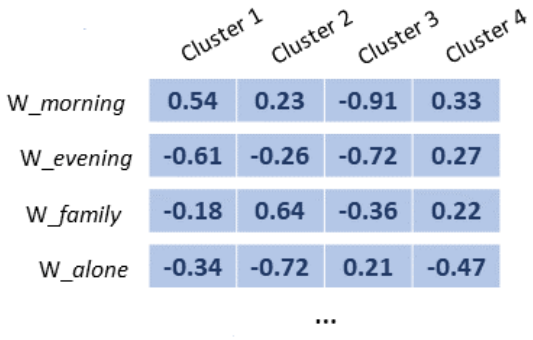


Figure 2: Exemples de représentations de conditions de contexte basées sur des clusters (Step 1)

La figure 2 illustre quelques exemples de représentations de conditions de contexte basées sur des clusters. Dans cet exemple, nous avons regroupé les items en 4 groupes différents. Dans l'exemple de représentation *matin ("morning")*, la valeur 0.54 illustre l'influence positive de *matin* sur les notes du premier groupe d'items (calculées par PCC), tandis que la valeur -0.91 fait référence à sa forte influence négative sur les notes du troisième groupe d'items.

Étape 2 : Nous pouvons à présent représenter chaque contexte en fonction des conditions de contexte qui le composent (étape 2 de la figure 1).

- ***Extension 2.1 :*** Nous proposons ici deux façons différentes de construire cette représentation :

  (a) agrégation : Chaque contexte peut être représenté par un vecteur dont les valeurs sont égales à l'agrégation des valeurs de ses conditions de contexte (méthode également utilisée par Codina et al. dans [40]). La figure 3 illustre quelques exemples de représentations de contexte basées sur la technique d'agrégation. Par exemple, la valeur 0.25 du premier contexte $< matin, famille, printemps >$ ("$<$ *morning*, *family*, *spring* $>$") correspond à l'influence de ce contexte sur les notes données aux items du premier cluster, et est égale à la valeur moyenne des trois valeurs correspondantes (pour le cluster 1) des représentations vectorielles de *matin*, *famille* et *printemps* (dans la figure 2).

  (b) concaténation : Par une analyse plus approfondie de la technique d'agrégation proposée par Codina et al. dans [40], nous avons réalisé que dans certains cas, cette agrégation peut neutraliser les influences des conditions de contexte. Expliquons ce fait par un exemple : imaginons qu'un contexte soit composé de trois facteurs *temps*, *compagnon* et *saison*, et que l'on veuille calculer la similarité entre deux contextes C1 : $< matin, famille, printemps >$ et C2 : $< soir, seul, été >$. Supposons que le calcul du PCC de chaque condition de contexte pour le cluster 1, donne une

Figure 3: Exemples de représentation de la situation contextuelle par agrégation (Étape 2)

Figure 4: Exemple de la représentation de la situation contextuelle $W_{\langle morning,family,spring\rangle}$ par concaténation (Étape 2)

grande valeur négative (proche de -1) pour *matin* et *seul*, une grande valeur positive (proche de 1) pour *famille* et *soir* et une valeur neutre (proche de 0) pour *printemps* et *été*. Supposons que ce schéma se répète pour les autres clusters. Dans ce cas, l'agrégation des valeurs des conditions de contexte pour chaque cluster (pour C1 : $(-1+1+0)/3 = 0$ et pour C2 : $(1+(-1)+0)/3 = 0$), donne des valeurs similaires pour les deux contextes $C1$ et $C2$. Ainsi, contrairement à ce que l'on attendait, nous obtenons une grande similarité. Pour éviter cela, nous proposons de représenter le contexte par un vecteur plus large, en concaténant les vecteurs de ces conditions de contexte, au lieu d'agréger leurs valeurs (voir Figure 4).

Étape 3 : Maintenant que nous avons représenté chaque contexte, nous pouvons trouver les contextes les plus similaires au contexte cible s^* en calculant la similarité entre chaque contexte possible s et le contexte cible s^*.

- ***Extension 3.1 :*** Cette similarité peut être calculée par différentes mesures. Nous proposons ici les deux mesures suivantes :

  (a) Similarité cosinus : on peut obtenir la similarité entre les représentations vectorielles $\overrightarrow{w_s}$ et $\overrightarrow{w_s*}$ par la mesure de similarité cosinus illustrée par l'équation 3, où d est la dimension du vecteur $\overrightarrow{w_s}$.

$$sim(s, s^*) = cosine(\overrightarrow{w_s}, \overrightarrow{w_{s^*}}) = \frac{w_s^T w_{s^*}}{\sqrt{\sum_{i=0}^{d} w_{s,i}^2}\sqrt{\sum_{i=0}^{d} w_{s^*,i}^2}} \tag{3}$$

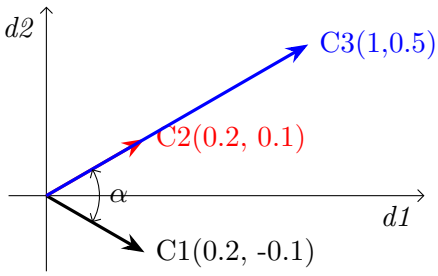


Figure 5: Illustration géométrique de la similarité cosinus entre différents vecteurs

La similarité cosinus est très utilisée dans les systèmes de recommandation en raison de sa rapidité de calcul et de son efficacité à gérer la parcimonie. Mais cette mesure n'est pas sensible à l'échelle et dans notre application, cette propriété pourrait affecter les résultats attendus. Voici un exemple pour expliquer ce fait : considérons trois contextes différents $C1[0.2, -0.1]$, $C2[0.2, 0.1]$ et $C3[1, 0.5]$ (les deux dimensions des contextes sont choisies pour rendre l'exemple plus simple). Nous pouvons voir que les deux contextes $C1$ et $C2$ ont des petites valeurs comparables (faisant référence à leur influence sur les notes), contrairement à $C3$ qui a des valeurs beaucoup plus importantes. En théorie, nous nous attendons donc à une plus grande similarité entre $C1$ et $C2$, et à une plus petite entre $C1$ et $C3$. Mais la mesure de similarité cosinus, nous donne deux similarités exactement identiques, car $C3$ est un vecteur mis à l'échelle de $C1$ ($C3 = 5 \cdot C1$). En effet, la similarité cosinus entre deux vecteurs correspond à l'angle entre eux. Comme vous pouvez le voir dans la figure 5 qui est une illustration géométrique de notre exemple, l'angle α entre $C1$ et $C2$ correspond également à l'angle entre $C1$ et $C3$.

Ainsi, dans les cas où les différentes échelles modifient la signification du vecteur et doivent être prises en compte, nous devons garder à l'esprit que la mesure de similarité cosinus ignore cette différence d'échelle. Dans notre approche CBF, les valeurs des représentations des vecteurs de contexte illustrent l'influence des conditions de contexte sur les notes. Ainsi, une petite valeur comme 0.2 signifie une petite influence et une grande valeur 1 signifie une influence maximale. Comme nous voulons saisir cette différence dans notre calcul de similarité, la mesure de similarité cosinus n'ai pas forcément la meilleure option.

(b) Similarité euclidienne : Pour éviter le problème de la similarité cosinus, nous proposons de calculer la similarité entre deux contextes s et s^* par la similarité euclidienne qui correspond à l'opposé de la distance euclidienne entre leurs vecteurs (Equation 4) :

$$sim(s, s^*) = \frac{1}{euclidean(\overrightarrow{w_s}, \overrightarrow{w_{s^*}})} = \frac{1}{\sqrt{\sum_{i=0}^{d}(w_{s,i} - w_{s^*,i})^2}} \tag{4}$$

Étape 4 : Nous pouvons à présent transformer l'ensemble des données contextuelles multidimensionnelles en un ensemble de données bidimensionel. Pour cela, nous sélectionnons les notes données dans les contextes similaires, identifiés à l'étape 3, et nous les rassemblons dans une matrice bidimensionel.

Alternative pour le calcul de corrélation : La figure 1 et les étapes expliquées ci-dessus ont détaillé le processus de filtrage de notre approche *CBF*. Notez que dans ce modèle, dans la première étape, nous avons utilisé le PCC pour calculer l'influence du contexte sur les notes pour représenter le contexte. Au lieu du PCC, nous pouvons penser à d'autres méthodes pour saisir cette influence. Une autre proposition pourrait être de modéliser cette influence par la différence entre la moyenne des notes données lorsque la condition de contexte est présente et les notes données lorsque celle-ci est absente. L'équation 5 illustre cette technique que nous avons appelée technique de *déviation moyenne*.

$$w_{c_j i} = \frac{\bar{r}_{ic_j=1} - \bar{r}_{ic_j=0}}{r_{max} - r_{min}} \tag{5}$$

Pour normaliser cette valeur afin qu'elle soit comprise entre [-1, 1], nous l'avons divisée par la différence entre les valeurs maximales et minimales possibles des notes.

Pertinence du contexte

L'intégration d'informations contextuelles vise à améliorer les performances du système de recommandation [77]. En théorie, on peut supposer que l'ensemble complet des informations contextuelles est pertinent et que tous les facteurs de contexte sont d'importance égale. Cependant, selon l'application, certains facteurs de contexte pourraient avoir un impact plus important que d'autres. La prise en compte de certains peut même introduire plus de bruit que de la vraie information. Par exemple, dans le cas de la recommandation de musique, l'activité de l'utilisateur a plus d'impact sur ses préférences que sa géo-localisation. Dans ce même exemple, un facteur de contexte comme la saison, qui n'a pas vraiment d'impact sur les préférences musicales de l'utilisateur, peut introduire plus de bruit que de la vraie information dans le processus de recommandation. Il est donc important d'identifier les facteurs de contexte pertinents et/ou considérer leur degré d'impact dans le processus de recommandation afin d'éviter une perte de performance. De plus, nous devons noter que la collecte automatique de certains facteurs de contexte (comme l'humeur de l'utilisateur ou son contexte social) est encore presque impossible et qu'ils doivent être renseignés directement par l'utilisateur. Ainsi, cette identification des facteurs de contexte pertinents permettrait également de minimiser l'effort demandé à l'utilisateur pour spécifier son contexte, ce qui n'est pas négligeable [34].

Dans cette thèse, nous proposons différentes méthodes pour considérer la pertinence du contexte dans sa représentation. Nous les avons regroupées en trois catégories d'approches : (a) les approches par *pondération*, où nous calculons un poids w_{f_i} pour chaque facteur de contexte f_i et l'utilisons pour pondérer le vecteur de condition correspondant dans la représentation du contexte, (b) les approches par *filtrage*, qui sont une façon plus stricte de considérer la pertinence du contexte, où dans une configuration binaire nous ignorons les facteurs de contexte non pertinents et excluons leurs vecteurs de la représentation du contexte. Et (c) les approches *hybrides*, qui sont une combinaison des deux premières approches, où comme l'approche par *filtrage*, nous identifions les facteurs de contexte pertinents pour ne garder que ceux-ci, et comme l'approche par *pondération* nous pondérons leurs vecteurs correspondants dans la représentation du contexte.

Recommandation contextuelle à base de corrélation

Les informations contextuelles des utilisateurs peuvent être intégrées dans le processus de recommandation de trois manières différentes : Les méthodes *pré-filtrage*, *post-filtrage* et *contextual modeling* [6] . Dans cette thèse, nous présentons deux approches différentes pour l'intégration de notre module de *filtrage basé sur la corrélation* dans un processus de recommandation afin de produire des recommandations contextuelles. Nous proposons une première configuration par pré-filtrage nommée *Correlation-Based Pre-Filtering (CBPF)*, et une seconde nommée *Correlation-Based Post-Filtering (CBPoF)*.
Les approches par pré-filtrage constituent une classe particulière des systèmes de recommandation contextuels (CARS) basés sur l'idée de filtrer les données contextuelles de manière à ajuster les données d'entrée d'un système de recommandation (traditionnel) afin d'améliorer son efficacité. Alors que les approches par post-filtrage appliquent d'abord une technique de recommandation traditionnelle sur les données en ignorant le contexte, puis contextualisent la liste de recommandations résultante en filtrant ou en réordonnant les items de la liste.

Pré-filtrage à base de corrélation

Un problème de recommandation est souvent considéré comme un problème de remplissage de matrice ou tenseur. Un système de recommandation estime d'abord les notes manquantes, puis recommande à chaque utilisateur les items avec les notes estimées les plus élevées. Dans le cas des approaches de recommandation contextuelles par pré-filtrage, nous intégrons les informations contextuelles de l'utilisateur dans la phase d'estimation des notes manquantes. Notre approche de pré-filtrage *à base de corrélation*, comme l'approche de pré-filtrage *basée sur la réduction* [4], fait l'hypothèse qu'un utilisateur évaluera un item de manière similaire dans deux contextes similaires. Sur la base de cette hypothèse, pour recommander un item à un utilisateur dans un

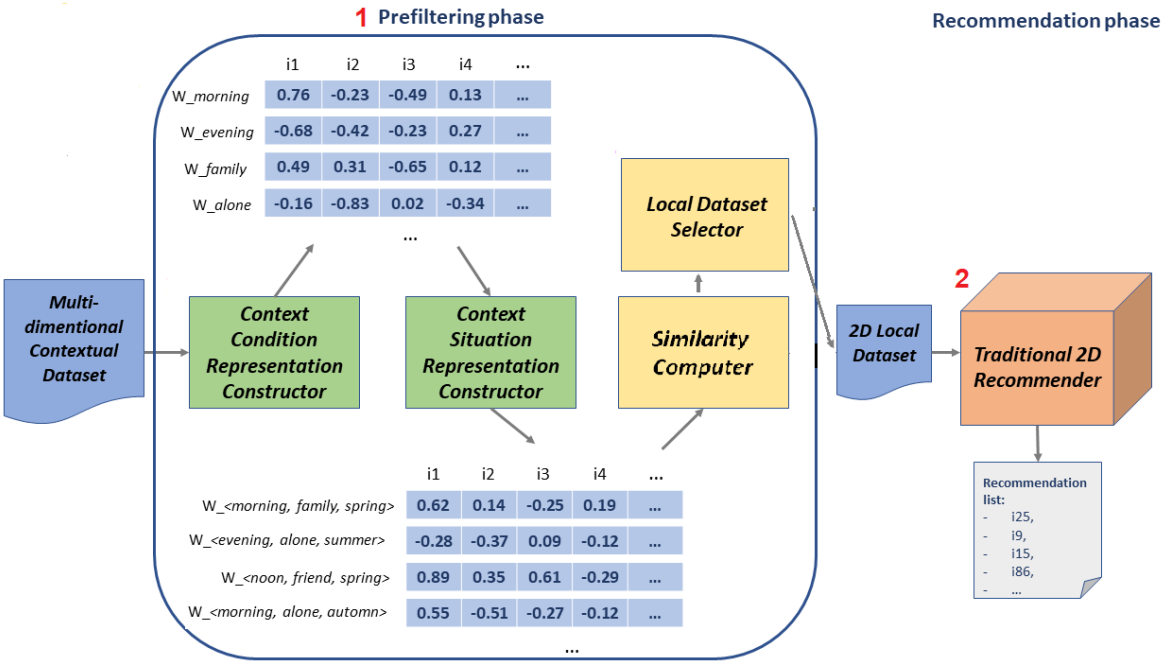


Figure 6: Chaîne de traitement CBPF

contexte spécifique, nous identifions les notes données dans les contextes similaires à ce contexte spécifique, et appliquons une technique de recommandation traditionnelle bidimensionnelle sur cette sélection. Pour identifier les contextes similaires nous utilisons notre module de *filtrage à base de corrélation (CBF)*. Nous proposons une configuration par pré-filtrage de ce module, que nous avons nommé *Correlation-Based Pre-Filtering (CBPF)*.

Méthodologie

La figure 6 illustre les deux principales étapes de notre approche de recommandation contextuelle par pré-filtrage basée sur la corrélation (CBPF).

Étape 1 : Nous transformons l'ensemble des données contextuelles multidimensionnelles en un ensemble de données bidimensionnel basé sur le contexte de l'utilisateur cible, en appliquant le module de filtrage à base de corrélation. Ce module de filtrage permet de réduire la parcimonie de l'ensemble de données contextuelles initiales, qui a un impact positif sur la performance du système de recommandations.
Étape 2 : Nous appliquons ensuite une technique de recommandation traditionnelle bidimensionnelle [50] sur cette sélection de notes, afin d'obtenir des recommandations contextuelles. Notre approche étant générique, tout type de technique de recommandation bidimensionnelle peut être appliqué, mais nous proposons ici la technique *BiasedMF* [55], qui est une technique de décomposition de matrice très performante.

Notre approche présente deux principaux avantages techniques : a) elle peut être facilement "branchée" à n'importe quel système de recommandation bidimensionnel existant : en effet, cette caractéristique permet aux entreprises, qui souhaitent bénéficier des informations contextuelles disponibles dans leur processus de recommandation, de réutiliser leur système de recommandation existant et de "brancher" le module de filtrage en amont, b) notre approche est également configurable : nous proposons plusieurs alternatives pour les différentes parties de l'algorithme (différentes représentations du contexte, mesures de similarité, algorithmes de clustering, techniques de corrélation, etc.). En fonction de la nature des données et des ressources disponibles, nous pouvons appliquer différentes configurations.

Post-filtrage à base de corrélation

Comme mentionné précédemment, les informations contextuelles des utilisateurs peuvent être intégrées dans le processus de recommandation de trois manières différentes : Les méthodes *pré-filtrage*, *post-filtrage* et *contextual modeling*. Étant dans un contexte industriel, où un système de recommandation traditionnel existe déjà, nous nous sommes tout d'abord intéressés à la technique de *pré-filtrage*, qui est beaucoup utilisée dans la littérature, et avons proposé une approche de *recommandation contextuelle par pré-filtrage basée sur la corrélation (CBPF)* entre les contextes et les notes. Ensuite nous avons proposé une adaptation post-filtrage de notre approche, appelée *Correlation-Based Post-Filtering (CBPoF)*. Notre principale motivation est le nombre très restreint de recherches sur la comparaison des approches par pré- et post-filtrage dans les CARS.

Méthodologie

La figure 7 illustre la chaîne de traitement de notre approche CBPoF pour recommander une liste d'items à l'utilisateur u^* qui se trouve dans le contexte s^*. Comme toutes les approches par post-filtrage, nous transformons d'abord l'ensemble des données multidimensionnelles en un ensemble de données bidimensionnel en ignorant les informations contextuelles. Ensuite, nous appliquons une technique de recommandation traditionnelle à cet ensemble de données sans contexte, pour lequel nous obtenons des prédictions de notes pour l'utilisateur u^* ($\hat{r}_{u^*,i}$). Nous proposons à présent d'identifier les voisins contextuels de l'utilisateur cible u^*, que nous appelons G. G est l'ensemble des utilisateurs similaires à u^* dans son contexte s^*. Ce voisinage est identifié en calculant la similarité cosinus entre l'utilisateur cible u^* et tous les utilisateurs qui ont notés des items dans des contextes similaires à s^*. Dans ce processus, l'identification des contextes similaires est effectuée sur la base de notre approche de *filtrage basé sur la corrélation (CBF)*, également utilisée dans *CBPF*. Enfin, nous contextualisons les notes prédites en fonction de la distribution des notes données par le voisinage de G. Et nous proposons une liste de recommandations basées sur ces éstimations de notes contextuelles ($\hat{r}_{u^*,i,s^*}$).

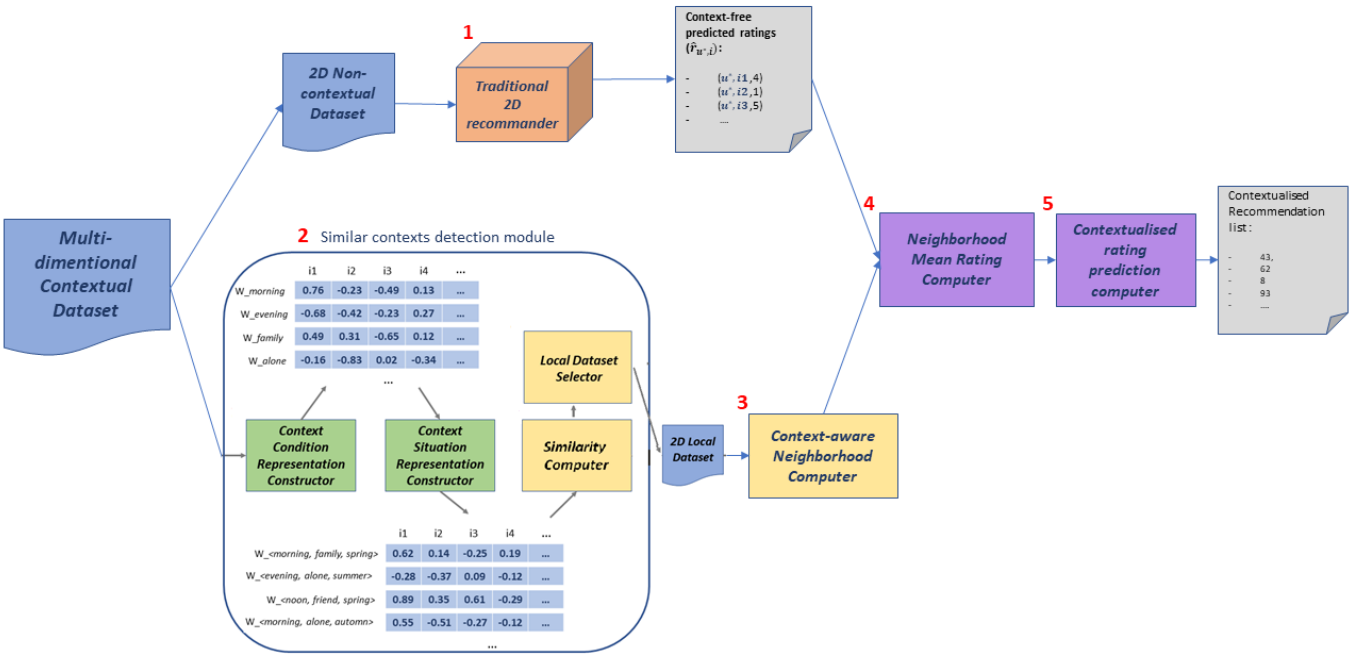


Figure 7: Chaîne de traitement CBPoF pour recommander un item à l'utilisateur u^* dans le contexte s^*

L'équation 6 illustre la combinaison convexe utilisée pour contextualiser les prédictions de notes non-contextuelle $\hat{r}_{u^*,i}$. La prédiction de note de l'item i par l'utilisateur u^* dans le contexte s^* ($\hat{r}_{u^*,i,s^*}$) est égale à la somme pondérée de la prédiction de note non-contextuelle $\hat{r}_{u^*,i}$ et de la moyenne des notes données par les voisins de l'utilisateur u^*.

$$\hat{r}_{u^*,i,s^*} = \alpha \times \hat{r}_{u^*,i} + (1-\alpha) \times \frac{\sum_{g \in G} r_{g,i}}{|G|} \tag{6}$$

Un système de recommandation contextuelle doit jongler entre deux paramètres : la personnalisation et la contextualisation de ses prédictions. Le coefficient α qui a une valeur entre 0 et 1, nous permet de paramétrer ces deux aspets : en augmentant sa valeur, nous augmentons l'impact de la prédiction non-contextuelle et donc la partie personnalisation, tandis qu'en la diminuant, nous donnons plus d'importance à la partie contextualisation. Dans le premier cas, les recommandations sont plus influencées par les similarités des préférences avec les autres utilisateurs, tandis que dans le second, nous mettons l'accent sur l'impact du contexte de l'utilisateur sur ses préférences.

Explications pour les recommandations contextuelles

La possibilité de fournir des explications pour les recommandations faites aux utilisateurs est un sujet qui a beaucoup attiré l'attention de la communauté des systèmes de recommandations. Une explication peut clarifier les raisons pour lesquelles un item spécifique est proposé. Comme l'indiquent Tintarev et Masthoff dans [96], proposer une explication à l'utilisateur peut présenter de multiples avantages : *transparence*, en expliquant le fonctionnement du système; *scrutabilité*, en permettant aux utilisateurs d'indiquer au système quand il est mauvais; *confiance*, en augmentant la confiance des utilisateurs envers le système; *efficacité*, en aidant les utilisateurs à prendre de bonnes décisions; *suasivité*, en convainquant les utilisateurs d'essayer ou d'acheter un item; *rendement*, en aidant les utilisateurs à prendre des décisions plus rapidement; et *satisfaction*, en augmentant la facilité ou le plaisir d'utilisation d'un item.

Dans les systèmes de recommmandations traditionnels, les explications sont souvent basées sur le contenu (e.g. "Nous vous recommandons A parce que vous avez aimé B"), sur les préférences (e.g. "Nous vous proposons A en fonction de vos préférences") ou sur la collaboration (e.g. "Les gens qui aiment A ont également aimé B"). Ce sujet n'est pas encore bien exploré dans le domaine des systèmes de recommandations contextuelles. À notre connaissance, seuls quelques travaux ont été réalisés sur ce sujet ([19, 60, 105]). Néanmoins, la prise en compte du contexte de l'utilisateur dans le processus de recommandation permet non seulement d'améliorer la qualité des recommandations, mais peut également être utilisée pour expliquer pourquoi un item est recommandé [19].

Dans cette thèse, nous proposons une méthode pour générer des explications pour les items que nos approches *CBPF* et *CBPoF* recommandent, en nous basant sur le résultat de notre module de detection de pertinence des facteurs de contexte :
Supposons que notre CARS recommande l'item i à l'utilisateur u^* dans le contexte s^* : $c_1, c_2, c_3, ..., c_m$ où m est le nombre total des facteurs de contexte. Dans le processus de recommandation, nous identifions les facteurs de contexte les plus pertinents, sur la base de notre module de détection de pertinence du contexte (par les techniques de *pondération* ou de *filtrage*). Nous proposons de générer l'explication pour l'item recommandé, en utilisant soit le facteur de contexte le plus impactant identifié par la technique de *pondération* (le facteur de contexte ayant le poids le plus élevé), soit l'ensemble des facteurs de contexte pertinents identifiés par la technique de *filtrage*.

Exemple : Pour un exemple de recommandations contextuelles de films où nous avons trois facteurs de contexte *temps, contexte social* et *saison*, les poids obtenus par la technique de pondération de notre module de détection de pertinence du contexte, pour ces facteurs de

contexte sont respectivement de 0.37, 0.55 et 0.31. Ainsi, dans ce cas, le facteur de contexte qui a le plus grand impact positif sur la prédiction de la note du film i est le contexte *social*. Supposons que la valeur du contexte *social* de l'utilisateur u^* soit $\ll$ *en famille* $\gg$. Par conséquent, l'explication que nous pouvons apportée pour l'item recommandé i ressemblerait à: $\ll$ Nous vous proposons le film i qui est intéressant à regarder en famille $\gg$.

Finalement, dans cette thèse nous avons proposé deux approches de recommandation contextuelles, qui intègrent les informations sur le contexte des utilisateurs par un module de filtrage à base de corrélation. Nos experimentations sur quatre jeux de données différents montent les bonnes performances de nos approches. Ce travail ouvre la voie vers de nombreuse perspectives, comme son adaption aux données implicites ou à la recommandation pour groupe d'utilisateurs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The available data and information on the web is becoming increasingly important while the users can easily be overwhelmed by these data and information. This is why we need strong filtering techniques to retrieve the appropriate information. One of these techniques is that based on recommendation. A Recommender System (RS) proposes items that can potentially be of interest for the user. Traditional recommender systems, which are essentially based on users' ratings on items, represented by a two-dimensional matrix ($user \times item \rightarrow rating$), have proven their reliability through the years, and are adopted by many platforms and companies. However, they had the limitation of not taking into account the contextual information of the users. In fact, the context in which a user is, at the moment she wants to use the item, can influence her preferences and choices. As an example, when choosing a movie to watch, the user will have different preferences depending on whether she wants to watch the movie with her kid or with her partner. Riboni and Bettini in [82] show a correlation between the user behavior and her context, which explains the importance of integrating the user context in the recommendation process. In these recent years a new family of recommendation approaches has emerged called *context-aware recommendation*. Such approaches try to improve the relevance of recommendations by adding information about the actual context of the user.

1.2 Research Motivations

Recently, by the growth of connected devices and Internet of Things (IoT), users' contextual information is more and more available and used in different information systems. So it would be interesting for many companies which already have a RS in production, to improve their recommendations by collecting this kind of additional information and integrating it into their RS. Thus, the very first question that we try to answer in this thesis is the following: ***what are the relevant and meaningful pieces of contextual information to collect so as to***

be used in CARSs (Context-Aware Recommender Systems)? In fact the importance of taking into account contextual information about the users is now evident, but still there exists no universal definition of it in the literature. Many researchers, from different domains, have worked on this concept, and proposed different categorizations of context factors from different points of view, but we can hardly find a complete and generic proposition that could be applied universally to any recommendation domains (e.g. music, movie, etc). Therefore, our first objective is to study the notion of context, and propose a generic and hierarchical categorization of context factors for context-aware recommendation, in order to be applied to a large spectrum of application domains.

After identifying and gathering interesting contextual information, the second question is ***how to integrate this kind of information into the RS with minimal implementation cost and no need of external data sources?*** In fact in the industrial world, financially, it is important to upgrade existing information systems while re-using them instead of re-implementing the whole systems from scratch. Furthermore, the accessibility to external information resources (like ontologies) could be difficult. As a result, in the case of context-aware recommendation, a data-driven method which could be plugged in to the existing traditional RS is more appreciated. So, to recommend an item to a user in a specific context, we propose a pre-filtering approach that uses contextual information to filter the rating data based on the actual context of the user, and apply a traditional recommendation technique only on this selection of ratings. Indeed by this filtering phase we transform the multidimensional initial contextual dataset ($user \times item \times context \rightarrow rating$) to a 2D dataset ($user \times item \rightarrow rating$) based on the user context, which then allows us to apply any kind of traditional recommender system on this reduced dataset.

This filtering can be done in several ways: one choice could be to select only ratings done in contexts which match exactly the actual user context (exact pre-filtering [4]). The problem of this strict filtering is that due to the small available number of ratings in each context situation[1], the resulting rating matrix will be extremely sparse, and recommendations for only restricted number of users/items could be produced. Indeed one of the main challenges in the domain of recommender systems in general is the data-sparsity due to the very large number of users and items in real-world applications. This limitation is even more pronounced for context-aware recommender systems, where the dimensions of the utility matrix are augmented by the context, especially when we have fine granularity in contextual information. This filtering strategy, not only does not overcome this challenge, but emphasizes it even more. The opposite filtering strategy to overcome this sparsity problem would be to consider the ratings done in all contextual situations, but this is as if we ignore the contextual information.

So the third question is ***what is the optimal strategy to filter ratings, without accentuating the sparsity problem and in order to be applicable to any recommendation***

[1]Please note that in thorough this document *context* and *context situation* are used as synonyms.

domain? An idea could be to find the most similar contexts to the actual context of the target user, and select ratings done in these similar contexts. The user context is mostly expressed by nominal data (e.g. morning, spring, happy, etc.). So, the similarities among contexts could be found by the help of some external resources like ontologies. But in practice, these kind of resources (especially in the case of ontologies) are very hard to be gathered in a generic way, and are mostly domain-specific. Thus, to be able to find similar contexts, without the need of such external resources and in a generic way, we need a significant numerical representation of the context. So now, the main question is ***how to represent the context in order to be able to find the proper similarities between each pair of contexts, which lead us to user-centric recommendations?*** We answer this question by proposing a new data-driven context-aware recommendation approach, called *Correlation-Based Pre-Filtering (CBPF)*, in order to efficiently integrate the contextual information of the users into the recommendation process, by modeling it with item/user-based influence of context on ratings. Our approach is in a sense, more user-centric, as we propose to model this influence based on the item- or user-based Pearson Correlation Coefficient (PCC) [27] between context and ratings. The distinctive feature of using PCC allows us to catch more precisely the influence of context on ratings, and so to compute more accurate similarities between contexts, which is a crucial point in our pre-filtering process. In addition, we use content information about items/users to improve our model, like, for instance, the category of a film or the age or gender of users.

In the context representation, usually all context factors are integrated into the context representation in the same way. However in real world applications, depending on the application domain, some context factors could have more impact on users' choices than others. Thus, the next question is: ***how can we efficiently take into account the impact degree of different context factors in the recommendation process?*** Toward this end, we propose a hybrid method to consider the context factors relevances, which filter irrelevant context factors and weight relevant ones in the context vector representations. Evaluations show the positive effect of this method on the recommendation performances.

In our *CBPF* approach, we plug a correlation-based filtering module before a traditional RS in a pre-filtering configuration. Our last question is the following: ***is it possible to adapt this approach in a post-filtering configuration? And how will be the performances comparison between the pre-filtering and the post-filtering?*** There exists three families of approach for CARS: *pre-filtering*, *post-filtering* and *contextual modeling* approaches. Contrary to *contextual modeling* approaches where the whole recommendation process have to be implemented from scratch, the *pre-filtering* and *post-filtering* methods have the advantage of re-using an existing traditional RS, and plug the filtering module to it. We present our *correlation-based pre-filtering* approach, and propose a *post-filtering* adaptation of it, that we named *correlation-based post-filtering (CBPoF)*. As there is very few research on the comparison of pre- and post-filtering approaches in CARS, we try to compare these two families of

approaches based on our *correlation-based filtering* proposition.

1.3 Summary of Contributions

This section describes the main contributions of this thesis to the context-aware recommender systems field:

1. In this thesis, we first study the notion of *context*, and propose a generic and hierarchic categorization of different context factors for context-aware recommender systems, in order to be applied to a large spectrum of application domains (like music, movies, news, etc.).

2. We propose a novel context-aware recommendation algorithm, based on a *correlation-based filtering* module:

   - We present a *pre-filtering* and a *post-filtering* adaptation of this approach. Indeed we can easily plug our *correlation-based filtering* module in a pre-filtering configuration (*CBPF*), as well as a post-filtering one (*CBPoF*).
   - This algorithm is data-driven, with no need of external resources like ontologies.
   - The algorithm is parametric: we design a generic algorithm, where we can specify different types of measures and techniques, depending on the data and available resources. In our experimental analysis, we suggest some choices based on the characteristics of the data.
   - Our propositions try to provide answers to two well-known challenges of the context-aware recommender systems, sparsity and explanations: besides the sparsity reduction that our *correlation-based filtering* module achieves in our *CBPF* approach, we are able to generate explanations for recommendations based on our context relevance detector.
   - Also, in this thesis, we guarantee the reproducibility of our approach by exposing a detailed pseudo-code of it.

3. This thesis is an industrial CIFRE [2] thesis, funded by ANRT [3] and the Coheris [4] company. One of the objectives of this thesis is the integration of the proposed context-aware recommendation approach into the Analytics and Artificial Intelligence (AAI) platform of Coheris.

1.4 List of Publications

Publications of the contributions of this work are the followings:

[2] Convention Industrielle de Formation par la Recherche
[3] Association Nationale Recherche Technologie: http://www.anrt.asso.fr
[4] https://www.coheris.com

- Z. Vahidi Ferdousi, E. Negre and D. Colazzo. (Poster) Context-Aware Recommender Systems: An Overview. In *MATHIAS 2016, International scientific conference organized by TOTAL Research & Development,* 2016.
- Z. Vahidi Ferdousi, E. Negre and D. Colazzo. Context Factors in Context-Aware Recommender Systems. In *AISR 2017: Atelier interdisciplinaire sur les systèmes de recommandation,* 2017. [101]
- Z. Vahidi Ferdousi, D. Colazzo and E. Negre. Correlation-based pre-filtering for context-aware recommendation. In *Proceedings of the 16th IEEE International Conference on Pervasive Computing and Communications Workshops (CoMoRea).* IEEE, 2018. [99]
- Z. Vahidi Ferdousi, D. Colazzo, E. Negre. Recommandations contextuelles à base de correlations. In *JdS 2018: Journées de Statistique,* 2018. [100]
- Z. Vahidi Ferdousi, D. Colazzo, E. Negre. CBPF: leveraging context and content information for better recommendations. *ADMA 2018: International Conference on Advanced Data Mining and Applications,* 2018. [98][5]

1.5 Organization of the Thesis

In addition to this introductory chapter, this document is organized as follows:

- **Chapter 2** presents the related work of the subject, organized in three main parts: firstly, we present the notion of context, its different factors and context modeling approaches. Secondly, we discuss about recommender systems, by presenting the major recommendation approaches proposed in the literature: content-based, collaborative filtering and hybrid approaches, but also the similarity measures used in these approaches. And finally, we present context-aware recommender systems, the main approaches: pre-filtering, post-filtering and contextual modeling, and describe different existing methods. In the following chapters, some of the methods presented here are used for performance comparisons.
- **Chapter 3** details the background material used within the thesis. We describe some statistical models, machine learning and recommendation techniques used in our project.
- **Chapter 4** presents a new, hierarchical and generic categorization of context factors proposed for context-aware recommender systems.
- **Chapter 5** proposes *Correlation-Based Filtering* approach, a filtering module that aims to transform the multi-dimensional contextual dataset to a 2D dataset, based on the influence of context on ratings. This filtering module will be the core of a pre-filtering or post-filtering context-aware recommender system.

[5]Please note that the last contribution (CBPoF) is a post-ADMA work which is not yet published.

- **Chapter 6** describes *Correlation-Based Pre-Filtering (CBPF)*, the proposed context-aware recommendation approach that alleviates the data-sparsity limitation of CARS by modeling a data-driven representation of context and integrating it in the pre-filtering phase of recommendation. In this chapter, we also propose a post-filtering adaptation of our correlation-based filtering approach, named *Correlation-Based Post-Filtering (CBPoF)*.

- **Chapter 7** reports the experimental analysis done to evaluate the pre-filtering (CBPF) and post-filtering (CBPoF) adaptation of our approach , and discusses the results. We evaluated our approach on four datasets of movie, music and tourism, and compared it with some well-known context-aware recommendation approaches of the state of the art.

- **Chapter 8** makes conclusive remarks and explains the future research lines to be investigated.

Chapter 2

Related Work

This chapter overviews related work in the field of CARS (Context-Aware Recommender System). For a better understanding of this type of recommendation, we start this chapter with a review of the notion of context, different context factors and context modeling approaches proposed in the literature. Next, we overview the main traditional recommendation approaches and similarity measures used in these approaches, by pointing out their strongest and weakest features. After this introduction to context and RS (Recommender System), we move on to context-aware recommendation, and present the major approaches in this field, by describing some well-known context-aware recommendation approaches of the literature.

2.1 Context

The notion of context has been studied by numerous researchers since the 90's and many definitions have been proposed for it. Bazire and Brézillon in [24] have explored and compared 150 different definitions for the context in various domains, like artificial intelligence, cognitive psychology, philosophy and linguistics. The authors conclude that because of the multiform nature of the context, it is difficult to find a unique definition.
In our research, we choose the definition proposed by Abowd et al. in [2], a clear and generic definition, which is the most widely accepted one in the context-aware computing community:

> "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

2.1.1 Context Factors

To be able to collect relevant contextual information and use it in an application, we need something more than this abstract definition. In fact, the context of a user is generally composed of a number of elements named context factors. Therefore, we study the multiple categorizations that have been proposed in the literature to describe the possible categories and values of context factors. Some of them correspond to a specific domain like contextual information retrieval [95] or context-aware recommender systems [3], and some of them are more generally proposed for contextual applications [2, 16, 117]. We can find more than 15 categories among these propositions. Table 2.1 shows some of the most relevant categorizations:

	Article	Location	Time	Physical	Identity	Activity/Status	Individuality	Modal	Social/Relations	Computing/Int. media	Environment	Dispositifs	User	Scenario
1	[2, 16]	x	x		x	x								
2	[117]	x	x			x	x		x					
3	[3]			x			x	x	x					
4	[72]	x	x							x	x	x	x	
5	[28]	x	x			x						x		
6	[38]		x	x					x			x		
7	[45]	x				x			x			x		

Table 2.1: Comparison of different context categorizations

- Abowd et al. in [2] and Baldauf et al. in [16] proposed a categorization based on three types of entity: *places*, *people* and *objects*. Abowd et al. [2] define their context categories as follow: **location**, represents information that help to determine what other objects or people are near the entity and what activity is occurring near it. **Identity**, regroups many pieces of related information such as phone numbers, addresses, email addresses, birth-date, list of friends, relationships to other people in the environment, etc. **Activity**, determines what is occurring in the situation, and finally the last context category is **time**. Baldauf et al. [16] proposed a similar categorization, but defined slightly differently: they defined the **identity** as a unique identifier for each entity. The **location** represents an entity position, co-location, proximity, etc. The **status (or activity)** regroups the intrinsic properties of an entity, e.g. temperature and lightning for a room, processes running currently on a device, etc. And **time** is used for timestamps to accurately define situation, ordering events, etc.

- Zimmermann et al. [117] distinguish four types of entity: *natural entity*, *human entity*, *artificial entity* and *group entity*, and explain their context categorization based on these four entity types: **individuality**, regroups information about the entity the context is bound to, which could be different based on the entity type: for *human entity*, the user properties such as preferences in language, color schemes, modality of interaction, menu options or security properties, and numberless other personal favorites. For *artificial entity*, the computing hardware descriptions, the product documentation, an application or service. And for *group entity*, characteristics that members of the group may share including interests, skills, cultural background or kinship ties in a social sense; and computing power, network connections, or display size in a technological sense. **Time**, the time zone of the client, the current time or any virtual time. **Location**, that can be physical or virtual (e.g. the IP address as a position within a computer network). The **activities** the entity is currently and in future involved in, described by explicit goals, tasks, and actions. And **relations** that an entity has established to other entities (persons, things, devices, services or information). In this category they identified three types of relations: *social relations*, which represent social aspects of the current entity context: informations about friends, neutrals, enemies, neighbors, co-workers, and relatives, the role that the person plays in this relationship, and the level of intimacy and sharing. *Functional relations*, identified when one entity makes use of the other entity for a certain purpose and with a certain effect. And *compositional relations*, which are relations between a whole and its parts.
  Villegas et al. [104] proposed a similar categorization based on the four categories described above.

- Adomavicius et al. [3] proposed the following four categories of context factors: **physical context**, like time, position, and activity of the user, but also the weather, light, and temperature when the recommendation is supposed to be used. **Social context**, which refers to the presence and the role of other people (either using or not using the application) around the user, and whether the user is alone or in a group when using the application. **Interaction media**, which represents the description of the device used to access the system, the type of media that are browsed and personalized (text, music, images movies, etc.). And the **modal context**, which refers to the current state of mind of the user, the user's goals, mood, experience and cognitive capabilities.

- Nguyen [72] identifies five dimensions for the context of a user: **temporal dimension**, like the user time zone, current time, virtual time, the beginning and the end of a situation, the duration of an event, an activity, a consulted resource, a planning, etc. The **spatial dimension**, like the object location (physical and virtual) or absolute position. And more specific characteristics like the place longitude, latitude, area, or the object orientation,

movement direction, speed, acceleration, etc. The **dispositif dimension**, which refers to information about devices, like screen size, screen resolution, processor power, RAM (Random-Access Memory), etc. The **environment dimension**, refers to the noise level, abnormal rate of carbon monoxide, etc. The **user dimension**, refers to some general informations like the name, age, birthday, nationality, mother tongue, etc., but also to some information related to the application domain like the goal, preferences, knowledge, competences, roles, center of interests, etc. And the last dimension named **scenario**, refers to the user intention, informations, knowledge, environment objects, etc.

- Benouaret [28] categorizes the user context by four categories: **user profile**, referring to user preferences and demographic informations. **Location**, represents user location defined by GPS (Global Positioning System). **Time** (morning/ night/ etc.), and **activity**, which represents the current user activity.
  Akermi and Faiz [9] use the same context categorization.

- Chen and Kotz [38] propose the following categories: **computing context**, like network connectivity, communication cost and bandwidth, etc. **User context**, which refers to information like user profile, location and social situation. **Physical context**, like lighting, noise, traffic condition and temperature. And **time context**, which refers to time of a day, week, month and season of the year.
  This context category is also used by Azouaou and Desmoulins in [13].

- Gu et al. [45] categorize contextual information about a user in four classes **person**, **location**, **computational entity** and **activity**.

- We can also add in this list the proposition of Schilit et al. [87], which is a more abstract decomposition of the user context. The authors identify three important aspects of context as: where you are, who you are with, and what objects are around you. These aspects define location and identity of people and objects in close surroundings (like the lighting, noise level, network connectivity, and even the social situation, e.g. whether you are with your manager or with a co-worker).

Based on these propositions of the context categorization and their categories definitions, we can extract some points that will help us to propose a generic categorization:

- We can find a first difference in the definition of entity types. Baldauf et al. in [16] categorize entities in 3 types: *places*, *people* and *objects*, while Zimmermann et al. [117] define four entity types: *natural*, *human*, *artificial* and *group* entity.
  We based our proposition (that we present in Chapter 4) on the first one, because it decomposes entities based on their nature, so it seems more differentiable. Indeed, the second one is slightly more detailed, but we can find some intersections between the categories.

- By comparing definitions made by different authors, we can find different granularities of context categorization, indeed some are more general than others. For example Adomavicius et al. [3] defined a *physical context* as a context type representing not only the time, the position, and the user activity, but also some environment properties like the weather, the light and the temperature at the moment of the recommendation. This big context category, include some more specific context categories of other propositions like *location* [2, 16, 72, 87, 117], *time* [2, 16, 72, 117], *user activity* [2, 16, 117] or *status of place entity* [16].

- In some cases, different authors named differently the same category. For example *interaction media* in [3], *dispositif* in [72] and *individuality for artificial entity* in [117], all refer to descriptions of devices (hardwares) or softwares used in the interaction.

- In some cases, different authors defined the same category, in different levels, with more or less details. It is the case for the social relation of the user [2, 3, 117]. Adomavicius et al. [3] focus only on the presence and the role of people around the user when using the application, and the fact that the user used the application alone or in group. But Zimmermann et al. [117] and Abowd et al. [2] go further by analyzing user's relations with other people [2] and entities [117], user's friends [2, 117] or even enemies [117], co-workers [117], relatives [117], etc. Zimmermann et al. in [117] explore even more than only social relations, and propose two other subdivided categories for entity relations: *functional relations* and *compositional relations*.

- We can also note that in some cases, authors classified the same properties in different categories. For example, Adomavicius et al. [3] classify the user's goal, beside the user's experiences and cognitive capabilities in the *modal context*, while Nguyen [72] puts it in the *user context*, beside general informations like the user name, age, nationality, etc.

As this study shows, a numerous identification and categorizations of context factors have been proposed in the literature. After having analyzed the characteristics of each one, we will propose in Chapter 4, a generic context categorizations for CARS, in order to be applied to a large spectrum of application domains.

2.1.2 Context Modeling

After identifying and collecting the relevant context factors, we can integrate them into different variety of context-aware applications. To define a context-aware application, we choose the definition of Abowd et al. [2] which mentions the dependence between the use of context and the user's task:

> "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

The representation of contextual information in a context-aware application is a question that has been studied by several researchers. The main models presented in the literature to represent and store context data are the followings: *key-value*, *mark-up scheme*, *graphical*, *object-oriented*, *logic-based*, *ontology-supported* models [49, 16], *graph/tree* models and *domain specific language* [49]. Few comparisons have been made among these models. While *key-value* models are the simplest ones, some researchers demonstrate that *ontologies* can be more expressive [90, 58], while being flexible and extensible models [58]. However ontologies are domain-specific and can not be used generically.

In this first part of our state of the art chapter, we studied the literature of the context, its definition, factors categorizations and modeling approaches. The next part focus on the recommender system literature.

2.2 Recommender System

Nowadays, we are faced with a rise of the amount of data on the web, provided by different sources. As a consequence, a user can quickly be overwhelmed by the huge volume of information. RSs [50] aim to help the user to find her appropriate information among all others, by providing suggestions for items the user is likely to be interested in. In RSs the users' preferences over items are illustrated via ratings into a utility matrix ($user \times item \rightarrow rating$). These ratings can be collected explicitly by asking directly the users to rate items, or implicitly by inferring from the user behavior and her interactions on the application, like clicks on items or item purchases. Technically, a recommendation problem is often viewed as a 2D matrix completion problem. In a nutshell, the aim of the recommender system is to estimate missing ratings of the utility matrix, and then to recommend to each user her corresponding items with higher estimated rates.

2.2.1 Recommendation Approaches

Recommendations are principally based on three main approaches [5], *content-based*, *collaborative filtering* and *hybrid* approaches (illustrated in Figure 2.1):

2.2.1.1 Content-Based Approaches

In this family of approaches, beside the utility matrix, the characteristics of items are also used to do the recommendations. The RS tends to recommend items similar to the ones the user liked in the past. The similarity among items is computed based on the associated characteristics of compared items.

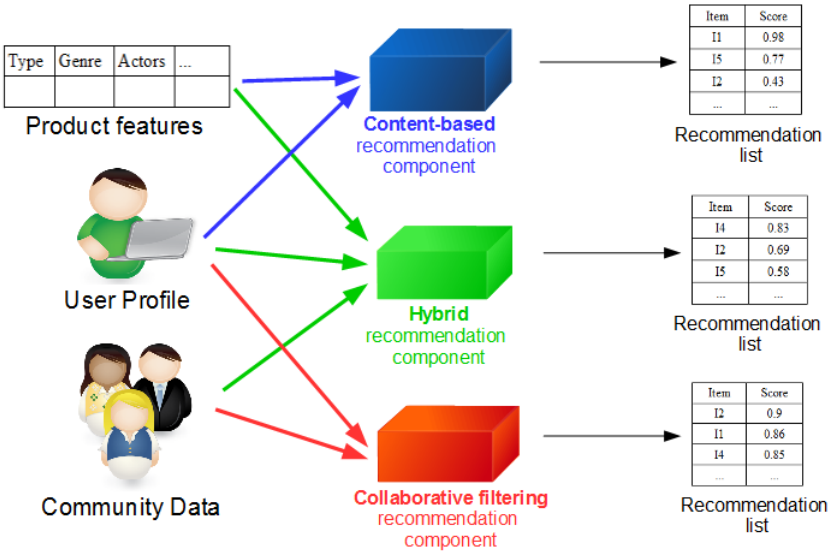


Figure 2.1: Recommendation approaches (adapted from [50])

Example 2.2.1. *As an example of movie recommendation, the Table 2.2 shows the characteristics of the available movies (the genre and the year of production of each movie), with the ratings given by the user (1 if he liked the movie, and 0 if not). Suppose the movies that John liked in the past are: The passenger, a 2018 action movie and Transformers 5, a 2017 sci-fi movie. Based on what he liked in the past, a content-based approach will recommend to him, recent movies of action or sci-fi genre, which are Fast & furious 8 and Star wars 8.*

Movie	Genre	Year	Rating
The passenger	**action**	**2018**	1
Her	romance	2013	0
Fast & furious 8	action	2017	?
Transformers 5	**sci-fi**	**2017**	1
Life as we know it	comedy	2010	?
Star wars 8	sci-fi	2017	?

Table 2.2: Example of movie recommendation by a content-based approach

2.2.1.2 Collaborative Filtering Approaches

This category of recommendation approaches is based on two key assumptions [85]: users who had similar tastes in the past will have similar tastes in the future, and users' preferences remain stable and consistent over time. The implementation can be user-based, which is based on users' neighborhood, or item-based [61], which is based on items' similarity. So in the user-based case, the RS will recommend items that people with similar tastes (her neighbors)

liked in the past. The similarity among tastes of two users is computed based on the similarity in the rating history of the users [84].

Example 2.2.2. *Table 2.3 illustrates a simplified example of utility matrix, with 1 and 0 instead of ratings, where 1 means that the user liked the corresponding item, and 0 means that she did not like it. In this example our target user is U4 and we want to know which one of the items I2 or I4 to recommend to her. In the case of user-based collaborative filtering, we can see that U4 is very similar to U1, because both of them liked I1 and I5 and disliked I3. So in this case, a good recommendation for U4 would be I2, an item liked by U1.*

	I1	**I2**	**I3**	**I4**	**I5**
U1	1	1	0		1
U2	0		1	1	
U3		0	1		1
U4	**1**	**?**	**0**	**?**	**1**

Table 2.3: Example of utility matrix for collaborative filtering

There exist two classes of collaborative filtering techniques [91]:

1. *memory-based techniques*, basically compute the similarity among users or items to build neighborhoods methods. Different similarity measures (cosine, Pearson correlation coefficient, etc.) and score functions (simple, weighted or normalized average) are proposed in the literature to identify the neighborhoods,

2. *model-based techniques* are machine learning techniques like clustering, decision tree, SVM, matrix factorization methods, etc. [5, 29] that try to build a global model to estimate missing ratings.

2.2.1.3 Hybrid Approaches

Hybrid approaches try to combine the previous approaches, in order to take advantages of both of them. Jannach et al. [50] propose three different ways to do this combination:

1. by a monolithic exploiting of different features, where features/knowledge sources of different paradigms are combined in a single recommendation component,

2. by a parallel use of several systems, where several existing recommendation techniques are applied in parallel, and their outputs are combined based on some weighting or voting scheme,

3. by a pipelined invocation of different systems, where we have a list of recommendation techniques, executed consecutively. In this pipeline successor's recommendations are restricted by predecessor, which means that one recommender system pre-processes some input for the subsequent one.

Approaches	Advantages	Disadvantages
Content-based	– User independence : no need for data on other users [64]. – Able to recommend to users with unique tastes [70]. – Able to recommend new and unpopular items [64]. – Transparency: explanations on how the recommender system works can be provided [64]. – No issues with low matrix density [70].	– Limited content analysis: finding features can be hard (images, movies, music) [64, 5]. – Overspecialization [64, 70, 5]. – Cold-start problem for new users [64, 70, 5]. – Impossible to represent everything using keywords [70].
Collaborative filtering	– Requires minimal knowledge engineering efforts [70]. – Produces good-enough results in most cases. – No need of item's descriptions.	– Cold start problem for new users/items [5, 29]. – Sparsity [5, 70]. – Scalability [29]. – Use limited information [29]. – Require a large number of reliable "user feedback data points" to bootstrap [70]. – Require products to be standardized (users should have bought exactly the same product). – Data security issues [70].
Hybrid	– The solution for some recommendation challenges like data sparsity and gray sheep [91].	

Table 2.4: Advantages and disadvantages of different recommendation approaches

2.2.1.4 Other Approaches

In the literature, we can find some less common approaches like knowledge-based [35], trust-based [11], graph-based [83], group-based recommendations [83] or social filtering [29].

Table 2.4 lists some advantages and disadvantages of the three main *content-based*, *collaborative filtering* and *hybrid* approaches. In *content-based* approaches, as we based the recommendations on the previous items that the user liked, without the need of any data from other users, we can propose recommendations even to users with unique tastes. Moreover, new and unpopular items have their chance to be recommended. However, this specificity can lead to an overspecialization of the recommendations, and contrary to *collaborative filtering* approaches, the degree of novelty would be limited, hence the user is going to be recommended items similar to those already rated. As a result, this kind of recommendation techniques would be useful for a limited range of applications [64]. On the other hand, *collaborative filtering* approaches have the advantage of no needing item's descriptions and characteristics, which are sometimes difficult to be gathered (e.g. for images, movies, etc.). But they are sensible to sparsity, and have to deal with low-density utility matrix. Also, as they are black boxes, it is almost impossible to explain the user why a certain items is recommended to her. Conversely, in the case of *content-based* approaches, we can provide explanation about the list of recommendations by explicitly listing the characteristics and descriptions of the recommended items. Hybrid methods can be good solutions to limit the drawbacks of these two families of approaches.

2.2.2 Similarity Measures Used in Recommendation

Different similarity measures can be used in recommendation approaches to measure the similarity between different entities (e.g. two users, two items, etc.). As an example we explain the formulas used for computing the similarity between two users u and v, or two items i and j:

1. **Jaccard similarity:** By this measure we ignore rating values in the utility matrix and focus only on the sets of items rated. The idea is that users are more similar if they have more common ratings. This measure is useful when we have only information about purchases and not more detailed ratings. Equation 2.1 illustrated the similarity between users u and v, where I_u and I_v are respectively the set of items rated by u and v.

$$sim_{jacc}(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{2.1}$$

2. **Cosine similarity:** The cosine of the angle between the vector representations of two entities (users/items) can be calculated to measure the similarity between them. So, the

cosine similarity between two items i and j would be computed as follows in equation 2.2:

$$sim_{cos}(i,j) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{\|i\| \, \|j\|} \tag{2.2}$$

3. **Adjusted Cosine similarity:** This measure is a modified form of the cosine similarity that takes into account the difference between the rating schemes of users. Indeed, some users tend to rate items highly in general, while some other are more severe and tend to rate items with lower ratings (Equation 2.3, where $R_{u,i}$ is the rating that user u gives to item i, and $\overline{R_u}$ is the average ratings given by user u).

$$sim_{adjCos}(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{(\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_u})^2}} \tag{2.3}$$

4. **Pearson Correlation Coefficient:** This measure is based on how much the ratings given by common users for a pair of items ($R_{u,i}$, $R_{u,j}$) deviate from average ratings for those items ($\overline{R_i}$, $\overline{R_j}$) (Equation 2.4). The resulted values are between -1 and 1.

$$sim_{pcc}(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{(\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_i})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_j})^2}} \tag{2.4}$$

5. **Other measures:** We can add some less common similarity measures, based on asymmetric cosine [8], Euclidean distance [41], Manhattan distance [41], Log-Likelihood [42], Spearman's rank correlation coefficient [88], etc. that can also be used in recommendation techniques [68].

Now that we have introduced the two main components of a CARS: the context and the RS, and presented their properties, in the next section, we can describe different CARS approaches to integrate the contextual information of users into the recommendation process.

2.3 Context-Aware Recommender System

CARSs aim to take into account the users' contextual information in order to propose more relevant and personalized recommendations [6]. So instead of the 2D rating function of traditional RSs ($R : user \times item \rightarrow rating$), in CARSs we have a multidimensional function, $R : user \times item \times context \rightarrow rating$ [4]. As mentioned previously, the context of a user is composed of a number of context factors like *time, location, weather, companion, etc.*. To each one of these context factors some values can be associated, called context conditions. For example possible context conditions for *time* could be *morning, afternoon, evening* and *night*, and for *companion* could be *alone, friend, family, etc.*.

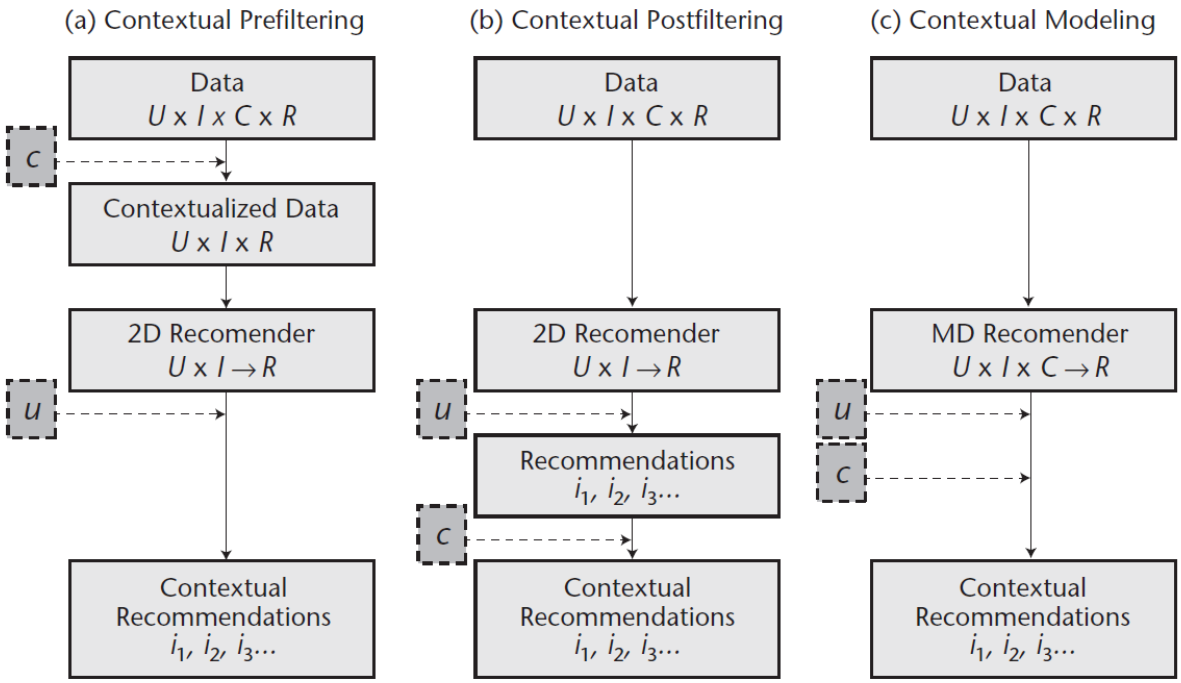


Figure 2.2: Approaches for incorporating context in recommender systems [6]

Example 2.3.1. *A simple example of context-aware movie recommendation dataset is illustrated by Table 2.5. In this example we want to estimate the rating that user U1 would give to the animation Toy Story 3, if she watch it with a children. As the previous experiences of U1 show, she did not like to watch another animation (Monsters, Inc) when she watch it with her partner, but she liked to watch it with a children. So she will probably like to watch a similar animation (Toy Story 3) in a similar context, so the estimated rating would be high.*

User	**Movie**	**Rating**	**Companion**
U1	Monsters, Inc	2	Partner
U1	Monsters, Inc	5	Children
U2	Taken	4	Friend
U1	Toy Story 3	?	Children

Table 2.5: Example for context-aware movie recommendation

The integration of contextual information in CARSs can be done by relying on either *pre-filtering*, *post-filtering* or *contextual modeling* [6] as illustrated in Figure 2.2, where U, I and C refer respectively to the user, item and contextual dimensions, R refers to ratings, and $i_1, i_2, i_3, ...$ refers to the list of contextual recommendations for user u in context c.

2.3.1 Pre-filtering Approaches

In pre-filtering approaches, contextual information is used to select only appropriate data based on the target user context situation, and then a traditional recommendation technique is applied on this selection. Numerous approaches have been proposed in this category. We can mention: the *reduction-based* approach [4], with its two variants *exact pre-filtering* and *generalized pre-filtering*; the *splitting* approaches: *item splitting* [23], *user splitting* [17] and *UI splitting* [110]; the *Differential Context Modeling (DCM)* approach, and its two variants *Differential Context Relaxation (DCR)* and *Differential Context Weighting (DCW)*, proposed in [109]; and the *Distributional Semantic Pre-Filtering (DSPF)* approach [40].
In the following, we briefly describe *DSPF*, *DCM* and *Splitting* approaches, which will be compared with our proposed approach in the experimental chapter. We choose these well-known approaches because similarly to ours, these context-aware approaches make use of context similarities in their context integration process.

2.3.1.1 DSPF

DSPF [40] models the influence of context on ratings based on the difference between context-free rating and the rating given in the specific context. So each context is represented based on this model to be used in the pre-filtering phase of the approach. Based on the assumption that two context situations are similar if their composing context conditions influence users' ratings in a similar way, at first each context condition (c) is represented by a vector (w_c) containing the item-based or user-based influence of the context condition on ratings. For example in the item-based case, the influence of the context condition c on ratings of item i, noted as w_{ci}, is computed based on the difference between the rating given by user u for item i in this context situation (r_{uic}), and the predicted context-free rating ($\hat{r}_{ui}$), as illustrated in Equation 2.5:

$$w_{ci} = \frac{1}{|R_{ic}| + \beta} \sum_{r_{uic} \in R_{ic}} (r_{uic} - \hat{r}_{ui}) \tag{2.5}$$

where R_{ic} is the set of ratings for item i in condition c, and β is a decay factor for decreasing the estimated deviation when $|R_{ic}|$ is small. The context-free rating, $\hat{r}_{ui}$, is calculated by the baseline context-free predictor of [56], which is the sum of the overall average ratings (μ) and the observed deviations of user u (b_u) and item i (b_i): $\hat{r}_{ui} = \mu + b_u + b_i$. Then, the context situation representation (w_s) is made by an aggregation of the context conditions (c) composing this context (s) (equation 2.6).

$$w_s = \frac{1}{|s|} \sum_{c \in s} w_c \tag{2.6}$$

Finally, based on these context representations, similarities between contexts are computed and ratings given in contexts similar to that of the target user are selected. A traditional recommendation can then be obtained on this selection.

In *DSPF*, the representation of the context by the influence of contexts on ratings is effective, but the computation of this influence is not user-centric enough. In fact, in this computation the context-free ratings are computed by the equation $\hat{r}_{ui} = \mu + b_u + b_i$, which is only behavior-based and not personalized. Suppose we want to estimate the context-free ratings of an over-rated science-fiction movie (with a highly positive b_i), by two strict users (with a highly negative b_u) who have radically different interests (one who loves science-fiction movies and the other who hates them). In this case, contrary to what is expected, the estimated ratings will be the same for these two persons. Therefore the influence calculated by this measure would be biased, and the values of context similarities in the pre-filtering phase would not be very relevant, which will affect the recommendation results.

2.3.1.2 DCM

DCM [109] is based on the user-based collaborative filtering algorithm [91]. The authors propose to separate the algorithm into different functional components, and apply differential context constraints to each component, in order to maximize the performance of the whole algorithm. They divide the approach into two parts: the *differential* part, and the *modeling* part, which can be performed by *context relaxation (DCR)* or *context weighting (DCW)*. The idea of *DCR* is to find an optimal context relaxation for each component of the recommendation algorithm. But *DCR* is a strict action, and there will be a sparsity problem in cases where we have less contextual information. Thereby, the authors proposed *DCW*, which assumes that the more similar the contexts of two ratings were given, the more valuable those ratings will be in making predictions. So in *DCW*, they do not filter out certain ratings, but assign a score to all ratings based on context. The similarity between two contexts is computed by the Jaccard similarity between their sets of known conditions. And in both of these *DCM* variants, the optimal weights are computed based on the Particle Swarm Optimization (PSO) [97] method.

2.3.1.3 Context-Aware Splitting Approaches (CASA)

Three different variants of splitting approaches have been proposed in the literature: Baltrunas and Ricci propose the *item splitting* in [23]. Its basic idea is to identify the context condition in which items are rated significantly differently, and split each item into two new ones based on this context condition. Toward this end, the algorithm iterates over all context conditions of each context factor, and evaluates the splits based on the impurity criteria. The outcome is multiple copies of each item based on the contexts in which they have been rated. Therefor the context dimensions are eliminated from the initial data matrix, transforming it to a 2D matrix. The *user splitting* proposed in [17] conducts a similar treatment with respect to users. And finally Zheng et al. propose in [110] a combination of these two approaches, named *UI splitting*. They propose to split both users and items of the dataset, by an application of *item splitting*

followed by *user splitting* on the resulting output.

2.3.2 Contextual Modeling Approaches

Contextual modeling approaches try to extend traditional recommendation techniques by integrating directly contextual information into the recommendation algorithm. Some of the most popular propositions in this category are the followings: *Tensor Factorization (TF)* and its variants *multiverse recommendation* [52] and *factorization machine* [81]; *the deviation-based Context-Aware Matrix Factorization (CAMF)* [21] with its several derived model: *CAMF-C*, *CAMF-CI*, *CAMF-CC* and *CAMF-CU*; *Contextual Sparse Linear Method (CSLIM)* [113]; *the similarity-based approaches of CAMF and CSLIM* [115] with their three versions *ICS*, *LCS* and *MCS*; and the *context-aware collaborative filtering* proposed by Chen in [37].
In the following, we briefly describe the *deviation-based CAMF* approach, for which good performances have been reported in the literature. In the experimental chapter, we will compare it to our proposed approach.

2.3.2.1 Deviation-based CAMF

CAMF [21] is an extension of matrix factorization [57]. The deviation-based version tries to take into account the context situation of users by integrating additional model parameters B_{ijc_j} in the matrix factorization equation. So the rating of item i by user u in the context situation $c_1...c_k$ will be estimated by the Equation 2.7:

$$\hat{r}_{uic_1...c_k} = \overrightarrow{p_u}.\overrightarrow{q_i} + \bar{i} + b_u + \sum_{j=1}^{k} B_{ijc_j} \tag{2.7}$$

where k is the total number of contextual factors, $\bar{i}$ is the average of the item i ratings, b_u is the baseline parameter for user u, $\overrightarrow{p_u} \in V$ is the column vector representation of the user u, and $\overrightarrow{q_i} \in Q$ the column vector representation of the item i in the factorized matrices V and Q. Depending on the version of *CAMF*, the B_{ijc_j} parameters will model different granularities of the interaction of context with ratings: *CAMF-C* uses one single parameter for each context condition, *CAMF-CI* uses one parameter per context condition and item pair, and *CAMF-CC* uses one parameter for each context condition and item category.

CAMF proved its effectiveness to improve recommendation performance in comparison to context-free recommendation and some of the context-aware recommendation approaches, but like other contextual modeling approaches, it has the disadvantage of needing to be implemented from scratch, with no possibility of re-using recommendation techniques already in production.

2.3.3 Post-filtering Approaches

In post-filtering approaches, first a context-free recommendation algorithm is applied on the data by ignoring the contextual information, and then the resulting recommendation list is contextualized by filtering or reordering items. This category of approach has received less attention than the two previous categories, but we can still cite the *content-based post-filtering* model [48]; and the *weight post-filtering* and *filter post-filtering* approaches proposed by Panniello et al. in [78]. The authors of this article propose to contextualize the predicted ratings (from the traditional recommender system) as follows: the basic idea is to firstly compute the contextual probability $P_s(u, i)$, which is the probability that the user u liked the item i in the context s. This probability is equal to the number of similar users to u (her neighbors) who liked the same item i in her context s, divided to the total number of neighbors who rated this item. Then the predicted ratings are either weighted based on this probability, by multiplying the predicted ratings by their corresponding probability (*weight post-filtering*), or filtered by eliminating items for which the corresponding probability is less than a threshold (*filter post-filtering*).

2.3.4 Conclusion

Among the above mentioned approaches (*pre-filtering*, *post-filtering* and *contextual modeling*) there is no clear winner [78], and the experiments showed that effectiveness depends on the dataset and the exact application domain.
Like *post-filtering* approaches, one of the advantages of *pre-filtering*, in comparison to *contextual modeling*, is the re-usability of traditional recommendation techniques. *Post-filtering* techniques are characterized by the fact that a context-free recommendation is firstly done, and only after that, a reordering or filtering is done on the recommendation list by taking into account contextual information. So, much of the recommendation has been done without considering contextual information. Instead, in *pre-filtering* approaches, the user's contextual information is set as first class citizen, and this is used at the first stage of the process to guide the recommendation based on it. This is the reason why *pre-filtering* approaches have gained substantial attention by the research community, and most of the proposed CARS approaches in the literature are *pre-filtering* approaches [47]. Therefore in this thesis, we first led our approach in the *pre-filtering* direction, but also adapted it to *post-filtering*, in order to take advantage of the re-usability of the filtering module and compare these two families of approaches.

2.4 Context Relevance in CARS

In the literature some definitions for the context have been proposed, and many studies have proved the effectiveness of taking into account the contextual information of the user in the

recommendation process [6]. But depending on the application, not all of the available contextual information is relevant. So, before integrating this kind of information into the system, we have to identify the relevant piece of information. Indeed in some cases irrelevant contextual information could cause a decrease in the recommendation performances. Also note that the automatic acquisition of some contextual information (like user's mood) is still hard or even impossible, and have to be specified directly by the user. So, avoiding useless user's effort in the data acquisition phase is also valuable. These two points are the main motivations for the identification of context relevance in CARS.
Here we mention some propositions in the literature:

- In [20], the authors conducted a work on the tourism domain, and ask users to imagine a given situation and evaluate the influence of the corresponding context conditions on their ratings. The main problem of such approaches is the fact that users rate differently in real and supposed contexts.

- Odic et al. in [73] compared two different approaches: in the first one (*assessment*) the users are asked to specify the relevancy of contextual information (similarly to [20]), and in the second approach (*detection*) the relevancy detection is done by the mean of statistical testing. They applied the Freeman-Halton test, which is the Fisher's exact test extended to contingency tables larger than 2x2. The null hypothesis of the test is that the context factor f_i and the ratings are independent. The reject of the null hypothesis shows that the context factor f_i is relevant. Each one of these approaches has their positive and negative points: Contrary to the *detection* approach, in the *assessment* approach we don't need a substantial number of rating data. But, *assessment* is intrusive (asking users to spend time on this task), and obtained from hypothetical situations, while the *detection* is made on real situation data, and done without the need of user's effort. The results showed that the *detection* performs better, which prove the fact that the users are not always aware of what really influences their decisions.

- The authors of [102] applied a Las Vegas Filter (LVF) algorithm, which repeatedly generates random subsets of contextual factors and returns the subset with the best evaluation, based on an inconsistency criterion.

- Braunhofer et al. [34] propose to compute a personalized relevance score for a context factor and user-item pair by the mean of a method called *Largest Deviation* . For each user and item pair, whose rating is acquired, they first compute the impact of each context condition, which is equal to the absolute deviation between the rating prediction when this specific condition holds and the context-free rating prediction. Then the relevance score for the corresponding context factor will be the aggregation (arithmetic mean) of the impacts of the possible context conditions for that context factor.

- Codina proposes in his thesis [39] to quantify the relevance of the contextual information at condition level instead of factor level. Therefor the relevance of context conditions will be the variance of their corresponding semantic vectors (presented in section 2.3.1.1).

$$r_c = \frac{1}{l} \sum_{i=0}^{l} (w_{ci} - \mu_c)^2 \tag{2.8}$$

  The Equation 2.8 shows the relevance (r_c) of a context condition c, where w_c is the influence vector of c with respect to items or users, and μ_c is its mean influence.
  This relevance measure can be used in two different ways: (a) the *weighting* method, in which the representation of a context situation is modified by using a weighted average of the vectors of its composing conditions (the weights correspond to the relevancies of each context condition); (b) the *filtering* method, in which the q least relevant context conditions are excluded in the context representations computation.

In this section, we state the importance of context relevance consideration in CARS recommendation process. The previous works on this point, described above, have shown the positive impact of this consideration on the recommendation performances.

2.5 Research Topics in CARS

In this section, we give a brief overview of different active research topics in the domain of CARS, by exposing recent works:

- Some works focus on a specific domain application of CARS, like tourism [12, 14, 67, 36], music [79], videos on Youtube [1], news [63], indoor shopping [75], post popularity prediction in social media [66], OLAP queries [71], etc. Contrary to these domain-specific propositions, in this thesis we propose a generic context-aware recommendation approach, in order to be applied to various application domains.

- Different methods have been proposed to be applied on CARS: ontologies [14, 12], cased-based reasoning [15], hidden Markov models [7], neural network based model [62], deep-learning and/or embedding-based CARS [86, 10], sequential predictions [111], etc. Our CARS proposition is a data-driven approach, with a correlation-based representation of the context, by modeling its influence on ratings in the recommendation process.

- Some works propose combinations of different families of RS: Véras et al. proposed a hybrid approach named CD-CARS that combines Cross-Domain RS (which try to enhance the quality of recommendations in a target domain by considering ratings from source and target domains) and CARS [103]. A context-aware group recommender system is proposed by Khoshkangini et al. in [54]. Some other works used the opinion mining information

beside the contextual information in CARS [94]. In this thesis, we focus on the context-aware recommendation, by leveraging the dynamic user context and the static user/item content information for user-centric recommendations.

- Well-known RS challenges still get attention of the CARS community, like cold start [31, 32, 108], sparsity [107] and explanation problem [44]. In our work, we try to address the sparsity and explanation problem by a filtering approach.

- We concentrate our effort on the contextual information identification and its integration to the RS to provide context-aware recommendations. Different other aspects of the context are also studied in CARS, like context extraction [92, 93], context suggestion [116] and context-driven RS [76].

2.6 Synthesis

In this chapter, we review the related work done on the domain of *context*, *recommender systems*, and *context-aware recommender systems*. We notice that there does not yet exist a universal definition of the notion of context, and several categorizations, from different points of view, have been proposed. Consequently, in this thesis, we try to propose a generic categorization of context factors for CARSs so as to be used for any application domains (like music, movies, news, etc.) to identify context factors. Also, the previous research in the literature demonstrates that integrating such contextual information into the recommendation process can improve the resulting recommendations. From this line of research, the second objective of this thesis is to propose a new data-driven approach to transform any existing traditional recommender system into a CARS, in order to produce more user-centric recommendations. We adapted our context-aware recommendation approach to two families of CARS: *pre-filtering* and *post-filtering*.
We also point out that it could be interesting to take into account the impact degree of context factors in the recommendation process. So, we consider this aspect in our proposed approach in order to improve the recommendation results.

Chapter 3

Preliminaries

This chapter surveys the background material used within this thesis. Besides stating conventions, we summarize some statistical models, machine learning and recommendation techniques used in our project.

3.1 Conventions

In the next chapters, we will use some expressions about contextual information. For a better understanding we explain each one by some examples.

- *Context or context situation* [1] refers to the set of information that describes the situation of an entity. For example the current context or context situation of James could be <*morning, alone, happy, at home*>, which means this is a morning where James is happy and alone at home.

- *Context factor:* Each context situation is composed of some context factors. In our example (<*morning, alone, happy, at home*>), the context of *James* is composed of the four context factors: *time*, *social*, *mood* and *location*.

- *Context condition:* To each context factor, some values can be associated. The possible values of a context factor are called context conditions. For example {*morning, noon, evening and night*} would be the context conditions associated to the context factor *time*.

3.2 Clustering Techniques

Clustering is a Machine Learning technique that aims to group similar data points together, in order to have data points with similar features in the same group, while data points in different

[1] Note that in information system, context and situation are stating for two different concepts. But in this thesis, we do not focus on this subject and use these two equivalently.

groups have highly dissimilar features. It is a non-supervised learning method, used in many fields, including, information retrieval, image analysis, pattern recognition, etc.

Different categorizations of clustering techniques have been proposed in the literature. For an exhaustive list, see [106]. Here we listed the most widely used clustering methods which are associated to one of the following four categories:

- *Centroid-based clustering (based on partition):*
  In this family of clustering methods, the center of data points corresponds to the center of the corresponding cluster. The main advantage of them is their relatively low time complexity. The two most famous techniques in this category are *K-means* [65] and *K-medoids* [53].
  For *K-means*, we first have to select a number (k) of groups/clusters. This can be done by taking a quick look at the data and trying to identify distinct groupings. Then we randomly select k data points as the center of these clusters. The distance between each data points of the dataset and each cluster center is computed to classify them in the group which center is closest to each one. Based on this classification, the centers of clusters are recomputed, and in an iterative process the last two steps are repeated until the cluster centers do not change much between iterations.
  K-medoids is a version of *K-means* which deals with discrete data.

- *Hierarchical clustering:*
  The basic idea of this family of approach is to construct a hierarchical relationship among data points in order to cluster them. In these algorithms, each data point is first treated as a single cluster, and then iteratively merge pairs of most similar clusters into a new cluster until there is only one cluster left. At the end we can select the number of clusters by choosing when to stop combining the clusters. This is the bottom-up version of this algorithm, also known as Hierarchical Agglomerative Clustering (HAC). A reverse method, top-down version is also applicable [51].

- *Distribution-based clustering:*
  This clustering approach cluster data points based on their distribution. When there exist multiple distributions in the dataset, data generated from the same distribution will belong to the same cluster. The Gaussian Mixture Models (GMM) [80] are well-known models of this family of approaches, which use the Expectation-Maximization (EM) [69] optimization algorithm to find Gaussian parameters. In these models, as *K-Means*, we first select the number of clusters, and initialize randomly the Gaussian distribution parameters for each cluster. Then by computing the probability of belonging to a particular cluster, we associate each data point to the cluster for which the Gaussian's center is the closest to that data point. Based on these probabilities, we recompute the parameters of the Gaussian distributions such that we maximize the probabilities of data points within the

clusters. The last two steps are repeated until the distributions do not change much between iterations.

- *Density-based clustering:*
  The basic idea of density-based clustering techniques is to define clusters as areas of high density, in sort to associate to the same cluster, data points which are in a region of the data space with high density. These algorithms do not assign outliers to clusters and consider them as noise. DBSCAN *(Density-Based Spatial Clustering of Applications with Noise)* [43] is the most well-known model in this family of approaches. *DBSCAN* works with two parameters *epsilon(ϵ)* and *minPoints* threshold. We begin with an arbitrary starting data point, and associate all data points within the ϵ distance as its neighborhood. If the size of the neighborhood is equal or greater than the *minPoints* threshold, the current data point becomes the first point in the new cluster, otherwise it will be labeled as noise. Then the neighbor data points are also be part of the same cluster and the last step will be repeated for all of the new data points just added to the cluster. Then, a new unvisited point is retrieved and processed, which will leads to the discovery of a further cluster or noise. And we will repeat this whole process until all data points will be visited and marked as a cluster member or noise.

Each class of approaches has its advantages, *centroid-based* techniques have relatively low time complexity, *hierarchical clustering* and *density-based* techniques do not require a pre-defined number of clusters and *distribution-based* techniques are more realistic by giving the probability of belonging of data points to clusters [106]. We can not define a clustering approach as the best one. However *K-means* and *HAC* are two of the most popular clustering techniques. In our approach, we used the *HAC* technique in order to cluster the items or users. We will explain more in section 7.1.2.

3.3 Matrix Factorization

Matrix factorization (MF) [57] methods came from the collaborative filtering approaches, which become more and more popular, due to their good performances, especially since the Netflix prize [25]. Multiple MF models have been proposed to improve recommendation performances by adding some weights or regularizations parameters.
The basic idea behind matrix factorization methods is to decompose the (user-item) utility matrix into two lower dimensionality rectangular matrices q_i and p_u (Figure 3.1). Each item i will be associated to a vector in q_i, and each user u will be associated to a vector in p_u. Then the missing rating of user u for item i would be predicted by the product of these two lower dimensional latent matrices (Equation 3.1, where $\hat{r}_{ui}$ is the predicted rating of user u to item i).

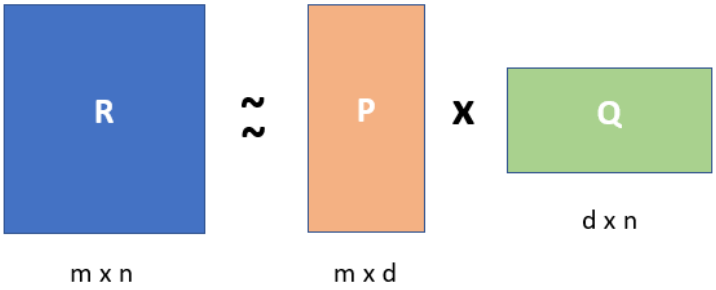


Figure 3.1: Matrix Factorization

$$\hat{r}_{ui} = q_i^T p_u \tag{3.1}$$

The system learns the factor vectors p_u and q_i by minimizing the squared error on the set of known ratings, as follows in Equation 3.2.

$$\min_{q_\cdot, p_\cdot} \sum_{(u,i)\in\kappa} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{3.2}$$

Note that in order to avoid overfitting the known ratings, a regularization is done in this process. In this equation κ is the set of (user, item) pairs in the training set, for which the ratings is known, r_{ui} refers to the rating of user u to item i, and λ controls the extent of regularization.

Two different approaches can be used to minimize the above equation [57]:

- *Stochastic Gradient Descent (SGD)* [30], which has the benefit of implementation ease and fast running time. In this algorithm, for each observation in the training set (u, i, r_{ui}), the system predicts the rating (r_{ui}), and computes the associated prediction error (e_{ui}):

  $$e_{ui} = r_{ui} - q_i^T p_u \tag{3.3}$$

  Based on this error, the system modifies the parameters in the opposite direction of the gradient, by a magnitude proportional to γ, as follow in Equations 3.4.

  $$\begin{aligned} q_i &\leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \\ p_u &\leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \end{aligned} \tag{3.4}$$

- *Alternating Least Squares (ALS)* [26], which is favorable to use in case of parallelization, or when we work with implicit data. The algorithm alternates between optimizing p_u and

q_i. In each step, the system fixes one of them, and recomputes the other one by solving a least-squares problem. This process is repeated until convergence.

SGD has the advantage of being easier and faster than ALS. This is why in this thesis we used an SGD-based matrix factorization technique, named *BiasedMF* [55] (explained below), in our recommendation process.

3.3.1 BiasedMF

In the rating prediction process presented above, the MF tries to capture the interactions between users and items (Equation 3.1). But it could be interesting to also take into account the effects of either users or items on the ratings. This is the idea of a matrix factorization technique proposed by Koren in [55], where these effects are known as user/item biases. By this way, for example, we would take into consideration the fact that some users are more critical, and tend to rate lower than the average, or some items tend to be usually overrated.
So, the bias involved in rating r_{ui}, denoted b_{ui} in Equation 3.5 will be the sum of the overall average rating (μ) and the observed deviation of item i (b_i) and user u (b_u).

$$b_{ui} = \mu + b_i + b_u \tag{3.5}$$

Thus instead of the Equation 3.1, the prediction of the rating of the user u for the item i would be calculated as follows in Equation 3.6:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u \tag{3.6}$$

Then, the system will minimize the following regularized squared error to learn the latent factors p_u and q_i:

$$\min_{p\cdot,q\cdot,b\cdot} \sum_{(u,i)\in\kappa} \left(r_{ui} - \mu - b_u - b_i - p_u^T q_i\right)^2 + \lambda(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2) \tag{3.7}$$

3.4 Conclusion

In this chapter we first explained some expressions about contextual information that we use in this thesis: *context/context situation*, *context factor* and *context condition*.
We detailed and compared different families of clustering techniques. We use the clustering in the representation of the context in our correlation-based filtering approach.
Finally, we described a matrix factorization technique, *BiasedMF*, that we use in our pre-filtering and post-filtering approaches (*CBPF* and *CBPoF*) as the traditional 2D recommendation technique.

Chapter 4

Context Factors in Recommender Systems

Much research has proved that context-aware recommender systems (CARSs) can outperform the traditional recommender systems (RSs) by proposing more relevant recommendations to users [6]. However their performances are impacted by the contextual information, which is not clear [24]. In this thesis our main focus is on the integration method of contextual information into the recommendation process. Though, in an industrial context, the first step for developing a context-aware recommender system is to identify and collect this contextual information. As mentioned in the Chapter 2, the context has been studied by researchers of various domains. But due to the lack of consensus, there does not yet exist a standard definition for the context. In this chapter, our objective is to identify and propose concrete and clear information to describe the context of the users, by improving the representation of the user context in the case of CARS. We propose a hierarchical categorization of context factors. Our proposition allows to be applied to a large spectrum of application domains of CARS.

4.1 Context Categorization

In this thesis, we took the well-known context definition of Abowd et al. in [2], which says that the context is any information that can be used to characterize the situation of an entity. This entity could be a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.
Based on this definition, to reach a complete and appropriate context model for CARSs, we propose to identify all possible context factors. Our objectives for this new proposition of context factors categorization are to answer the needs of CARS, while (1) satisfying the definition of Abowd et al. [2], (2) improving the previous propositions, (3) allowing to work with context in different levels, and (4) allowing its application to a large spectrum of application domains.
We have been inspired by the context factors proposition of Adomavicius et al. in [3], and

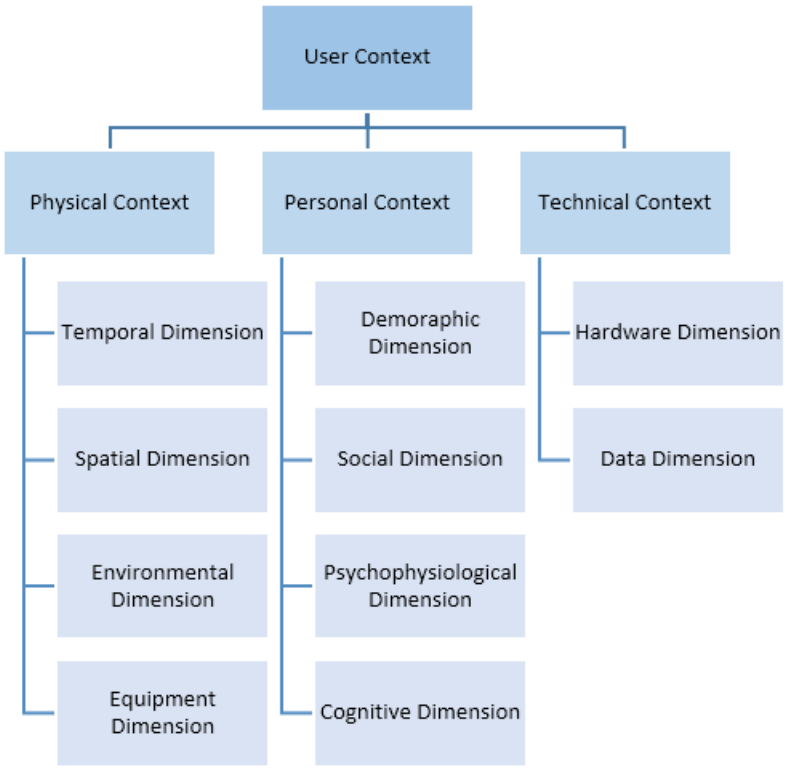


Figure 4.1: Context categorization in CARSs

we have completed and structured it in a hierarchical manner. Our hierarchical categorization (illustrated in Figure 4.1) has three principal categories of context: *physical*, *personal* and *technical context*. The user context is the union of these categories of context and their respective dimensions:

1. **The physical context** represents all aspects that can be influenced by the geographic position of the user. We have gathered four dimensions in this category:
   (a) *Temporal dimension* like the moment of the day, weekday/weekend, the season, events (birthday, new year, etc), etc,
   (b) *Spatial dimension* that can be represented by exact geographic positions (GPS coordinates, longitude/latitude) or nominal classes (at work, at home, in travel, etc),
   (c) *Environmental dimension* that can represent environmental characteristics like the temperature, the weather, the brightness or the noise level of the user's place, and/or the local situation of that place, like a war, a natural disaster, economic crisis, etc,
   (d) *Equipment dimension* that regroups all non-human entities (objects or spaces) that is around the user, like barbecue, home appliance, printer, garden/terrace, etc.
2. **The personal context** represents personal information about the user, and has four dimensions:

(a) *Demographic dimension* gathers information about the identity of the user (name, age, gender, nationality, etc),

(b) *Social dimension* is about the presence and the role of the persons around the user. Depending on the use case, it can be only the persons who accompanied the user while using the application (e.g. music recommendation in car), the persons with whom the user want to share the activity (e.g. going to theater with friends or cooking a recipe to share with friends), or going further by considering subtle relations like friends, family, colleagues, neighbors, etc (recommendation of persons or news on social networks),

(c) *Psychophysiological* [1] *dimension* represents psychological and physiological aspects of the user, like her state of mind, her mood, her degree of tiredness, etc.

(d) *Cognitive dimension* refers to the user experiences, her objectives, constraints, activity, etc.

3. **The technical context** gathers characteristics of the devices used by the user to access the application:

(a) *Hardware dimension* refers to the material used by the user to access the CARS, like the device, processors, network capacity, etc.

(b) *Data dimension* refers to manipulated data by the application, its type (text, audio, video, image, etc), sources, quality, validity period, exactitude, etc.

In section 7 (Table 7.1) we will show that we can clearly use this categorization in three different domains (movie, tourism and music), which validate its genericity and applicability to different domains.

Example 4.1.1. *Based on this proposition, available and interesting contextual information for a context-aware movie recommender system could be time, day type, season, location, weather, companion, emotion and mood, as illustrated in Figure 4.2.*

[1] "Combining or involving mental and bodily processes" (Merriam-Webster)

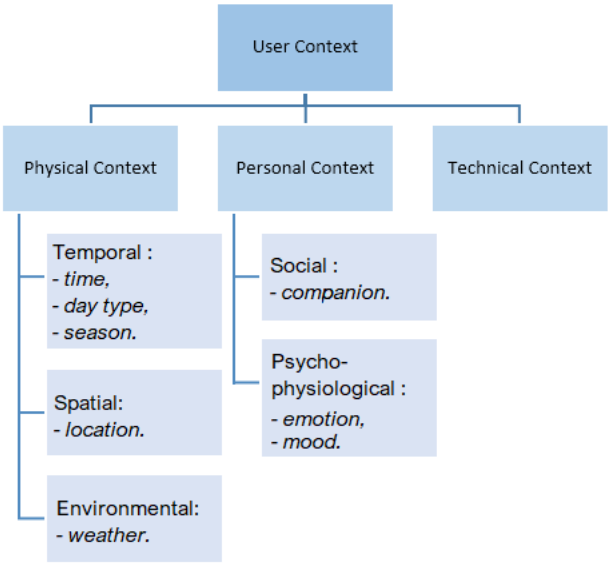


Figure 4.2: Example of context factors in movie recommendations

4.2 Conclusion

In this chapter, we proposed our viewpoint on the user context and the categorization of its different factors for CARSs. As mentioned in Chapter 2, several authors have proposed different categorizations for the context. Differently, our model of the user context is much richer than that of previous propositions, so we expect our model meets the requirements of larger spectrum of application domains.

In fact, contrary to Abowd et al. in [2] we include environmental, technical, psychological and cognitive context. The categorization of Adomavicius et al. in [3] misses demographical and equipment context, and the one of Nguyen [72] misses psycophysiological and equipment context. Differently from Zimmermann et al. in [117] we propose a categorization based on a different viewpoint of entity types, and a more clear and concrete proposition than the one of Baldauf et al. in [16].

It should be noted that depending on the application, some context factors can play a more important role than others. For example, in the case of recipe recommendation, factors like season, objects and tools around the user, and her cooking competence would be more important. While in music recommendation, activity and psychophysiology context would be more influencing.

As long as we know, our proposition of context factors is the most complete categorization. Referring to it can guaranty to do not missing interesting contextual information in the phase of identification and collection of context.

Chapter 5

From Multi-Dimensional to 2D Data Based on Context Influence

Traditional recommender systems (based on content-based and/or collaborative filtering techniques) have proved their effectiveness in different areas [83], including music, movies, places of interest, news, research articles, online courses, etc. But they have the limitation of not considering the contextual situation in which the user is, at the moment she wants to use the item. In fact this information can roughly influence her preferences for items [6]. As an example, when choosing a movie to watch, the user will have different preferences depending on whether she wants to watch the movie with a kid or with her partner. In this case, a context-aware recommender system (CARS), integrating such contextual information about the user in its process, can provide more relevant recommendations [4].
In this thesis, we propose a context-aware recommendation approach, that integrates contextual information about users by modeling it with the influence of context on ratings. This chapter presents the core of our context-aware recommender system, named *correlation-based filtering* module, and the next chapter describes its integration into the recommendation process within pre- and post-filtering approaches.

5.1 Correlation-Based Filtering

The main objective of our filtering approach is to transform the initial contextual dataset which is multi-dimensional ($user \times item \times context \rightarrow rating$) to a 2-dimensional dataset ($user \times item \rightarrow rating$) containing only the data pertaining to the target user context.
With respect to CARS state of the art [40], our approach is, in a sense, more user-centric, as we propose to compute the influence of context on ratings based on the item- or user-based PCC (Pearson Correlation Coefficient) [27] between context and ratings. The distinctive feature of using PCC allows us to catch more precisely this influence, and so to compute more accurate similarities between contexts, which is a crucial point in our pre/post-filtering process. In

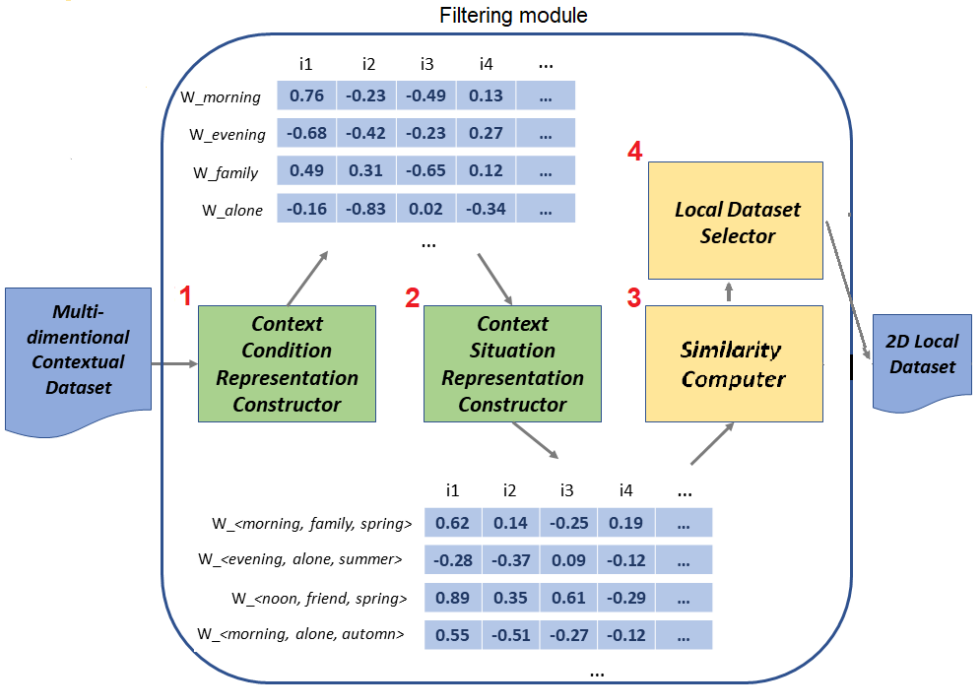


Figure 5.1: Correlation-based filtering process chain

addition, we use content information about items/users to improve our model, like, for instance, the category of a film, or the age or gender of users.

5.1.1 Methodology

The whole filtering process of *CBF (Correlation-Based Filtering)*, illustrated in Figure 5.1, can be decomposed in the following four steps: (1) building the context condition representations, (2) based on the results of the first step, building the context situation representations, (3) identifying similar contexts by computing the similarity between the target user context and the other contexts, (4) building a 2D local dataset which gathers the ratings given in similar contexts. Note that for the operations of some steps we propose different extensions:

Step 1 : As said before, to be able to find similar contexts, we need a strong representation of them. Indeed the user context is mostly expressed by nominal data (e.g. morning, spring, happy, etc.). One way to measure the semantic relations and similarities among contexts could be by the help of external resources like ontologies. However, this kind of resources are mainly domain-specific, and it is hard to find a generic ontology which can be used for any application domain. So, we propose a numerical representation of contexts based on their influence on ratings. We compute this influence by calculating the PCC of the rating variable r, and each context condition variable c_j, with $j \in [1, n]$, where n is the total number of context condition variables. We choose this correlation measure because in statistics, PCC (with values between -1 and 1) is widely used to measure the strength of association between two variables, and this

User	Item	Rating	Time	Companion	Season
U1	I1	3	morning	family	spring
U1	I2	1	evening	alone	summer
U2	I1	5	noon	friend	spring
U3	I6	4	morning	alone	autumn

(a) Original matrix

User	Item	Rating	Time= Morning	Time= Noon	Time= Evening	Time= Night	Companion =Alone	Companion =Family	Companion =Friends	Season= Summer	Season= Winter	Season= Spring	Season= Autumn
U1	I1	3	1	0	0	0	0	1	0	0	0	1	0
U1	I2	1	0	0	1	0	1	0	0	1	0	0	0
U2	I1	5	0	1	0	0	0	0	1	0	0	1	0
U3	I6	4	1	0	0	0	1	0	0	0	0	0	1

(b) Transformed matrix

Table 5.1: Transformed rating matrix

corresponds to what we want, since we want to catch the influence of context conditions on ratings.

In a context-aware environment, an observation will be the cross-tabulation of the variables of user, item, rating and m different context factors (e.g. *daytype, season, location, social, etc*). To apply PCC, we transform context factors into binary variables (see Table 5.1). So let us denote with $X_t = (u_t, i_t, r_t, c_{1t}, c_{2t}, ..., c_{nt})$ the t^{th} observation, which represents the evaluation r_t of the user u_t for the item i_t in the context situation $c_{1t}, c_{2t}, ..., c_{nt}$, where as said before, n is the total number of context conditions, and $c_{pt} = 1$ means that the p-th context condition is present in the context of the user, and $c_{pt} = 0$ means that it is not present. For instance, in the example of the Table 5.1, we have a notation from 1 to 5, where 1 means that the user did not really like the item, and 5 means that she liked it very much. The observation in this table X_1= *(U1, I1, 3, morning=1, noon=0, evening=0, night=0, alone=0, family=1, friends=0, summer=0, winter=0, spring=1, autumn=0)* means that *U1* had evaluated the item *I1* by *3*, when he consumed the item with his *family* in a *morning* of *spring*.

- ***Extension 1.1:*** To obtain a more precise influence of context on ratings, we compute the item- or user-based influence of context on ratings.

  (a) item-based: The reason why the item-based influence could be interesting is that the context can influence the ratings differently, according to items. For example in the case of points of interest recommendation, a snowy weather will have a positive influence on some winter sport centers, but a negative influence on natural parks. This is why it is important to compute this influence according to items.

  So the item-based correlation between the ratings and a context condition c_j is calculated as follows in Equation 5.1.

$$w_{c_j i} = PCC_i(r, c_j) = \frac{\sum_{k \in K}(r_k - \overline{r_i})(c_{jk} - \overline{c_{ji}})}{\sqrt{\sum_{k \in K}(r_k - \overline{r_i})^2}\sqrt{\sum_{k \in K}(c_{jk} - \overline{c_{ji}})^2}} \tag{5.1}$$

In our PCC calculation, the summations are taken over K, which is the set of observations $X_k = (u_k, \mathbf{i}, r_k, c_{1k}, c_{2k}, ..., c_{nk})$ where the item is equal to i. $\overline{r_i}$ represents the mean of ratings done for the item i, and $\overline{c_{ji}}$ represents the mean value of the context condition c_j over observations where the item is equal to i.

(b) user-based: It has to be noted that the user-based influence on ratings is also interesting. Indeed, we can say that the influence of context on ratings also depends on users, and will differ from one user to another. For example, a "family person" could like to practise activities with her family, whereas another person may not like this and prefer to practise activities with her friends. So the social context will influence differently these two persons.

In this case the user-based correlation between the ratings r and a context condition c_j is calculated as follows in Equation 5.2.

$$w_{c_j u} = PCC_u(r, c_j) = \frac{\sum_{k \in K}(r_k - \overline{r_u})(c_{jk} - \overline{c_{ju}})}{\sqrt{\sum_{k \in K}(r_k - \overline{r_u})^2}\sqrt{\sum_{k \in K}(c_{jk} - \overline{c_{ju}})^2}} \tag{5.2}$$

where K is the set of observations $X_k = (\mathbf{u}, i_k, r_k, c_{1k}, c_{2k}, ..., c_{nk})$ with user u. $\overline{r_u}$ is the mean of the ratings given by the user u, while $\overline{c_{ju}}$ is the mean value of the context condition c_j over observations for user u.

Note that in the following sections/chapters, the suffix *IB* refers to the item-based version of our approach, and *UB* refers to the user-based version.

- ***Extension 1.2:*** Based on the above explanations, we can build a vector representation for each context condition, by one of the two followings methods:

  (a) non-clustered: The size of the context condition representation vector will be the total number of items or users, and the values of this vector (between -1 and 1) are equal to the item- or user-based PCC between the rating vector and the context condition vector.

  (b) clustered: In real world recommendation problems, the total number of items/users is often very large, and the correlation calculation would be computationally consuming. To overcome this computational cost we propose to cluster items/users into a limited number of groups, and to compute the influence based on clusters of items/users. Also, as the number of known ratings is often very poor, the clustering strategy could help the PCC to catch more precisely the correlation between context conditions and ratings. Indeed as the available data for two variables increases, possibilities of catching real correlations by means of PCC increases as well. By clustering the items/users, we regroup the ratings of multiple items/users in a single rating variable, and so the PCC will be computed on richer variables.

Figure 5.2: Examples of representation of cluster-based context condition by PCC values $\in [-1, 1]$ (Step 1)

This clustering could be done by different clustering approaches, and based on the available static information about items' characteristics (e.g. genre, year of production, etc)/users' characteristics (e.g. age, sex, etc), or directly based on the ratings.

Figure 5.2 illustrates some examples of the resulting cluster-based context condition representations. In this example we clustered the items in 4 different clusters. In the *morning* representation example, the value 0.54 refers to the positive influence of *morning* on the ratings of the first items cluster (computed by PCC), while the value -0.91 refers to its strong negative influence on the ratings of the third items cluster.

Note that in the following sections/chapters, the letter C in the suffix $\underline{C}IB$ and $\underline{C}UB$ refers to the cluster-based versions of the approach.

Step 2: We can now represent each context situation based on its composing context conditions (step 2 of Figure 5.1).

- ***Extension 2.1:*** Here we propose two different ways to build this representation:

  (a) aggregation: Each context situation can be represented by a vector with values equal to the mean aggregation of the values of its corresponding composed context conditions (also used by Codina et al. in [40]). The Figure 5.3 illustrates some examples of context representations based on the aggregation technique. For example, the value 0.25 of the first context situation $< morning, family, spring >$ refers to the influence of this context situation on the ratings of the first items clusters, and is equal to the mean value of the three corresponding values (for cluster 1) of the vector representation of *morning*, *family* and *spring* (in Figure 5.2).

Figure 5.3: Examples of representation of context situation by aggregation (Step 2)

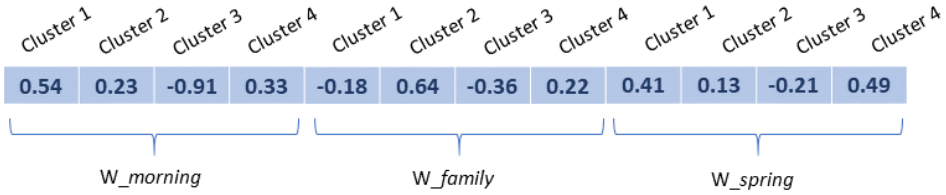


Figure 5.4: Example of the representation of the context situation $W_{\langle morning,family,spring\rangle}$ by concatenation (Step 2)

(b) concatenation: By a much deeper analysis of the aggregation technique proposed by Codina et al. in [40], we have realized that taking mean value over composed context condition correlations, can neutralize the influence of each context condition; this is as if we ignore these contextual information in the recommendation process. Let us explain this fact with an example: imagine that a context situation is composed of three context factors *time*, *companion* and *season*, and one wants to compute the similarity between two context situations C1: $< morning, family, spring >$ and C2: $< evening, alone, summer >$. Suppose that computing the PCC of each context condition for cluster 1, gives a large negative value (near -1) for *morning* and *alone*, a large positive value (near 1) for *family* and *evening* and a neutral value (near 0) for *spring* and *summer*. Suppose this kind of pattern is repeated for the other clusters. In this case, aggregating over the values of context conditions for each cluster (for C1: $(-1+1+0)/3 = 0$ and for C2: $(1+(-1)+0)/3 = 0$), gives similar values for both context situations $C1$ and $C2$. Hence it yields an extremely high similarity, contrary to what was expected.
To avoid this, we propose to represent context situation by a larger vector, by concatenating vectors of composed context conditions, instead of aggregating their values (see Figure 5.4).

Note that in the following sections/chapters, the suffix AG in the acronym of the approach (e.g. *CBPF-CIB-AG*) refers to the aggregation extension, and the suffix *CN* (e.g. *CBPF-CIB-CN*) refers to the concatenation extension.

Step 3: Now that we have represented each context, we can find the contexts most similar to the target context s^* by computing the similarity between every possible context s and the target context s^*.

- ***Extension 3.1:*** This similarity can be computed by different measures. Here we propose the two following ones:

  (a) Cosine similarity: we can obtain the similarity between the vector representations $\overrightarrow{w_s}$ and $\overrightarrow{w_s*}$ by the mean of cosine similarity measure illustrated in Equation 5.3, where d is the dimension of vector $\overrightarrow{w_s}$.

$$sim(s, s^*) = cosine(\overrightarrow{w_s}, \overrightarrow{w_{s^*}}) = \frac{w_s^T w_{s^*}}{\sqrt{\sum_{i=0}^{d} w_{s,i}^2}\sqrt{\sum_{i=0}^{d} w_{s^*,i}^2}} \tag{5.3}$$

  The cosine similarity is very used in recommender systems because of its speed and its efficiency for sparse data. But this measure is scale-invariant and in our application this property could affect the expected results. Let us explain it throw an example: Consider three different contexts $C1[0.2, -0.1]$, $C2[0.2, 0.1]$ and $C3[1, 0.5]$ (the only two dimensions of the contexts are chosen for making the example simpler). We can see that the two contexts $C1$ and $C2$ have comparable small values (referring to their influence on ratings), contrary to $C3$ which have much larger values. So in theory we expect a larger similarity between $C1$ and $C2$, and a much smaller one between $C1$ and $C3$. But by the cosine similarity measure, these two similarities will have exactly the same value, because the $C3$ is a scaled vector of $C1$ ($C3 = 5 \cdot C1$). Indeed the cosine similarity between two vectors corresponds to the angle between them. As you can see in Figure 5.5 which is a geometrical illustration of our example, the angle α between $C1$ and $C2$ corresponds also to the angle between $C1$ and $C3$.
  So in the cases where scaling vectors change the meaning of the vector and has to be considered, we have to keep in mind that the cosine similarity measure will ignore this scaling difference. In our CBF approach, the values of context vector representations illustrate the influence of context conditions on ratings. So a small value like 0.2 would mean a small influence and the large value 1 means a maximum influence. As we want to catch this difference in our similarity computation, the cosine similarity measure could not be the best option.

  (b) Euclidean similarity: To overcome the cosine similarity weakness, we propose to compute the similarity between two contexts s and s^* by the euclidean similarity which corresponds to the inverse of the euclidean distance between their vectors (Equation 5.4):

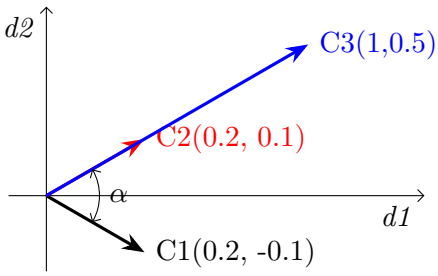


Figure 5.5: Geometrical illustration of cosine similarity between different vectors

$$sim(s, s^*) = \frac{1}{euclidean(\overrightarrow{w_s}, \overrightarrow{w_{s^*}})} = \frac{1}{\sqrt{\sum_{i=0}^{d}(w_{s,i} - w_{s^*,i})^2}} \tag{5.4}$$

Step 4: Now we can transform the multidimensional contextual dataset to a 2D dataset. For that, we select the ratings given in the similar context situations, found in the Step 3, and make a 2D *local dataset*.

Alternative for correlation computation: The Figure 5.1 and the above explained steps detailed the filtering process of our *CBF* approach. Note that in this model, in the first step, we used the PCC to compute the influence of context on ratings in order to represent the context. Indeed the PCC value between the rating variable r and each context condition variable c_j refers to the influence of c_j on the ratings. Instead of the PCC we can think of other methods to catch this influence. Another proposition could be to model this influence by the difference between the mean of ratings given when the context condition is present and those given when this one is absent. Equation 5.5 illustrates this technique that we named *mean deviation* technique. For example, in the case of the context condition *morning*, the difference between the mean values of ratings given when it was morning and when it was not would illustrate the influence of the context condition *morning*.

$$w_{c_j i} = \frac{\bar{r}_{ic_j=1} - \bar{r}_{ic_j=0}}{r_{max} - r_{min}} \tag{5.5}$$

To normalize this value in order to be in the range of [-1, 1] we divided it by the difference between the maximum and minimum possible values of ratings.

In this chapter we have kept the PCC proposition as the main approach for computing the influence of contexts on ratings and representing the context vector. But in the experimentation phase we also test the outlined above proposition (*mean deviation*), to compare the performances of these different methods.

5.1.2 Pseudo-code

In this section, we detail the pseudo-code of our *correlation-based filtering* algorithm, in order to guarantee its reproducibility.

The Algorithm 1 describes the *CBFiltering* function with the objective of transforming the initial multi-dimensional dataset to a 2D local dataset, which regroups the ratings given in contexts similar to the context of the target user. This filtering function is used in the pre-filtering approach (*CBPF*), as well as the post-filtering approach (*CBPoF*). In the former, the module is plugged before the traditional recommender system, while in the latter, it is plugged after it (we detail *CBPF* and *CBPoF* in the next chapter).

In this algorithm, S is the 2D transcript of all context situations present in T, and M is a 2D matrix, with users or items ids in rows and static characteristics of users or items in columns. Note that here what we call static characteristics are different from contextual information. The contextual information describes the situation of the user at a specific time, it is dynamic and will potentially differ from time t to time $t+1$ (like location, weather, etc). But static characteristics are some information about users or items which are static and do not change over time (like user's gender, movie's director, etc). Here we use this information for clustering users/items, though in cases such information is not available, the clustering could be done directly based on ratings.
In the filtering process, if the chosen representation model is cluster-based, we first apply a clustering on items or users, in order to replace their ids by the corresponding cluster ids in the contextual dataset T (lines 1 to 4). In line 5 we binarize the matrix S to be able to compute the correlation between context conditions and ratings (see the binarized example of Table 5.1b). Then (lines 6 to 10), for each context condition we create a vector representation, following an iterative process described in Algorithm 2 (*createContextConditionRepresentation*). Based on these context condition representations, the vector representations of the target user context (line 11) and other contexts of the dataset (line 13) are created (detailed in Algorithm 3, *transformContextRepresentation*). Then, in line 14, the similarities between the target user context s^* and other contexts s_k are computed by a similarity measure like the cosine or the euclidean similarity. The most similar contexts to the target user one, which are those whose similarity is greater than the threshold t, are identified (lines 15 to 17), and finally their corresponding data are selected as a local dataset (line 19).

Algorithm 2 creates the vector representation of each context condition (step 1 of the process chain in Figure 5.1). The size of these vectors depends to the representation model r, and is equal to the total number of users (in case of *UB*), items (in case of *IB*), users clusters (in cases of *CUB-AG* or *CUB-CN*) or items clusters (in cases of *CUI-AG* or *CUI-CN*). Lines 1

Algorithm 1 : CBFiltering

Input : T: multi-dimensional tensor of contextual rating data,
u^*: the target user,
s^*: the context situation of the target user,
M: matrix of items/users characteristics,
S: matrix of context situations (the 2D transcript of all the context situations present in T),
t: similarity thresholds,
r: model of context representation
$\in \{IB/UB, CIB-AG/CUB-AG, CIB-CN/CUB-CN\}$.

Result : L: local dataset of ratings given in contexts similar to the context s^* of the user u^*.

if *r is a cluster-based model* **then**
  $clusters \longleftarrow clusteringItemsOrUsers(M)$;
  $T \longleftarrow replaceIdsByClusterIds(T, clusters)$;
end
binarization(S);
foreach $f \in contextFactorsOf(S)$ **do**
  foreach $c \in contextConditionsOf(f)$ **do**
    $ccRepresentations[c] \longleftarrow createContextConditionRepresentation(c, r)$;
  end
end
$s^* \longleftarrow transformContextRepresentation(s^*, r, ccRepresentations)$;
foreach $s_k \in S$ **do**
  $s_k \longleftarrow transformContextRepresentation(s_k, r, ccRepresentations)$;
  $sim_k \longleftarrow computeSimilarity(s_k, s^*)$;
  if $sim_k \geq t$ **then**
    $S' \longleftarrow addToSimilarContext(s_k)$;
  end
end
$L \longleftarrow createLocalDataset(T, S')$;

to 9 identify the set (e) of users/items or clusters of users/items based on r. For each element of e, we create an array of the corresponding ratings, *ratingArray* (line 11), and an array of the corresponding context conditions values, *ccArray* (line 12). The values of the vector representations of each context condition are equal to the PCC between these two arrays of the corresponding observations in T (line 13 and 14).

Algorithm 2 : createContextConditionRepresentation

Input : T: multi-dimensional tensor of contextual rating data,
c : context condition,
r: model of context representation
$\in \{IB/UB, CIB-AG/CUB-AG, CIB-CN/CUB-CN\}$.

Result : $ccRep_c$: vector representation of the context condition c.

```
1  if r is IB then
2  |   e = set of items ids
3  else if r is UB then
4  |   e = set of users ids
5  else if r is CIB − AG||CIB − CN then
6  |   e = set of items cluster ids
7  else
8  |   e = set of users cluster ids
9  end
10 foreach io ∈ e do
11 |   ratingArray_io ⟵ array of ratings given to/by io in T;
12 |   ccArray_io ⟵ array of the values of context condition c of io observations in T;
13 |   pcc_io = PearsonCorrelationCoefficient(ratingArray_io, ccArray_io);
14 |   ccRep_c[io] = pcc_io
15 end
16 return ccRep_c
```

Algorithm 3 illustrates the step 2 of the recommendation process chain and has as objective to create a numerical vector representation for each context situation, based on the results of Algorithm 2. Two different techniques are proposed: by the *aggregation* technique (lines 2 to 12), we create a numerical vector of the same size of the context conditions vectors, and the value of each cell will be the mean value of the corresponding values of its composed context conditions. If the *concatenation* technique is chose (lines 13 to 18), we create a numerical vector by concatenating the vector representations of its composed context conditions.

The next section presents a variation of our *CBF* approach which takes into account the different impact degrees of context factors.

Algorithm 3 : transformContextRepresentation

Input : s: context situation

S: matrix of context situations (the 2D transcript of all the context situations present in T)

$ccRep$: context condition representations

r: model of context representation

$\in \{IB/UB, CIB - AG/CUB - AG, CIB - CN/CUB - CN\}$

Result : $ctxRep_s$: representation of the context s

```
switch technique of r do
    case AG do
        foreach io ∈ items/users or cluster of items/users do
            foreach f ∈ contextFactorsOf(S) do
                c ⟵ contextConditionValue(f, s);
                pcc ⟵ ccRep_cc[io];
                pccSum+ = pcc;
                counter++;
            end
            ctxRep_s[io] ⟵ pccSum ÷ counter;
        end
    end
    case CN do
        foreach f ∈ contextFactorsOf(S) do
            c ⟵ contextConditionValue(f, s);
            ctxRep_s ⟵ concatenate(ctxRep_s, ccRep_c);
        end
    end
end
return ctxRep_s;
```

5.2 Context Relevance

The integration of contextual information aims to improve the performance of the RS [77]. In theory it could be assumed that the complete set of contextual information is significant and all context factors are equally important. But in real-world cases, depending on the application, some context factors could have a more important impact than others. Considering some context factors can even may cause more noise than effective information. For example, in the case of music recommendation, the activity of the user is more impacting her preferences than her location. Also in this example a context factor like the season, which does not really impact the listening preferences of the user, can produce more noise than information in the recommendation process. So, identifying relevant context factors and/or considering their impact degree in the recommendation process is important to avoid performance degradation. Furthermore, we have to note that the automatic acquisition of certain context factors (like the user's mood or her social context) is still almost impossible and has to be specified directly by the user. So this relevant context factors identification would also minimize the user's effort to specify her context, which is not negligible [34].

In this thesis we propose different methods to consider the context relevance in its representation. We regrouped them in three categories of approaches: *weighting*, *filtering* and *hybrid* approaches:

I. ***weighting*** approach, where we compute a weight w_{f_i} for each context factor f_i and use it to weight the corresponding condition vector in the context representation.
To obtain this weight, we first compute the correlation between the ratings r and each context condition c_j (based on PCC) as follows:

$$w_{c_j} = PCC(r, c_j) = \frac{\sum_{o \in O}(r_o - \overline{r})(c_{jo} - \overline{c_j})}{\sqrt{\sum_{o \in O}(r_o - \overline{r})^2}\sqrt{\sum_{o \in O}(c_{jo} - \overline{c_j})^2}} \tag{5.6}$$

We consider the contextual dataset as a set of observations O, where each observation $o = (u_o, i_o, r_o, c_{1o}, c_{2o}, ..., c_{no})$ is composed of the evaluation r_o of the user u_o for the item i_o in the context situation $c_{1o}, c_{2o}, ..., c_{no}$, where n is the total number of context conditions. In the Equation 5.6 the summation is taken over all observations (O) of the dataset. r_o and c_{jo} are respectively the value of the rating and the value of the context condition c_j for the observation o. $\overline{r}$ represents the mean of ratings over all observations, and $\overline{c_j}$ represents the mean value of the context condition c_j over all observations.
Then we propose to compute the weight corresponding to each context factor f_i by one of the two followings methods:

(a) We can attribute to each context factor f_i, a weight equal to the average of the absolute value of its possible context conditions c_j (equation 5.7).

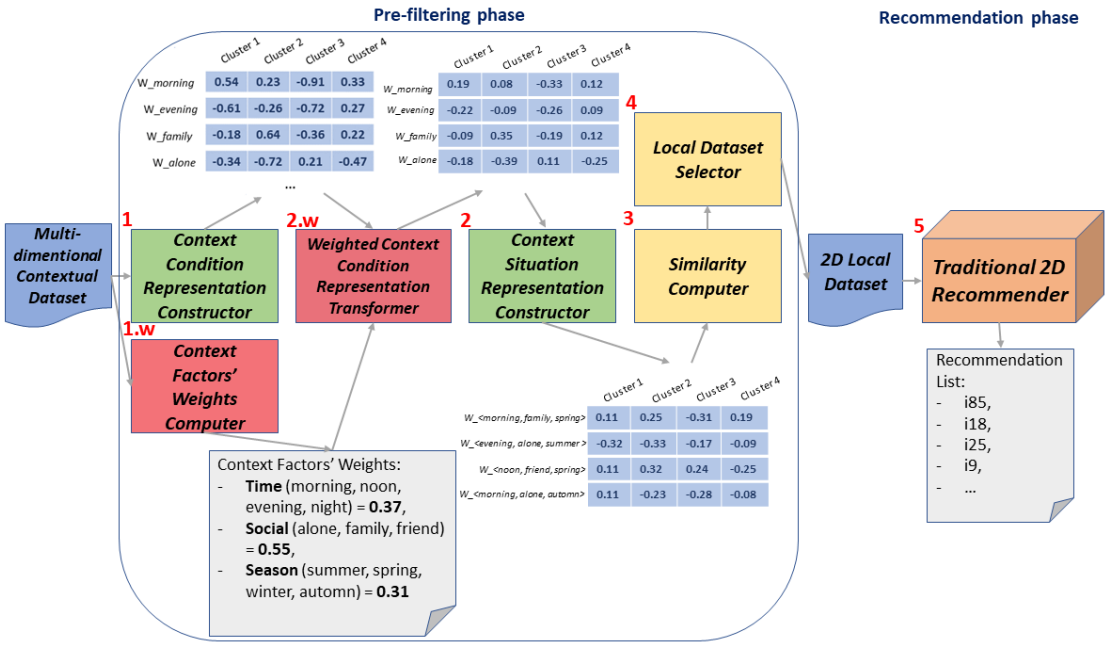


Figure 5.6: Process chain with the weighting module

$$w_{f_i} = \frac{\sum_{c_j \in F_i} \mid w_{c_j} \mid}{|F_i|} \tag{5.7}$$

Where F_i is the set of possible context conditions of the context factor f_i. As an example, for the context factor $f_1 = time$, the set F_1 of its possible context conditions would be $\{morning,\ noon,\ evening,\ night\}$.

We consider the absolute value of w_{c_j} because here, what is interesting for us is the strength of the impact of context factors, and not their positive or negative influence.

(b) By exploring the datasets, we realized that in some cases, for each context factor, all of possible context conditions do not occur equally (please refer to the diagram of Figure 7.1, which illustrates the context conditions frequencies of a real-world dataset). Because of that, we propose a second notion of weight where we take into account the number of occurrences of each context condition in this aggregation. Therefore we will have the following weight equation:

$$w_{f_i} = \frac{\sum_{c_j \in F_i} \frac{\#O_{f_i=c_j}}{\#O_{f_i \neq unknown}} \mid w_{c_j} \mid}{|F_i|} \tag{5.8}$$

Where we multiply the weight w_{c_j} by the ratio of the number of observations for which the context condition c_j is present to the total number of observations where the context factor f_i is known.

Figure 5.6 illustrates the modified filtering process chain which includes the weighting module (between the initial steps *1* and *2*): after the first step where we have repre-

sented each context condition by a vector representation, just before aggregating or concatenating them to obtain the context representation (step *2*), we add step *1.w* where we compute the weights of all context factors w_{f_i} for $i \in [1, m]$ (where m is the total number of context factors), and step *2.w* where we multiply the values of each context condition vector by its corresponding w_{f_i}. The subsequent steps (*2, 3* and *4*) remain the same.

Example 5.2.1. *In the case of our filtering module (illustrated in Figure 5.6, which will be used in a context-aware movie recommender system), where we have three context factors time, social and season, the resulting weights of the step 1.w for these context factors are respectively* 0.37*,* 0.55 *and* 0.31*. So, in step 2.w, to obtain the weighted representations of context conditions, we multiply the values of the vectors* $W_{morning}$*,* W_{noon}*,* $W_{evening}$ *and* W_{night} *(context conditions of time) by* 0.37*, the values of vectors* W_{alone}*,* W_{family} *and* W_{friend} *(context conditions of social) by* 0.55*, and the values of the vectors* W_{summer}*,* W_{spring}*,* W_{winter} *and* W_{automn} *(context conditions of season) by* 0.31*.*

II. ***filtering*** approach is a more strictly way to consider context relevance, where in a binary configuration we ignore the non significant context factors and exclude their vectors from the context representation. Here we propose two different methods to identify these non significant context factors:

   (a) The first strategy is based on a weight threshold, defined empirically. In the context representation, we keep only context factors for which we have a weight greater than a certain threshold.

   (b) Another strategy is based on a causal inference test [89], and the relevant context factors are identified as follows: we perform multiple execution of our CARS, where each time we ignore one of the context factors (f_i). If the performance do not change or increase, we can conclude that considering this specific context factor have no influence or negative influence on the performance (noisy context factor). And if the performance decrease, it means that this context factor is a relevant factor and should be considered in the recommendation process to have better results. By this way we identify the important and relevant context factors and consider only these ones in the context representation.

III. ***hybrid*** approach which is a combination of the two former approaches, where like the *filtering* approach, we identify the relevant context factors to keep only these ones, and like the *weighting* approach we weight their corresponding vectors in the context representation.

In the literature some propositions focus on context factor scale like [74], and some others on context condition scale like [39]. By the *hybrid* approach we combine these two levels in our relevance consideration. In fact we obtain for each context factor a weight based on the weights of its corresponding possible context conditions, and then we filter on a factor level the irrelevant context factors.

Note that in the following sections, *weighting_a* refers to the approach which uses Equation 5.7 to compute the relevance of each context factors, and *weighting_b* refers to the one which uses the Equation 5.8 instead. *filtering_a* refers to the filtering method based on a weight threshold, and *filtering_b* refers to the filtering method based on causal inference test. Also, as the Table 5.2 show, the *hybrid_a* refers to the method combining the *weighting_b* and *filtering_a* methods, and *hybrid_b* refers to the combination of methods *weighting_b* and *filtering_b*. (It should be noted that in our *hybrid* combinations, we only use the *weighting_b* methods, because it uses a more complete weighting formula (Equation 5.8) than *weighting_a*.)

Approach	**weighting_a**	**weighting_b**	**filtering_a**	**filtering_b**
hybrid_a		x	x	
hybrid_b		x		x

Table 5.2: Hybrid combinations

5.3 Conclusion

In this chapter we present our *Correlation-Based Filtering* approach, to transform the multi-dimensional contextual dataset to a 2D dataset based on the contextual information. This filtering approach will be used in the recommendation process of a CARS (which will be discussed in the next chapter). In this approach, we propose a vector representation of the context, based on its influence on ratings. We model this influence based on the correlation between the ratings and the context, calculated by the *PCC* or a *mean deviation* method. This numerical vector representation of the context allows us to catch the similarities between different contexts, and identify the most similar contexts to the target user one. Then we can filter only ratings done in these similar contexts, and create a 2D matrix which reassemble these selected ratings.

We offer different configurations of this approach for the context representation:

- item-based vs. user-based,
- clustered vs. non clustered,

- aggregation vs. concatenation technique,
- different correlation techniques: PCC vs. the mean deviation,
- different similarity measures: Cosine vs. Euclidean similarities,
- different techniques to take into account the effect of context relevance: *weighting*, *filtering* or *hybrid* techniques.

The effectiveness of each configuration will be discussed in the experimental chapter (Chapter 7).

In the next chapter, we will propose two context-aware recommendation methods, which integrate the context of the users in their recommendation process by the mean of the *correlation-based filtering* module.

Chapter 6

Correlation-Based Context-Aware Recommendation

In this chapter we present two different approaches for the integration of our *correlation-based filtering* module in a recommendation process to produce context-aware recommendations.
We propose to plug our filtering module, either in a pre-filtering configuration or a post-filtering one: we named the former approach *Correlation-Based Pre-Filtering (CBPF)* and the latter *Correlation-Based Post-Filtering (CBPoF)*.
Pre-filtering approaches are a particular class of CARSs (context-aware recommender systems) based on the idea of pre-processing contextual data so as to tune the input of a given (traditional) recommender system (RS) in order to increase its effectiveness. While post-filtering approaches firstly apply a traditional recommendation technique on the data by ignoring the context, and then contextualize the resulting recommendation list by filtering or re-ordering the recommended items.

6.1 Correlation-Based Pre-Filtering

A recommendation problem is often viewed as a matrix/tensor completion problem. A recommender system firstly estimates missing ratings, and then recommends to each user her corresponding items with highest estimated ratings.
In the case of pre-filtering CARSs, we want to integrate the user contextual information into the estimation phase of missing ratings. Our *correlation-based pre-filtering* approach, like the *reduction-based pre-filtering* approach [4], makes the hypothesis that a user will rate an item similarly in two similar contexts. Based on this hypothesis, to recommend an item to a user in a specific context, we can identify ratings given in contexts similar to this specific context, and apply a traditional 2D recommendation technique on this selection.
One of the main challenges in this procedure is to identify correctly the similar contexts to the target user context. As in general, the context is mainly categorical data (like time= {morning,

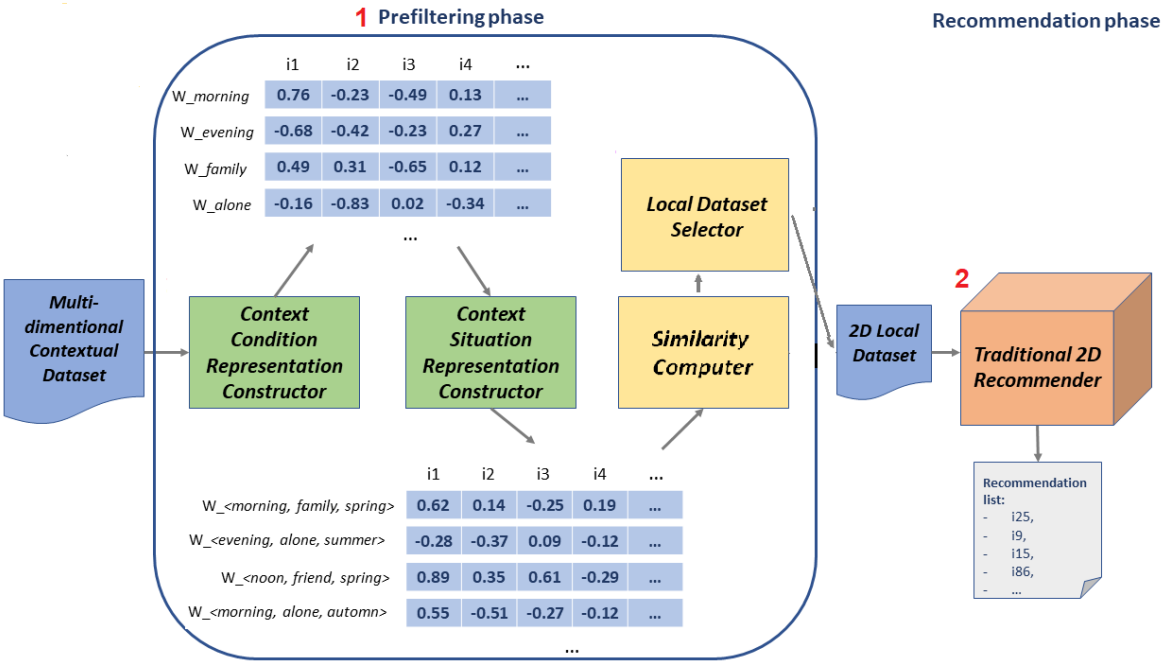


Figure 6.1: Correlation-based pre-filtering process chain

noon, evening}, social= {alone, family, friends}, etc), the automatic similarity computation between two context instances is not trivial. In a specific domain, we can ask experts to determine and quantify the similarity between different context components or use ontology-based approaches. But as we want a generic approach which can be applied to any domain, we use the *Correlation-Based Filtering* module (described in the previous chapter), which is a data-driven approach with a numeric representation of the context, to compute context similarities automatically. We propose a pre-filtering configuration of this module, that we named *Correlation-Based Pre-Filtering (CBPF)*.

6.1.1 Methodology

Figure 6.1 illustrates the steps of the correlation-based pre-filtering method to produce context-aware recommendations.

Step 1: Transforming the multidimensional contextual dataset to a 2D dataset based on the context of the target user, by applying the correlation-based filtering module (described in section 5.1.1). This filtering module can allow to reduce the very high sparsity of the initial contextual dataset, which impact the recommendations accuracy.

Step 2: We then apply a traditional 2D recommendation technique [50] on this selection of ratings (*local dataset*), to obtain contextual recommendations. As our approach is generic and parametric, any kind of 2D recommendation technique can be applied, but here we suggest the *BiasedMF* technique [55], which is a good performing matrix factorization technique.

Our approach has two main technical advantages: (a) it is easily pluggable to any existing 2D recommender system: indeed this feature allows companies, which want to take benefit of the available contextual information in their recommendation process, to re-use their existing recommendation engine and plug the pre-filtering module prior to it. (b) it is also configurable: we propose some alternatives for the different parts of the algorithm (different context representations, similarity measures, clustering algorithms, correlation techniques, etc. detailed in the previous chapter). Depending on the dataset and the available resources, we can configure differently these features.

6.1.2 Pseudo-code

This section details the pseudo-code of *CBPF* in order to guarantee the reproducibility of the algorithm.

Algorithm 4 : CBPF

Input : T: multi-dimensional tensor of contextual rating data,
u^*: the target user,
s^*: the context situation of the target user,
M: matrix of items/users characteristics,
S: matrix of context situations (the 2D transcript of all the context situations present in T),
t: similarity thresholds,
r: model of context representation
$\in \{IB/UB, CIB-AG/CUB-AG, CIB-CN/CUB-CN\}$.

Result : $recomList$: list of recommended items for user u^* in context s^*.

1 $L \longleftarrow CBFiltering(T, M, S, t, r, u^*, s^*)$;
2 $recomList \longleftarrow 2DRecommender(L)$;

The Algorithm 4 shows the procedure of CBPF to recommend relevant items to user u^* in context s^*. It takes as input a set of contextual rating data of users ($T : user \times item \times context \longrightarrow rating$), a set of items/users characteristics (M), the minimum similarity required to designate a context as similar to the target user one (t), and the intended context representation (r).
In this algorithm, first (line 1), the multi-dimensional dataset (T) is transformed to a 2D dataset (L) by the *CBFiltering* module (described later in Algorithm 1). Then (line 2), a traditional 2D recommender system is applied on this 2D dataset and in result we would have a recommendation list for the target user in her context. In our experiments we used one of the best performed 2D recommendation technique: BiasedMF, a matrix factorization method proposed by Koren in [55] (see section 3.3.1 for the details of the algorithm). But as our approach is generic and configurable, any other type of 2D recommender system (e.g. itemKNN, userKNN,

SVD++, etc. [91]) can be plugged to our pre-filtering module.

6.2 Correlation-Based Post-Filtering

As mentioned before, the contextual information of the users can be integrated in the recommendation process in three different ways: *pre-filtering*, *post-filtering* and *contextual modeling* methods. Being positioned in an industrial context, where a traditional recommender system already exists, we were first interested in *pre-filtering* technique which is mostly used in the literature, and proposed a *Correlation-Based Pre-Filtering (CBPF)* approach based on the correlation between contexts and ratings. Since there is not a real winner between these approaches, we propose a post-filtering adaptation of our approach, called *Correlation-Based Post-Filtering (CBPoF)*. Our main motivation is that there is very few research on the comparison of pre- and post- filtering approaches in CARS. Panniello et al. demonstrate in [78] that in some cases post-filtering can beat pre-filtering technique. In our case, as we plugged our *correlation-based filtering* module in a *pre-filtering* process, we would like to plug it in a *post-filtering* one, in order to compare the performances of these two families of CARS based on our correlation-based approach.

6.2.1 Methodology

Figure 6.2 illustrates the post-filtering process chain to recommend a list of appropriate items to the user u^* who is in the context s^*. Like all post-filtering approaches, first we transform the multi-dimensional dataset to a 2D context-free dataset by ignoring the contextual information. Then, we apply a traditional recommendation technique to this context-free dataset, for which we obtain a set of predicted ratings for user u^* ($\hat{r}_{u^*,i}$). Now, we propose to identify the contextual neighbors of the target user u^*, that we call G. G is the set of similar users to u^* in her context s^*. This neighborhood is identified by computing the cosine similarity between the target user u^* and all users who have rated items in contexts similar to s^* (here we used the cosine similarity which is mostly used in collaborative filtering techniques to determine the similarities between users, but other similarity measures can as well be used). In this process, the similar contexts identification is done based on our *correlation-based filtering* approach, also used in *CBPF* (steps 1 to 4 described in section 6.1.1).
Now we contextualize the predicted ratings based on the distribution of ratings of G. And propose a list of recommendations based on these contextual predicted ratings ($\hat{r}_{u^*,i,s^*}$).

The equation 6.1 illustrates the convex combination used to contextualize the context-free predicted rating $\hat{r}_{u^*,i}$. The predicted rating of item i by user u^* in context s^* ($\hat{r}_{u^*,i,s^*}$) is the weighted sum of the context-free predicted rating $\hat{r}_{u^*,i}$ and the mean of ratings done by the neighbors of user u^*. As said before, G is the set of u^*'s neighbors in contexts similar to s^*, so

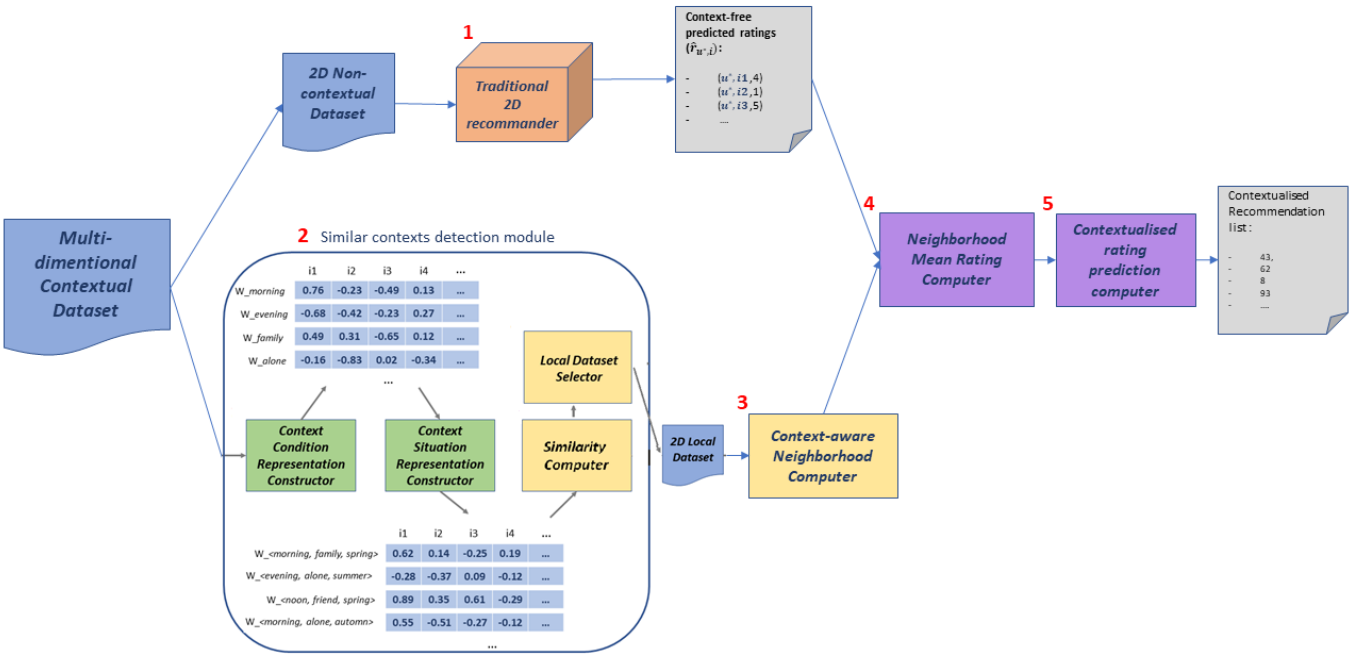


Figure 6.2: Post-filtering context-aware recommendation process chain to recommend relevant items to user u* in context s*

$r_{g,i}$ refers to the rating of the neighbor $g \in G$ for item i.

$$\hat{r}_{u^*,i,s^*} = \alpha \times \hat{r}_{u^*,i} + (1-\alpha) \times \frac{\sum_{g \in G} r_{g,i}}{|G|} \tag{6.1}$$

A context-aware recommender system has to juggle with two parameters: the personalization and the contextualization of its predictions. The coefficient α which has a value between 0 and 1, allows us to set up these two parameters: by increasing its value we rise the impact role of the context-free prediction and so the personalization part, while decreasing it will give more importance to the contextualization part. In the former, the recommendations are focused on the similarity of preferences with other users, while in the latter, we put the stress on the impact of user's context on her preferences.

6.2.2 Pseudo-code

Algorithm 5 describes the recommendation process of our post-filtering approach, *CBPoF*, to recommend items to user u^* in context s^*. Its inputs are a set of contextual rating data of users (T : $user \times item \times context \longrightarrow rating$), a set of items/users characteristics (M), the minimum similarity required to designate a context as similar to the target user one (t), the intended context representation (r) and the coefficient α.

In this algorithm, first (line 1), the context dimensions of the initial dataset T are ignored by

Algorithm 5 : CBPoF

Input : T: multi-dimensional tensor of contextual rating data,
u^*: the target user,
s^*: the context situation of the target user,
M: matrix of items/users characteristics,
S: matrix of context situations (the 2D transcript of all the context situations present in T),
t: similarity thresholds,
r: model of context representation
$\in \{IB/UB, CIB-AG/CUB-AG, CIB-CN/CUB-CN\}$,
α: coefficient of the contextualizing equation.

Result : $recomList$: list of recommended items for user u^* in context s^*.

1 $R \longleftarrow mTo2Dimensions(T)$;
2 $PR \longleftarrow 2DRecommender(R)$;
3 $similarCtxDS \longleftarrow CBFiltering(T, M, S, t, r, u^*, s^*)$;
4 $CPR_{u^*} \longleftarrow contextualizePredictedRatings(PR, similarCtxDS, \alpha)$;
5 $recomList \longleftarrow topN(CPR_{u^*})$

transforming it to a 2D dataset (R). In line 2 a traditional 2D recommendation technique is applied on this context-free dataset. For example, in our case, we used the matrix factorization method BiasedMF [55]. The result would be a list of predicted ratings (PR). Then (line 3), we apply our correlation-based filtering module (detailed in Algorithm 1) on the initial contextual dataset T, in order to obtain a matrix of ratings given in contexts similar to the target user one, s^* ($similarCtxDS$). Now (line 4), based on this matrix we can contextualize the predicted ratings (detailed in Algorithm 6) and recommend the top N items with highest scores for user u^* (line 5).

Algorithm 6 : contextualizePredictedRatings

Input : PR: list of predicted ratings,
$similarCtxDS$: matrix of ratings given in similar contexts,
α: coefficient of the contextualizing equation.

Result : CPR: contextualized predicted ratings.

1 **foreach** $pr \in PR$ **do**
2 $\quad G \longleftarrow computeNeighbors(pr.u, similarCtxDS)$;
3 $\quad m \longleftarrow computeMeanNeighborsRatings(pr.i, G, similarCtxDS)$;
4 $\quad pr.r \longleftarrow \alpha \times pr.r + (1-\alpha) \times m$
5 **end**

Algorithm 6 illustrates the process of contextualizing the context-free predicted ratings of the first step of *CBPoF*. It has as input PR, the set of (context-free) predicted ratings, where each instance is a triplet of (user u, item i, predicted rating r), and the parameter α.
To contextualize the predicted ratings of each instance, we first (line 2) identify the neighborhood (N) of its user $pr.u$ in $similarCtxDS$: we do this by computing the similarity between the rating vectors of the user $pr.u$ and all users who have rated items in contexts similar to the context of the target user s^* (here we used the cosine similarity which is mostly used in collaborative filtering techniques to determine the similarities between users, but other similarity measures can as well be used). Then (line 3), we compute the mean of ratings done by its neighborhood for item $pr.i$ (m). Now (line4), we can contextualize each predicted rating based on a weighted sum of the context-free predicted rating ($pr.r$) and the value of m, weighted by the pre-defined coefficient α.

We detailed our pre-filtering and post-filtering context-aware recommendation approaches. In the next section, we denote that we are able to generate explanations for our context-aware recommendations.

6.3 Explanations for Context-Aware Recommendations

One of the topics that gets attention in the recommender systems field is the possibility to provide explanations about the recommendations to the user. An explanation can clarify the reasons why a specific item is proposed. As indicated by Tintarev and Masthoff in [96] proposing an explanation to the user can have multiple advantages: *transparency*, by explaining how the system works; *scrutability*, by allowing users to tell the system it is wrong; *trust*, by increasing users confidence in the system; *effectiveness*, by helping users to make good decisions; *persuasiveness*, by convincing users to try or buy an item; *efficiency*, by helping users to make decisions faster; and *satisfaction*, by increasing the ease of usability or enjoyment of an item.

In traditional RS, explanations are often content-based (e.g. "We recommend you A because you liked B"), preference-based (e.g. "Your preferences suggest that you would like A") or collaborative-based (e.g. "People who like A, also liked B") [96].
This interesting topic is not yet well exposed in the context-aware recommendation domain. To our knowledge, only few work focused on this issue ([19, 60, 105]). Nonetheless, taking into account the user contextual situation in the recommendation process not only allows to improve the quality of the recommendations, but it can also be used to explain why an item is recommended [19].

In this research direction, we propose a method to generate adequate explanations about the items that our *CBPF* and *CBPoF* approaches recommend, based on the result of our context relevance detector (described in Section 5.2):
Suppose our CARS recommends an item i to the user u^* in context $s^* : c_1, c_2, c_3, ..., c_m$ where m is the total number of context factors. In the recommendation process, we identify the most relevant context factors, based on our context relevance module (by *weighting* or *filtering* techniques). We propose to generate the explanation of the recommended item, by using either the most impacting context factor revealed by the *weighting* technique (the context factor with the highest weight), or the set of relevant context factors identified by the *filtering* technique.

Example 6.3.1. *If we return to our context-aware movie recommendation (Example 5.2.1), where we have three context factors time, social and season, the resulting weights, obtained by the weighting technique of our context relevance module, for these context factors are respectively* 0.37*,* 0.55 *and* 0.31*. So, in this case, the context factor which has the largest positive impact on the rating prediction for the movie i is the social context.*
Suppose that the value of the social context of user u^* is family. Therefore, the explanation phrase for her recommended item i would be like "It is great to watch the movie i with family".

6.4 Conclusion

In this chapter, we propose to integrate our *Correlation-Based Filtering (CBF)* approach in a pre-filtering and post-filtering configurations to produce context-aware recommendations. The former named *correlation-based pre-filtering (CBPF)* and the later *correlation-based post-filtering (CBPoF)*. For recommending an item to a user in a specific context, we do as follow:
In *CBPF*, we first filter ratings based on the context of the target user by our *Correlation-based filtering* module, and then apply a traditional RS on this selection of data.
In our post-filtering proposition, *CBPoF*, we first apply a traditional RS in the initial data while ignoring the contextual information, and then propose to contextualize the context-free predicted ratings based on a convex combination (Equation 6.1) reached from the *Correlation-Based Filtering* method.

Table 6.1 illustrates different versions of our approach and their specific attributes described in the previous chapter.
To each one of the versions of Table 6.1 we can apply a *weighting*, *filtering* or *hybrid* method in order to consider the context relevance.
The context can be represented by the correlation between ratings and contexts computed by either the PCC (Equations 5.1 and 5.2) or the mean deviation formula (Equation 5.5).
Also different similarity measures can be used in the computation of similarities among

	Approach	item-based	user-based	non-clustered	clustered	aggregation	concatenation
pre-filtering	**CBPF-IB**	x		x		x	
	CBPF-CIB-AG	x			x	x	
	CBPF-CIB-CN	x			x		x
	CBPF-UB		x	x		x	
	CBPF-CUB-AG		x		x	x	
	CBPF-CUB-CN		x		x		x
post-filtering	**CBPoF-IB**	x		x		x	
	CBPoF-CIB-AG	x			x	x	
	CBPoF-CIB-CN	x			x		x
	CBPoF-UB		x	x		x	
	CBPoF-CUB-AG		x		x	x	
	CBPoF-CUB-CN		x		x		x

Table 6.1: Different versions of our CBPF and CBPoF approaches

contexts. We detailed the cosine and euclidean similarities respectively in Equations 5.3 and 5.4.

We also propose a method to generate explanations for our context-aware recommendations, based on the results of our context relevance detector.

In the next chapter, we will evaluate and compare these different versions of *CBPF* and *CBPoF*, discuss about the properties of each one and compare with baselines and state of the art approaches.

Chapter 7

Experimental Analysis

In the two previous chapters, we first present a filtering approach to transform a multi-dimensional contextual dataset to a 2D dataset named *Correlation-Based Filtering* approach. Then we propose two context-aware recommender systems which integrate this filtering module in their recommendation process, in order to take into account the user context and generate context-aware recommendations: *Correlation-Based Pre-Filtering (CBPF)* and *Correlation-based Post-Filtering (CBPoF)*.
This chapter evaluates the features and performances of our proposed context-aware recommendation approaches.

7.1 Datasets and Parameters

In this section, we report about our experimental analysis. We first describe the four datasets that we used, we then report about parameters used in our approach, and the metrics for experimental evaluation.

7.1.1 Datasets

We evaluated our approach on four real world datasets, which are well-known among the CARS community: (a) *CoMoDa*, a contextual dataset for movie recommendation, collected from surveys [59]. In this dataset, context situations are defined by 12 different context factors: *time, day type, season, location, weather, social context, end emotion, dominant emotion, mood, physical context, decision* and *interaction*; (b) *STS* [33], a tourism dataset, containing contextual ratings for places of interest, collected using a mobile tourist application. In this dataset, context situations are expressed using 14 context factors: *distance, available time, temperature, crowdedness, knowledge of surroundings, season, budget, day time, weather, companion, mood, weekday, travel goal* and *means of transport*; (c) the *Music* dataset, containing ratings for contextual music recommendation, collected by an in-car music recommender developed by

Baltrunas et al. [18]. In this dataset we have a total number of 8 context factors (*driving style, landscape, mood, natural phenomena, road type, sleepiness, traffic conditions* and *weather*), but it has this specificity that for each context situation, the value of only one context factor is known; and (d) the *Trip* dataset [112], a larger tourism dataset scripted from online reviews on tripadvisor.com, with the unique context factor *trip type*.[1]
We have to note that there are only few contextually tagged rating datasets available, and these four are amongst the more popular ones used in the literature.

7.1.1.1 Mapping to our generic context factor categorization

In Table 7.1, we have mapped the context factors of these four datasets to our proposed context categories presented in Chapter 4. We can see that for these datasets, we can find contextual information about mostly all categories except the categories of *technical context (hardware and data)*, *equipment* and *demographic context*. The two first ones can be useful in more specific applications, like recipe recommendation where *equipment* would be an important context factor, or software recommendation, where *technical context* would matter. The last one, *demographic* context has a particularity. Indeed, the *demographic* category is not an entire dynamic category like other context categories, but a semi-dynamic category. So depending on the application, information of this category like age, genre, etc, could be taken into account either as contextual information of the user, or as static characteristics of her. The former will be the case of applications where the time spent between two consecutive recommendations is very long, like recommendations done for buying a car or a house. In these cases attributes like the age of the user will count like contextual information, because it is likely that from one purchase to another their values change. But in applications like music or movie recommendations, we will not have enormous changes of these demographic information along recommendations. So we will treat these kind of information like static characteristics of the user. This is the case for our application examples of movie, travel and music recommendation, and this explain why this context category is empty.

7.1.1.2 Items/users characteristics information

In these datasets beside the contextual information about the users, we also have some (static) characteristics about items/users which will essentially be used in the clustering phase. In *CoMoDa* we have movies' characteristics like the genre, year, language, country, director, actors and budget. But also the age, gender, city and country of the users. In *STS* we have the point of interest category and some static information about the users like the birth-date, gender, openness to experience, conscientiousness, extroversion, agreeableness and emotional stability.

[1]The three last context-aware datasets are available on the repository of the CARSKit application: https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets

Context Factors		CoMoDa	STS	Music	Trip
Physical	**Temporal**	time, daytype, season.	available time, season, daytime, weekday.	-	-
	Spatial	location	crowdedness	landscape, road type, traffic condition.	-
	Environmental	weather	weather	weather	-
	Equipment	-	-	-	-
Personal	**Demographic**	-	-	-	-
	Social	social	companion	-	-
	Psychophysiological	end emotion, dominant emotion, mood, physical.	mood	mood, sleepiness.	-
	Cognitive	decision, interaction.	knowledge of surroundings, budget, travel goal.	driving style	trip type
Technical	**Hardware**	-	-	-	-
	Data	-	-	-	-

Table 7.1: Correlation between the contextual information in datasets and our context categorization (Chapter 4)

Characteristics	**CoMoDa**	**STS**	**Music**	**Trip**
#ratings	2296	2534	4012	14063
#users	121	325	42	2371
#items	1197	249	139	2269
rating scale	1-5	1-5	1-5	1-5
rating's mean	3.83	3.47	2.37	4.13
rating's median	4	4	2	4
rating's standard deviation	1.05	1.29	1.48	0.93
non-contextual sparsity	98.41%	96.86%	31.27%	99.73%
contextual sparsity	99.99%	99.99%	99.99%	99.94%
#context factors	12	14	8	1
#context conditions	49	59	26	5
#items characteristics	7	1	5	3
#users characteristics	2	7	0	2

Table 7.2: Datasets' descriptive statistics

For the *Music* dataset we have some static information about the songs like the title, the artist name and the music category. And finally in the *Trip* dataset we have the user state and time zone, but also the city, state and time zone of hotels.

7.1.1.3 Descriptive statistics

Table 7.2 illustrates some descriptive statistics about these datasets. Note that we calculated the non-contextual and contextual sparsities respectively by means of the following equations (Equation 7.1 and 7.2):

$$1 - \frac{\#ratings}{\#users \times \#items} \tag{7.1}$$

$$1 - \frac{\#ratings}{\#users \times \#items \times \prod_{i=0}^{m} |CC_{f_i}|} \tag{7.2}$$

In the latter, we consider the dimensions of contextual information: m is the total number of context factors, f_i refers to the i-th context factor, and $|CC_{f_i}|$ refers to the number of context conditions of the context factor f_i. Due to the large number of contextual information, this sparsity is extremely high (at least 99.94%)

As Table 7.2 shows, contrary to the *Music* dataset, the three other datasets are very sparse (while ignoring the context). In addition, the *Music* and *STS* datasets have the disadvantage of a lack of fully context situation information. The ratings of the four datasets go from 1 to 5.

But the distribution of the ratings are not similar: in *CoMoDa*, *STS* and *Trip*, the items are mostly well rated, with a mean of around 3.5-4 and a median of 4. But in the *Music* dataset, the rating distribution is more important in the middle and lower part. In fact we have a median of 2 and a mean of 2.37.

7.1.1.4 Context conditions frequencies

The number of occurrences of context conditions of each factor is not always balanced. Figures 7.1, 7.2, 7.3 and 7.4 illustrate their distribution for the four datasets. In these plots, each bar corresponds to a context factor, and each colored partition in a bar illustrates the percentage of a possible value.

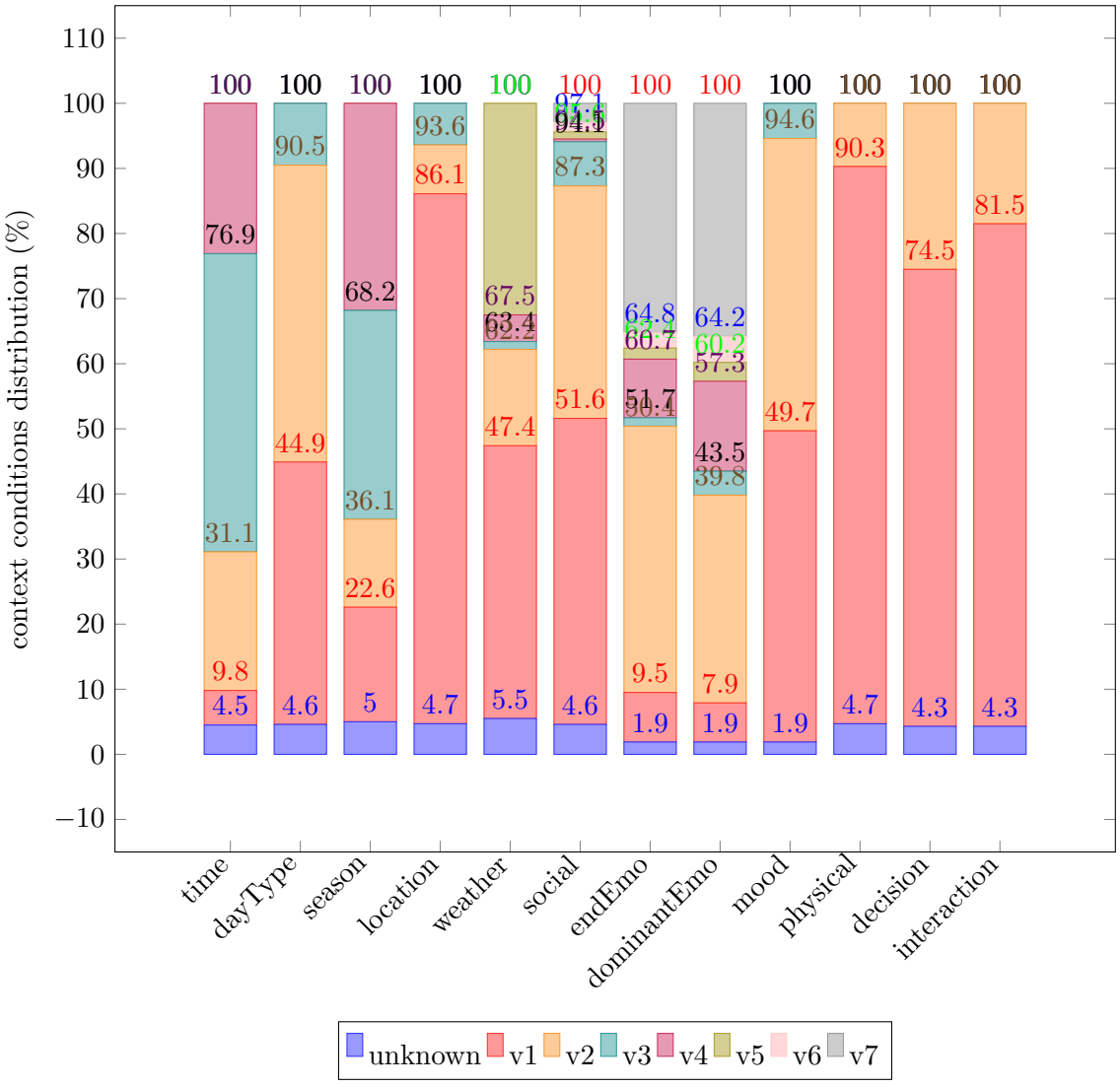


Figure 7.1: Context conditions distribution of *CoMoDa* context factors

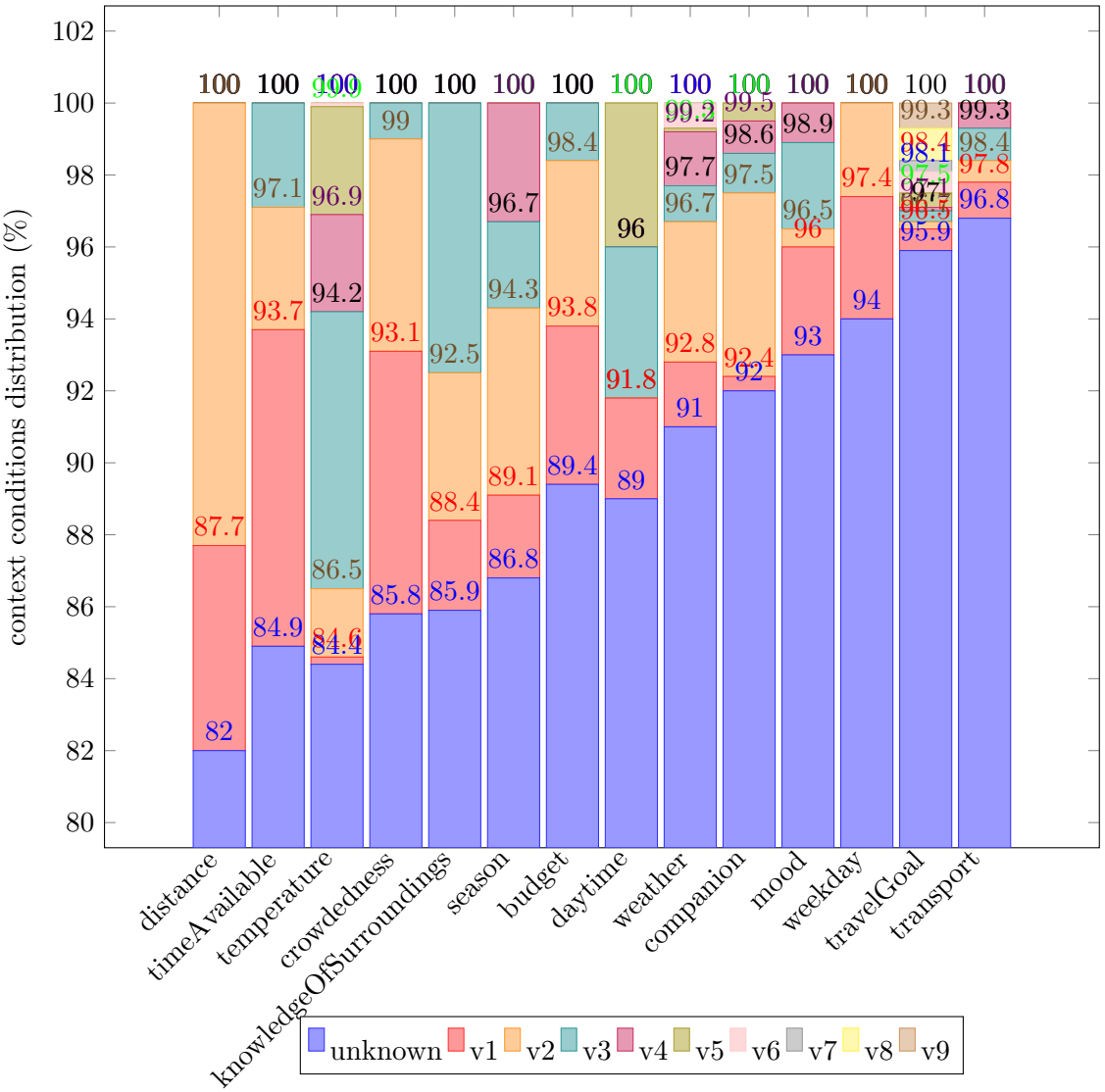


Figure 7.2: Context conditions distribution of *STS* context factors

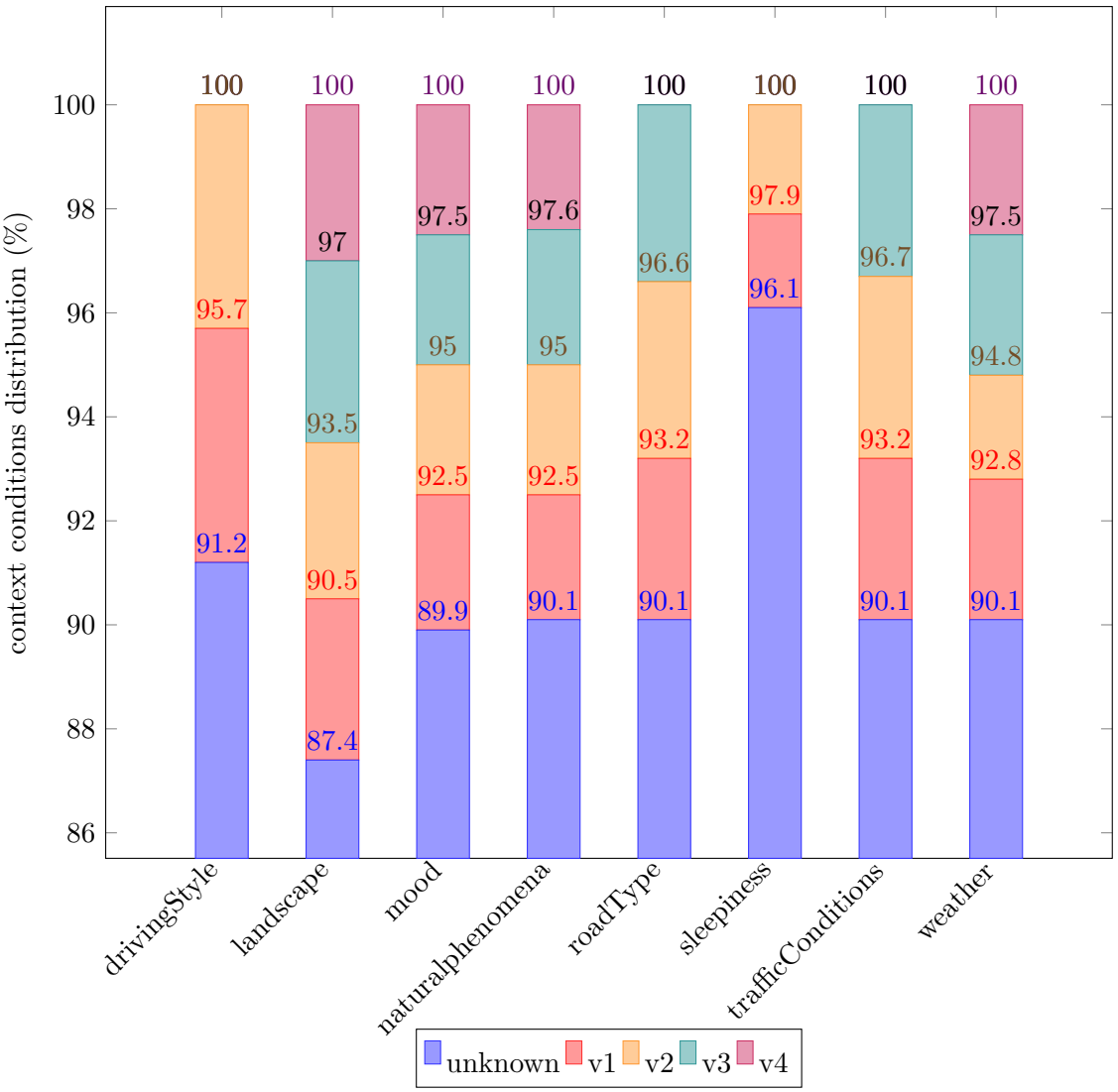


Figure 7.3: Context conditions distribution of *Music* context factors

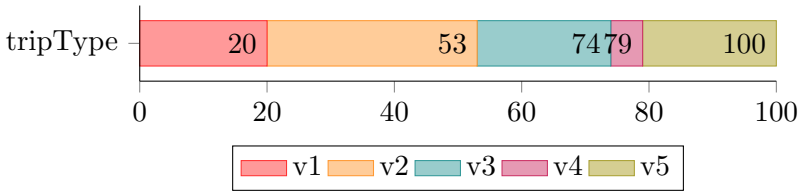


Figure 7.4: Context conditions distribution of *Trip* context factor

The first blue color in a bar of these plots corresponds to the percentage of not specified value by the user (unknown), and the others correspond to the possible context conditions. For example in Figure 7.1, the first bar (in left) illustrates the distribution of possible values of the context factor *time*, which are from bottom to top : unknown (4.5%), morning (5.3%), afternoon (21.3%), evening (45.8%) and night (23.1%). In this case we can see that almost the half of observations occurred in the *evening* (45.8%), while only 5.3% occurred in *morning*. We

can observe this kind of more or less imbalance feature for all of the context factors of *CoMoDa*, as well as for the unique context factor of the *Trip* dataset (Figure 7.4). The case of *STS* and *Music* is roughly different: these two datasets are not as well specified as *CoMoDa* in term of contextual information. As we can see in Figures 7.2 and 7.3, for each context factor, more than 80% of the observations have lack of contextual information. For the rest, similarly to *CoMoDa* and *Trip*, we can see an imbalance of context conditions in *STS*, while in *Music*, the proportion of context condition frequencies is balanced.

7.1.2 Modeling Parameters

In this section, we will explain some modeling parameters such as the clustering process, the context similarity threshold and the context-free algorithm used in our experimentations, in order to ensure the reproducibility.

The item/user clustering in the first step needs some pre-treatments:
Firstly we have put aside non-characteristic parameters of items/users, like actors and directors in the *CoMoDa* dataset, and artist in the *Music* dataset. By non-characteristic, we mean that they will not be of help for clustering, as each one has a huge number of possible values in comparison to the total number of items/users. Moreover some characteristics have redundant information, like the hotel city, state and time zone in the *Trip* dataset. In this case we can keep only one of these characteristics, like the state.
Generally, items' or users' characteristics are a mixture of numerical and nominal variables. We have made these uniform by transforming numerical variables such as year and budget of movies in *CoMoDa*, and age of users in *CoMoDa* and *STS* datasets (note that we transform birthday to age in the *STS* dataset, for an easier treatment). By an analysis of the data distribution, we have created two classes for the *year* variable: *ancient movies*, those realized before 1988, and *recent movies*, realized after this date. For the *budget* variable, we have made tree segmentations of *weak budget* (less than 18,000,000 $), *moderate budget* (between 18,000,000 and 50,000,000 $) and *large budget* (more than 50,000,000 $). And for *age*, we grouped by interval of 5 years.
In some cases where we do not have equally distributed values of variables, a grouping is needed: in *CoMoDa*, for movie language, we have 28 different languages, but 88.61% of movies are in English. So we have replaced the values of languages other than English by a new value *"other"*, and we have done a similar treatment for the movie country variable; in *STS*, the Point Of Interest (POI) category is defined by a number from 1 to 29. We have kept the POI categories 1, 3, 4 and 9, and we have grouped all others in a single cluster, because the frequency of each one was less than 5% of the total.

After these pre-treatments, we can cluster items or users. Depending on the available information about items' or users' characteristics, two strategies exist for the clustering:

- in case of more than one available characteristic, we can apply a standard clustering

algorithm like Hierarchical Clustering (HC) based on these characteristics. We choose the HC technique, which contrary to *k-means* does not require a pre-defined number of clusters. HC uses a bottom-up approach, it starts to place each item in a cluster, and iteratively merges the two closest clusters (represented in a dendrogram), until all the items are merged into a single cluster. By interpreting the dendogram, we can stop at whatever number of clusters we find appropriate.

- otherwise, if we have only one characteristic (e.g. POI category in *STS* or music category in *Music* datasets), we can directly use this variable as cluster identifier.

So we applied HC on the items and users of *CoMoDa*. As result, the best segmentation proposed was 4 items' clusters and 5 users' clusters. In the case of *STS*, we obtain 2 clusters of users by HC. But for clustering items, as we had only one characteristic about them, which is the POI category, we used it directly as cluster number. For *Music*, where we had 5 characteristics about items: music category, title, artist, mp3 and image url. We used the music category, which is between 1 and 10, as items cluster number. And finally for *Trip* we use the states of items and users as items and users cluster id.
In step 3 of our *correlation-based filtering* approach (Figure 5.1), we need to set a similarity threshold for identifying the most similar contexts. To identify the best threshold for each dataset, we evaluate the approach on the dataset by different threshold from 0.1 to 0.9, and select the one that yielded better prediction accuracy (minimum MAE). So, we set the similarity threshold equal to 0.6 for *CoMoDa*, 0.3 for *STS*, and 0.5 for *Music* and *Trip*. A similarity threshold equal to 0.5 means that when we want to select local datasets, we select ratings that have been given in context situations which are more than 50% similar to the target user context situation.

The traditional (context-free) recommendation technique used in the last step of our approach is the Biased Matrix Factorization model [57] (from LibRec Java API [46]), which is one of the best-performing techniques reported in the state of the art [40].

7.1.3 Evaluation Parameters

For the evaluation of our approach, we avoided to exclude items or users with low counts, in order to match as closely as possible the conditions of real recommendation applications. Due to the relatively small size of our datasets, we evaluated our approach based on 5-fold cross-validation. As many research in the domain, we used Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics to evaluate the rating estimation. As illustrated in Equations 7.3 and 7.4, where n is the total number of ratings, these metrics compute the difference between the actual (r_i) and predicted ratings ($\hat{r_i}$), but the RMSE penalizes large errors more. Lower values of these metrics show better performances.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - \hat{r}_i| \tag{7.3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2} \tag{7.4}$$

Note that due to the small size of the available contextual datasets, we obtain very low values for recommendation performance metrics such as Precision and Recall, which are not enough representative of the CARS performances. This is the reason why we have abandoned these metrics, and based our evaluations on the rating estimations metrics.

7.2 Results and Discussion

We implemented our approach by Java (jdk 1.8) in the Eclipse IDE, and evaluated our approach in seven steps: (1) we compared the performances of the derived versions of our approach in term of context representation in the pre-filtering configuration, (2) we studied the sparsity reduction obtained by our pre-filtering method, (3) we evaluated the effect of context relevance consideration in the recommendation performances, (4) we studied the difference of performance by different similarity measures, (5) we compared the performances of pre- and post-filtering adaptation of our approach, (6) we compared our context-aware recommendation approaches with a context-free recommendation approach and two baselines, and finally, (7) we compared our approaches with well-known state of the art CARS approaches.

7.2.1 Comparing Different Context Representations

Tables 7.3 and 7.4 illustrate the performances of the derived versions of our *CBPF* approach, in terms of rating estimation, respectively by the *PCC* and the *mean deviation* (see Section 5.1.1) methods. *CBPF-IB* and *CBPF-UB* refer to the item- and user-based correlation model,

	CoMoDa		**STS**		**Music**		**Trip**	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF-IB	0.84	1.05	0.95	1.19	1.25	1.51	0.82	1.06
CBPF-CIB-AG	0.84	1.05	0.87	1.10	1.25	1.50	**0.81**	**1.05**
CBPF-CIB-CN	0.84	1.05	0.97	1.19	**1.21**	**1.44**	**0.81**	**1.05**
CBPF-UB	0.85	1.05	0.95	1.18	1.27	1.50	0.82	1.06
CBPF-CUB-AG	0.84	1.05	**0.86**	**1.09**	—	—	**0.81**	**1.05**
CBPF-CUB-CN	**0.81**	**1.02**	0.98	1.22	—	—	**0.81**	**1.05**

Table 7.3: MAE/RMSE of the derived techniques of our CBPF approach (with *PCC*)

	CoMoDa		STS		Music		Trip	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF-IB	0.84	1.05	0.95	1.18	1.23	1.48	0.81	1.05
CBPF-CIB-AG	0.84	1.05	0.87	1.10	1.23	1.48	0.82	1.06
CBPF-CIB-CN	0.85	1.06	1.02	1.25	**1.20**	**1.43**	0.82	1.06
CBPF-UB	0.83	1.05	0.96	1.19	1.23	1.48	0.81	1.05
CBPF-CUB-AG	0.84	1.05	**0.86**	**1.09**	—	—	**0.81**	**1.04**
CBPF-CUB-CN	**0.77**	**0.99**	0.99	1.23	—	—	**0.81**	**1.04**

Table 7.4: MAE/RMSE of the derived techniques of our CBPF approach (with *mean deviation*)

CBPF-CIB-AG and *CBPF-CUB-AG* refer to the correlation models based on the cluster of items or users, with the aggregation technique, and finally *CBPF-CIB-CN* and *CBPF-CUB-CN* refer to the same model, but with the concatenation technique (Please refer to the *Extension 2.1* of the Section 6.1.1).

As we can see in both tables (7.3 and 7.4), there is not a single winner between the different context representation models. Contrary to the *STS* dataset where we obtain better results by the aggregation technique, for the datasets *CoMoDa* and *Music* the concatenation technique perform better. The case of the *Trip* dataset is a bit different: in this dataset the user context is expressed by only one context factor. So, the context representation by the aggregation and concatenation models will be exactly the same, and this is why the results of these two families of models are identical for this dataset.
However in any case we can note that by clustering items/users we not only gain in term of computation cost but also in term of rating estimation performance. This confirms the fact that the more variables are rich, better we can catch the correlation between them. Another interesting point is that it seams the user-based influence model gives comparable or slightly better results (CBPF-CUB-CN for *CoMoDa*, CBPF-CUB-AG for *STS* and CBPF-CUB-AG/CN for *Trip*). It could show that the influence of contexts on ratings is more user-based than item-based (but have to be tested on more datasets). Note that we could not do this comparison on the *Music* dataset, because we did not have information about users' characteristics for user clustering.
By comparing the performances of the models based on *PCC* (Table 7.3) and the ones based on *mean deviation* (Table 7.4), we can see that in most cases the *mean deviation* performances are comparable or better than *PCC*. This can be explained by the relatively small size of the datasets, and the fact that the PCC is impacted by the number of available data. Indeed, by PCC it is harder to catch the real correlation in small datasets. Because in *PCC* we have a product function (see Equation 5.1) which represent the interaction of ratings and context

condition values. So the more data we have, the best we can model this interaction, and the PCC value will be more significant. Whereas the *meanDeviation* method compute a more simple correlation between ratings and contexts, only based on the mean value of ratings done when a certain context condition is present and when it is not.
The winner models for each dataset is illustrated in bold: while for *STS*, *Music* and *Trip* we have comparable results, for *CoMoDa* we have an improvement by the *mean deviation* (MAE decrease from 0.81 to 0.77).

We can conclude that the choice of the best model depends strongly to the application and data. But in any cases the clustering strategy have to be taken in order to not only reduce computational cost, but also catch more precise correlation which results in more relevant context representations.

7.2.2 Sparsity Reduction

As stated earlier, one of the main challenges of context-aware recommender systems is data sparsity: producing relevant recommendations, while limited amount of user's ratings are available to train the recommendation model.
Our pre-filtering approach, *CBPF*, tries to decrease the sparsity (or augment the density) level of the dataset which will be used as the input of the recommender system, by transforming the initial multi-dimensional dataset to a 2-dimensional dataset. This is done by filtering the initial data and selecting only ratings given in contexts similar to the target user one (by our *correlation-based filtering* module).

Table 7.5 displays the density of data for each dataset, before (first line) and after (second line) doing the correlation-based pre-filtering. The former density is computed by the Equations 7.5 and the later is done by Equation 7.6 (where m is the total number of context factors, and $|CC_{f_i}|$ refers to the number of possible context conditions for the context factor f_i).

$$\frac{\#ratings}{\#users \times \#items \times \prod_{i=0}^{m} |CC_{f_i}|} \tag{7.5}$$

$$\frac{\#ratings_{similarContexts}}{\#users \times \#items} \tag{7.6}$$

The results of Table 7.5 demonstrate that in most cased (expected for *Trip*), the density has been substantially augmented by *CBPF*: it has increased the data density by 39 times for *CoMoDa*, 93 times for *STS* or even 208 times for *Music*. The only dataset for which the pre-filtering does not change the sparsity level is the *Trip* dataset, because its contextual information is already limited by only one context factor, and five context conditions.

Density	**CoMoDa**	**STS**	**Music**	**Trip**
contextual density	0.01%	0.01%	0.01%	0.06%
pre-filtered density	0.39%	0.93%	2.08%	0.06%

Table 7.5: Density of the multi-dimensional contextual dataset vs. pre-filtered 2D dataset

7.2.3 Effect of Context Relevance Consideration

Versions / Models	**CoMoDa** *(CBPF-CUB-CN)*		**STS** *(CBPF-CUB-AG)*		**Music** *(CBPF-CIB-CN)*	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF without context relevance	0.81	1.02	0.86	1.09	1.21	1.44
CBPF Weighting (a)	0.82	1.03	0.87	1.11	1.20	1.44
CBPF Weighting (b)	0.82	1.03	0.87	1.11	1.20	1.43
CBPF Filtering (a)	0.81	1.02	0.86	1.09	1.20	1.44
CBPF Filtering (b)	0.79	0.99	0.81	1.03	1.20	1.43
CBPF Hybrid (a)	0.82	1.03	0.84	1.07	1.20	1.44
CBPF Hybrid (b)	**0.75**	**0.95**	**0.8**	**1.03**	**1.19**	**1.42**

Table 7.6: Best MAE/RMSE performances of the CBPF versions with context relevance consideration

Table 7.6 shows the best performance among the different versions of CBPF, where we consider the context relevance in the recommendation process (see Section 5.2). The *Trip* dataset is the absent of this analysis section, because of its single context factor.
For each dataset we test the different propositions of integrating the context relevance, on the best performing CBPF model based on the results reported in Table 7.3. So, for *CoMoDa*, we test on *CBPF-CUB-CN*, for *STS* we test on *CBPF-CUB-AG* and for *Music* we test on *CBPF-CIB-CN* approach. The weight threshold used in methods *filtering (a)* and *hybrid (a)* is set empirically to the maximum context factor weight divided by 2.
The results show that taking into account the context relevance can effectively improve the performances of the CARS. Between the different proposed methods, the last one which is the hybrid method (b) give us the best performances. In this method we identify the relevant context factors based on an offline causal inference test and integrate a weighted vector representation of them into the context representation, while ignoring the irrelevant context factors.

From this, we can conclude that not all contextual information is useful to take into account in the recommendation process. Some of them would be noise, and it is not only crucial to detect the relevant contextual information, but also treat each one based on their impact degree for a better recommendation.

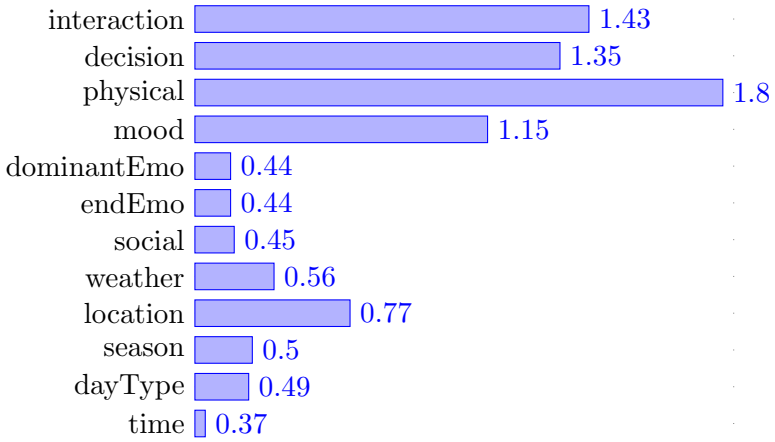
Let us take a closer look at the context factors weights and their relevance identified by the different methods proposed in section 5.2. Figures 7.5a, 7.5b and 7.5c illustrate the weights computed by the *weighting (b)* method (Equation 5.8) for the three datasets *CoMoDa*, *STS* and *Music*.

Based on these weight values, in the *filtering (a)* method, we set the weight threshold ($weight_{max}/2$) for each dataset: for the *CoMoDa* dataset we obtain a threshold equal to 0.90, for *STS*: 2.43 and for *Music*: 0.57. Then the context factors with a weight greater than its corresponding threshold are marked as relevant context factors.

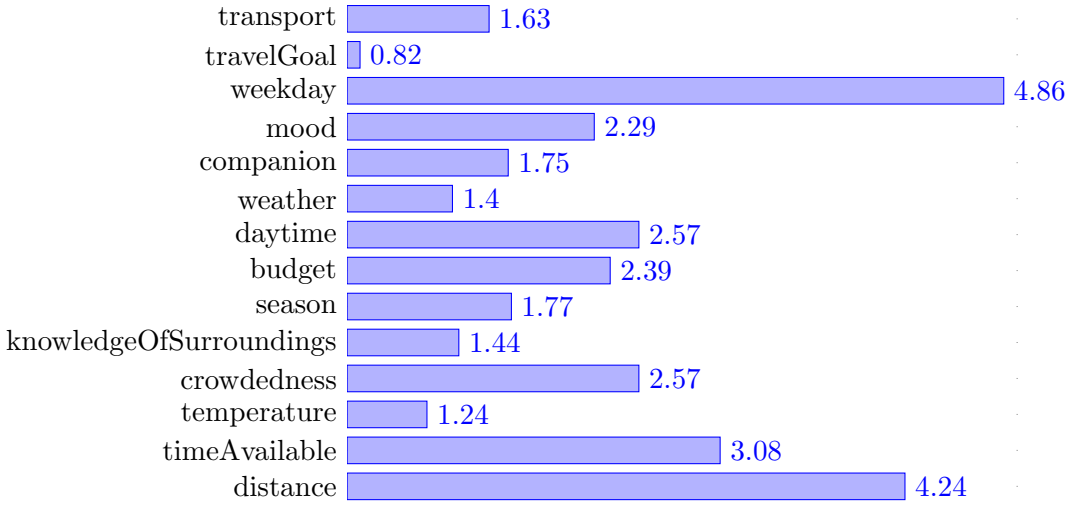
Note that we test multiple threshold formulas ($weight_{max} - weight_{min}/2$, $weight_{min}/2$, mean of weights, etc), and choose the best performing one which is the half of the maximum weight ($weight_{max}/2$).

Tables 7.7a, 7.7b and 7.7c show the relevant context factors identified by each *filtering* methods *a* or *b*. The green mark (✓) means the context factor is identified as relevant and the red one (✗) means it is identified as irrelevant.

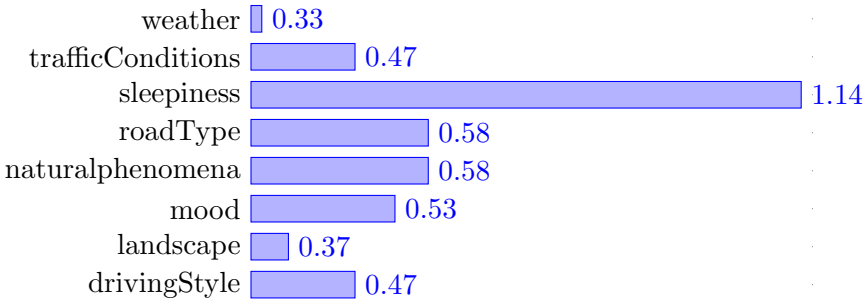
Contrary to the case of *STS* and *Music* where we have some common context factors detected as relevant by the two methods, in the case of *CoMoDa* there is no intersection between the two sets of identified relevant context factors.



(a) *CoMoDa*



(b) *STS*



(c) *Music*

Figure 7.5: Context factors weights

Table 7.7: Context relevance detection results by the methods *filtering(a)* and *filtering(b)*

Context factors	filtering(a)	filtering(b)
time	✗	✗
daytype	✗	✗
season	✗	✗
location	✗	✗
weather	✗	✗
social	✗	✗
endEmo	✗	✓
dominantEmo	✗	✓
mood	✓	✗
physical	✓	✗
decision	✓	✗
interaction	✓	✗

(a) *CoMoDA*

Context factors	filtering (a)	filtering (b)
distance	✓	✗
timeAvailable	✓	✗
temperature	✗	✗
crowdedness	✓	✓
knowledgeOfSurroundings	✗	✗
season	✗	✗
budget	✗	✗
daytime	✓	✓
weather	✗	✗
companion	✗	✗
mood	✗	✗
weekday	✓	✓
travelGoal	✗	✗
transport	✗	✗

(b) *STS*

Context factors	filtering(a)	filtering(b)
drivingStyle	✗	✗
landscape	✗	✗
mood	✗	✗
naturalphenomena	✓	✓
roadType	✓	✗
sleepiness	✓	✓
trafficConditions	✗	✓
weather	✗	✗

(c) *Music*

7.2.4 Euclidean vs. Cosine Similarity

	CoMoDa		**STS**		**Music**		**Trip**	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Cosine	**0.75**	**0.95**	**0.8**	**1.03**	1.19	1.42	**0.81**	**1.05**
Euclidean	0.83	1.04	0.93	1.19	**1.0**	**1.26**	**0.81**	**1.05**

Table 7.8: Euclidean vs. Cosine similarity performances

Table 7.8 illustrates the comparison between the resulting performances of similarity measures *cosine* and *euclidean*. Here the best results for each dataset is reported (see Table 7.6). Contrary to *Music* where the *euclidean* similarity outperforms the *cosine*, for *CoMoDa* and *STS* we have considerably better results with the *cosine* similarity.
To explain this difference, we have to take a closer look to the datasets: two main points differentiate the *Music* dataset from the two other datasets: (1) comparing to *CoMoDa* and *STS*, in the *Music* dataset, we have more available rating data (around 4000 ratings for *Music*, comparing to around or less than 2500 ratings for *STS* and *CoMoDa*), (2) but less available contextual information: only 8 context factors (comparing to 12 or 14) and the particularity that the value of only one context factor is known for each observation. This two points make potentially more scaled context vectors, which are not well treated by the cosine similarity measure. For the *Trip* dataset, results show that the similarity measure does not affect the recommendation performances.

7.2.5 Pre-Filtering vs. Post-Filtering

	CoMoDa		**STS**		**Music**		**Trip**	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF	**0.75**	**0.95**	0.80	1.03	1.0	1.26	**0.81**	**1.05**
CBPoF	0.81	1.01	**0.59**	**0.76**	**0.88**	**1.09**	0.87	1.12

Table 7.9: Recommendation performance of CBPF vs. CBPoF

Table 7.9 shows a comparison of the best results of *CBPF* (Correlation-Based Pre-Filtering) vs. *CBPoF* (Correlation-Based Post-Filtering). In the second line the results correspond to the following configurations: the *CBPoF-CUB-CN* for *CoMoDa*, with α equal to 0.7, the *CBPoF-CUB-AG* for *STS*, with α equal to 0.5, the *CBPoF-CIB-CNT* for *Music* with α equal to 0.9, and the *CBPoF-CUB-CNT* with α equal to 0.5 for *Trip* (we will discuss about the value of α bellow in the next paragraph). The results show that in some cases the pre-filtering

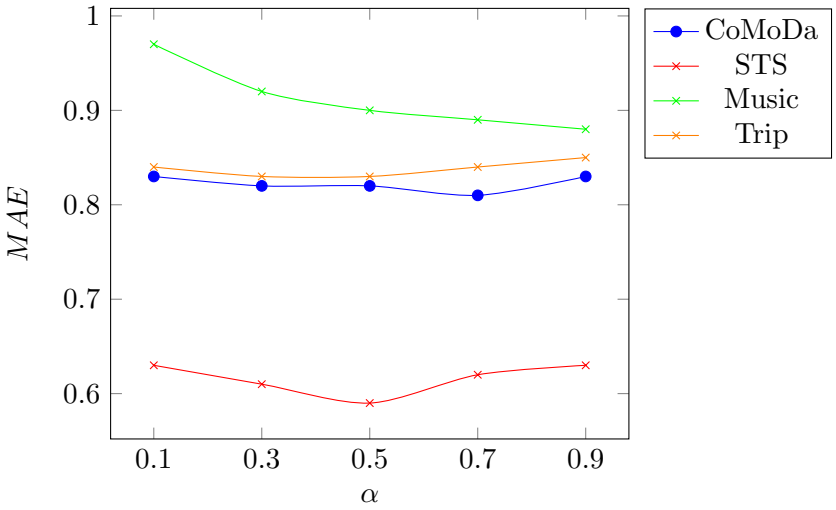


Figure 7.6: Effect of α on the MAE performance of *CBPoF*

adaptation of our approach dominates the post-filtering (*CoMoDa* and *Trip*) and in some other cases (*STS* and *Music*) the post-filtering outperforms. Contrary to *STS* and *Music* where there is a lack of full declared contextual information, the datasets *CoMoDa* and *Trip* contains more complete contextual data for each observation. Indeed as the Figures 7.2 and 7.3 show, for each context factors of *STS* and *Music*, the value of more than 80% of the observations are not reported and unknown. This fact could explain the reason why in cases like *CoMoDa* and *Trip* where the contextual information is well-reported for each observation, the pre-filtering approach performs better.

Figure 7.6 shows the evolution of MAE for *CBPoF* depending on the value of α. As said in the previous chapter, the value of α (which could be between 0 and 1) is set to determine the distribution of the impact degree between the personalization and the contextualization part of the context-aware recommendation task. A higher value will give more importance to the personalization, while a lower value makes the context impact more important.
In the case of our four datasets: for *Music* we obtain better results, which can even beat the pre-filtering performances, by a very high α (0.9). This means that the context is not impacting a lot the results, and can be explained by the nature of the dataset, where there is a poor contextual information for each observation. For *STS* and *Trip*, we have an equally distributed impact of the two aspects (personalization and contextualization). For *CoMoDa* an α equal to 0.7 give a higher importance to the personalization, while keeping still the impact of the context in its recommendations.

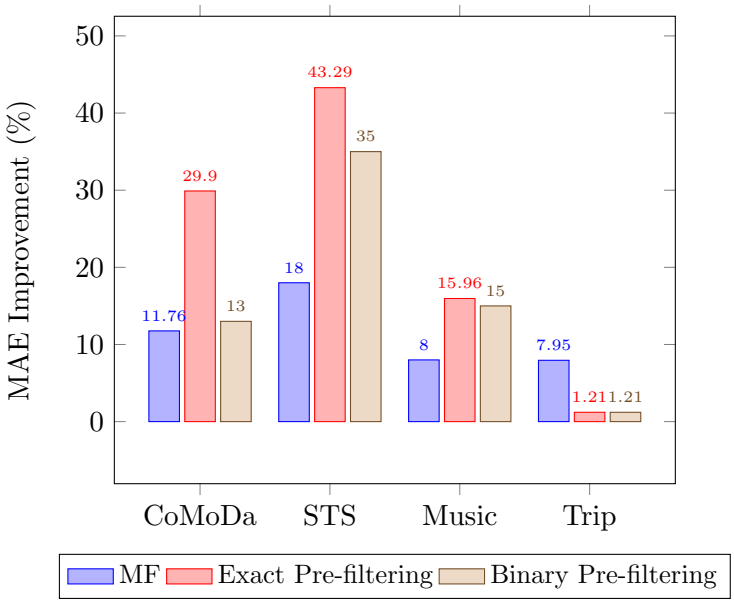


Figure 7.7: MAE improvement (%) of CBPF with respect to context-free MF and baselines

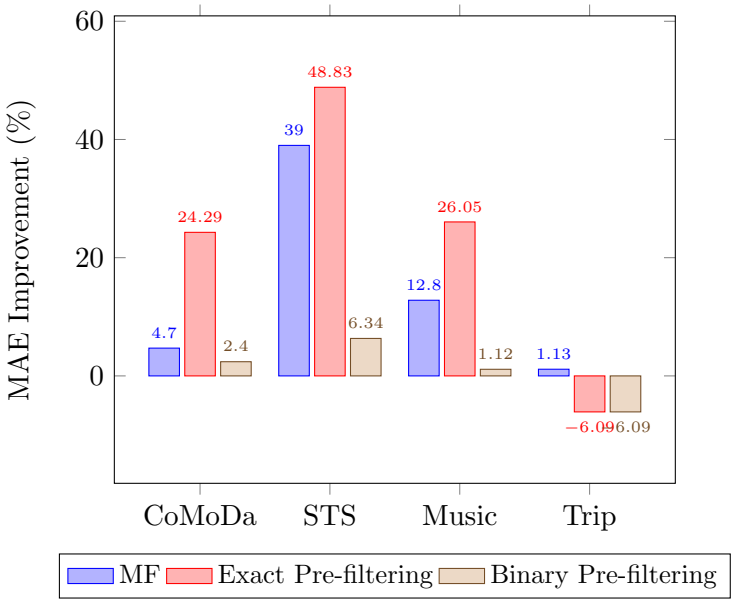


Figure 7.8: MAE improvement (%) of CBPoF with respect to context-free MF and baselines

7.2.6 Comparing CBPF and CBPoF with Baseline Methods

Figures 7.7 and 7.8 illustrate the MAE improvement that our correlation-based approaches, respectively *CBPF* and *CBPoF*, make over the context-free recommendation and the baselines. The performance improvements are expressed in percentage and computed as follows in Equation 7.7:

$$Improvement_{baseline} = (\frac{MAE_{baseline} - MAE_{CBFP/CBPoF}}{MAE_{baseline}}) \times 100 \tag{7.7}$$

The context-free recommendation technique used in this experimentation is a Matrix Factorization (MF) technique named BiasedMF, proposed by Koren in [55] (technique described in section 3.3.1). The comparison of our context-aware recommendation and this context-free matrix factorization confirms that users' contextual information can help the recommender to improve its performance.

The first baseline we used is the *exact pre-filtering* approach proposed by Adomavicius et al. in [4], and the second one is a *binary pre-filtering* method which uses a binary representation of the context in a pre-filtering configuration. In this method, we represented the context by means of a binary vector with a size equal to the total number of context conditions, where the value of each cell is equal to 1 if the corresponding context condition is present in the context situation, or equal to 0 if it is not present. We did a pre-filtering recommendation using this binary context representation.

As Figures 7.7 and 7.8 show, except for the post-filtering of *Trip* (discussed in the next paragraph), the two versions of our approach outperform these baselines. The improvements over the *exact pre-filtering* show that the idea of filtering the ratings based on the ones done in similar contexts is effective. And the improvements over the *binary pre-filtering* show the positive effect of representing the context based on the influence of context on ratings.

In the case of *Trip*, we can see that the post-filtering can not beat the pre-filtering approaches, even the simple ones (exact pre-filtering and binary pre-filtering). The explanation that we can bring for this observation is that a single context factor can not efficiency conduct the contextualization of the results of a context-free recommender system in a post-filtering configuration. So, in cases where we have very limited contextual information, pre-filtering approaches have to be adopted.

7.2.7 Comparing CBPF and CBPoF with the State of the Art

Finally, we compared our approach with four well-known state of the art approaches, which report interesting performances in the literature and in some cases their models are more closer to our approach: (a) *DSPF (Distributional Semantic Pre-Filtering)* [40], (b) *DCM (Differential Context Modeling)* [109], (c) *Splitting* approaches [110] and (d) *Deviation-based*

	CoMoDa		STS		Music		Trip	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF	**0.75**	**0.95**	0.80	1.03	1.0	1.26	**0.81**	**1.05**
CBPoF	0.81	1.01	**0.59**	**0.76**	0.88	1.09	0.87	1.12
DSPF	0.86	1.08	1.26	1.62	1.76	2.49	0.84	1.06
DCM	0.79	1.04	0.96	1.24	1.11	1.41	0.87	1.14
Splitting	0.82	1.03	0.82	1.18	**0.65**	**1.0**	0.88	1.13
CAMF	0.76	1.02	1.03	1.37	0.82	1.06	0.87	1.12

Table 7.10: Comparison with state of the art

CAMF (Context-Aware Matrix Factorization) [21]. In fact *DSPF* and *CAMF* approaches try to model the context based on the influence of contexts on ratings, and *DCM* and *DSPF* uses the similarities among contexts in their approaches. Each one of these approaches have different versions (cited in Chapter 2). We tested all the possible versions, and the performances of the best version of each approach, in terms of rating estimation are illustrated in Table 7.10 for each dataset.
We have to note that we exclude ranking-based approaches (such as *similarity-based CAMF* or *CSLIM* approaches) from our comparison, because as explained before, we based our evaluation on rating estimation metrics. Indeed, we can categorize the CARS algorithm to rating-based and ranking-based algorithms. Contrary to rating-based ones, which first estimate the missing ratings of the dataset and then recommend items with highest rates, the ranking-based CARS algorithms, generate a ranking of items and recommend the best ranked items. So MAE/RMSE could not be evaluated for this kind of algorithms.

We tested state of the art algorithms by relying on the CARSkit Java API [114]. So for each dataset, we report the performances of the followings versions:

- *CoMoDa*: the hybrid (b) version of CBPF-CUB-CN, CBPoF-CUB-CN, DSPF-IB, DCW, UISplitting and CAMF-CU,
- *STS*: the hybrid (b) version of CBPF-CUB-AG, CBPoF-CUB-AG, DSPF-IB, DCW, Item-Splitting and CAMF-CU,
- *Music*: the hybrid (b) version of CBPF-CIB-CN (with euclidean similarity), CBPoF-CIB-CNT, DSPF-UB, DCW, UserSplitting and CAMF-CU,
- *Trip*: CBPF-CUB-CN, CBPoF-CUB-CNT, DSPF-UB, DCW, UISplitting and CAMF-CU.

The reported results are average of multiple executions based on 5-fold cross-validation. For each dataset, the values in bold are statistically significant better (95% confidence level) than other approaches. The statistical significance has been calculated using the Wilcoxon rank test. The illustrated performances in Table 7.10 show that, our correlation-based filtering approach can outperfoms state of the art in most cases: *CBPF* outperforms the state of the art for *CoMoDa* and *Trip*, and *CBPoF* dominates in the case of *STS*. Though in the case of *Music* the performances of our approach are not as good as the state of the art.
The four datasets used in our experimentations are from different domains (movie, tourism and music), with different characteristics in terms of density, rating distribution and the number of available context and content information. For the case of *Music* where *CBPF* is not very well performing, by consulting the Table 7.2, we can observe that this dataset is built on rating data of very few users (only 42 users), with limited contextual information. But our correlation-based filtering approach is essentially based on context representations and work well when we have enough user in the system, because it is the users who convey the contextual information. This can explain why we have good performances for *CoMoDa*, *STS* and *Trip*, but not for *Music*.

7.3 Conclusion

This chapter details our experimental analysis, where we evaluate our approach on four different contextual datasets (*Comoda*, *STS*, *Music* and *Trip*), and analyze performances based on the characteristics of these datasets.

We can summarize the experiment observations as follows:

- Our experiments validate the positive effect of taking into account contextual information about the user in the recommendation process.
- We proved that our correlation-based approach can easily be plugged to any 2D recommender system in a pre-filtering configuration, as well as a post-filtering configuration. This point allows us to keep the eventual existing traditional 2D recommender system, and contextualize it by plugging our filtering module.
- Furthermore, experimental results show that the *PCC* can effectively catch the influence of context on ratings, but in the cases of small datasets, the *mean deviation* method could be more efficient.
- Based on the experimental results, we can conclude that there is not a single winner between the user- or item-based versions of CBPF/CBPoF, and it depends on the dataset. A similar conclusion can also be done for the comparison of the aggregation and concatenation techniques.

- However, due to the large number of items/users, clustering them has shown to have beneficial effects. In fact, besides the computational cost reduction, our results indicate that grouping items/users can roughly help the model to catch more significantly the influence of context on ratings.

- In addition, we observe that the consideration of context relevance is crucial in the recommendation process. For this purpose, a hybrid method which filter irrelevant context factors and weigh relevant ones in the context representation is the best option.

- Moreover, between the *cosine* and *euclidean* similarity measures, we can not define a single winner, and it also depends on the nature of the dataset. In cases where we have poor contextual information, the *euclidean* similarity can outperforms the *cosine* one.

- Also, we demonstrate that our pre-filtering approach ($CBPF$), can effectively reduce the sparsity problem of the multi-dimensional contextual datasets, by transforming them to 2D datasets, with the consideration of only ratings done in contexts similar to the target user one.

- Depending on the quality of the contextual available data, we can choose to execute a pre-filtering ($CBPF$) or a post-filtering ($CBPoF$): in cases where more complete contextual dataset is available, the pre-filtering can outperform, while when the context is only partially reported, we can have better outcomes with the post-filtering approach.

Chapter 8

Conclusions and Future Work

This chapter briefly summarizes the contributions of this thesis, makes conclusive remarks and indicates some directions for future work.

8.1 Conclusions

In this thesis we work on the integration of contextual information into the recommendation process, in order to propose user-centric recommendations. This research is motivated from one hand by the increasing volume of information on the web, which in consequence, rose the need of information filtering techniques such as recommendation; and from the other hand, the availability of more and more connected devices, which can provide contextual information about the user, and can be integrated into the recommendation techniques in order to provide more relevant recommendations.

From this line of research, we first try to identify the relevant and meaningful contextual information to collect and integrate into the recommendation process. Indeed, the notion of context have grabbed attention of multiple communities since last decades. However there exists not yet a universal definition of it in the literature, and many context factors categorizations, from different points of views have been proposed. By adopting the definition of context as "... any information that can be used to characterize the situation of an entity ..." [2], we propose a new, generic and hierarchical categorization of context factors for CARSs (Context-Aware Recommender Systems). We identify the user context as the union of three categories of context and their respective dimensions: the *physical* context (with its *temporal, spacial, environmental* and *equipment* dimensions), the *personal* context (with its *demographic, social, psychophysiological* and *cognitive* dimensions), and the *technical* context (with its two *hardware* and *data* dimensions). We show that our proposition of context factors categorization can meet the requirements of larger spectrum of application domains (e.g. music, movies, tourism, etc.).

Then, we propose a new approach to integrate such contextual information into the recommendation process. Being in an industrial context, we set an objective of minimal implementation cost and need of external data sources. Indeed, many companies have already a traditional recommender system in production, and want to upgrade it by benefiting from the emerging contextual data in their recommendation process. Our proposed approach, *CBF (Correalation-Based Filtering)*, is a data-driven filtering method that uses the contextual information about the users to filter the ratings pertaining to the target user context in order to efficiently guide the recommendations based on it. It has the advantage to be easily plugged in to any traditional recommender system already in production.
Different filtering strategies can be applied: aiming to tackle the data-sparsity problem of RSs (Recommender Systems), we propose to filter ratings on the basis of the most similar contexts to the target user one.
In this case, the choice of the context representation is crucial inasmuch it will influence the results of the context similarities computations, and thus impact the recommendation results relevance. In *CBF*, we model the context of the users based on the influence of context on ratings, computed by the Pearson Correlation Coefficient (PCC). The distinctive feature of using PCC allows us to catch this influence more precisely, and so to compute more accurate similarities between contexts. We propose two context-aware recommendation methods based on our *correlation-based filtering* approach: a pre-filtering method, *Correlation-Based Pre-Filtering (CBPF)* and a post-filtering one, *Correlation-Based Post-Filtering (CBPoF)*.
We offer several extensions of our approach in order to improve its performances: we propose an alternative for correlation computation, named *mean deviation*. In this method we model the influence of context on ratings by the difference between the mean of ratings given when the context condition is present and those given when this one is absent. Our experiments demonstrate that this correlation method is more suitable for small datasets, as the PCC computation needs a considerable amount of data to catch the real correlation between two variables. Moreover, we show that the influence of the context on ratings could be either item-based or user-based. And we demonstrate that, depending on the dataset, by computing the item-based correlation between the context and ratings, in some cases, and the user-based correlation, in some other cases, we can obtain better performances. Also, we demonstrate that clustering the items/users, not only reduces computational cost, but also increases performances, since it regroups more data in each one of the two variables (context, ratings) that we want to compute the correlation.

Another point that we explore in our context-aware recommendation proposition is the impact degree of each context factors. Indeed, in real world applications, not all context factors have the same importance and impact on ratings. Depending on the application, some context factors can play a more important role than others. For example, in the

case of recipe recommendation, factors like season, available tools around the user, and her cooking competence would be more important. While in music recommendation, activity and psychological context would be more influencing. So, to take this fact into consideration, we propose a hybrid method which filter irrelevant context factors and weight relevant ones in the context vector representations. Experiments show a significant improvement in the performances of our CARS by this method.

Moreover, our approach tries to answer two well-known problem of CARSs: sparsity and recommendation explanation. Indeed, we demonstrate that *CBPF* can efficiently reduce the sparsity level of the dataset which is the input of the recommendation technique. Besides, we proposed a method to generate explanations for our context-aware recommendations based on the context relevance.

Also, as there is very few research in the literature on the comparison of pre- and post-filtering approaches, we compare them based on our correlation-based filtering proposition. We demonstrate that the winner between *CBPF* and *CBPoF* is not always the same, and depends on the richness of the available contextual information. Indeed, the pre-filtering can outperform when more complete contextual information is available, while post-filtering can better manage when context is only partially reported.

To summarize, the performed experimental evaluations proved the effectiveness of taking into account the contextual information into the recommendation process. Also, comparisons with some well-known state of the art approaches show that our correlation-based filtering approach can outperform state of the art approaches. However, our thesis shows that the context-aware recommendation problem remains a data mining problematic, and the performance of a method strongly depends on the nature of the data on which it is applied. It is therefore quite difficult to propose a single generic approach that we could apply as is to any dataset and always obtain the best results. As we can see in the literature, among the numerous CARS propositions, we cannot distinguish a winning approach for all cases. So, we propose a configurable framework, with some parameters to obtain the best rendering in terms of performance for each data. Nevertheless, by a deep data analysis of different type of datasets, we suggest appropriate configurations according to the characteristics of the data.

8.2 Future Work

In this section, we discuss the possible further improvements of the proposed approach, as well as promising lines of future research:

Reducing computational cost by clustering strategy. In our *correlation-based filtering* approach presented in Chapter 5, we mentioned that by clustering items/users, we gain in terms of computational cost in the correlation computing phase. We can gain even more by clustering context situations. Indeed, a limitation of *CBF*, especially when the number of possible context situations increases, is the cost of contexts similarities and local model building computations. So in the future, we would like to apply a clustering on context situations (similarly to [40]) to limit these computational costs.

Different clustering strategy. In our experiments we cluster items/users based on their available static characteristics information. However, this kind of information is not always available, so it would be interesting to test other clustering strategies, like clustering items/users based on ratings. For example, to cluster users, we can represent each user by a vector representation of her ratings for items, and apply a clustering technique based on these vector representations. Though, we are conscious that the huge sparsity of ratings data, could significantly affect the clustering result quality by this method.

User satisfaction study. In Section 6.3, we propose a method to generate explanations for recommendations to users, based on the most relevant context factor. We would like to evaluate our approach by conducting a user study to measure user satisfaction.
The effectiveness of our CARS explanations can be evaluated by one of the two followings ways: the first option is to measure the ratings done prior to and after consumption. The second option is to test the same system with and without an explanation facility, and compare the satisfaction of the users who receive explanations for their recommended items, with the ones that do not receive any explanations [96].

Combination of correlation-based and ontology-based approaches. Our *correlation-based filtering* approach demonstrates that we can efficiently integrate the contextual information about the user, by a data-driven method, without the need of external data sources. Our approach computes the correlation between context and ratings to model the influence of context on ratings. However, in the case of very sparse datasets, where the available data to compute efficiently this correlation is limited, a combination of correlation-based and ontology-based approaches could improve the overall recommendation performances.

Adaptation to group recommendations. Recommender systems intended for group recommendation have gained more attention in these recent years. As the context of the group members are more or less similar, it would be interesting to try to adapt our context-aware recommendations to group. An idea could be to do an aggregation of the results of the CARS for each group member. Two aggregation strategies could be adopted: rate aggregation or

rank aggregation. In the former strategy, we estimate the missing rating for each member group, and aggregate the corresponding values of the group members to obtain the resulting estimated ratings for the group. The later strategy, inspired by the results of [22], takes a set of predicted ranked lists, one for each group member, and produce one combined and ordered recommendations' list for the group.

Implicit data. In this thesis, as most of research in CARS field, we applied our context-aware recommendation approach on explicit data, where the preferences of users for items in different contexts are expressed by ratings (e.g. 1 to 5). However in many real-world recommendation applications, such explicit user feedback is not available, and it is much easier to collect implicit feedback from the behavior and actions of users on the applications, like user clicks, purchases, etc. This kind of implicit feedback provides indirect indication of the user preferences, and so its integration in the recommendation process is more challenging than explicit ratings. Indeed, such observed implicit feedbacks indicate more the positive association between users and items in different contexts, and we have an absence of indicators of negative association between these factors. Therefore the adaptation of *CBF* and its correlation computation to implicit user feedback can be a promising research line for future research.

To conclude, in this section we discussed some propositions of future research among others. CARS is an active research field, where some well-known RS challenges are still poorly explored, like the cold start (phenomena which occurs for new user, new item or new context), scalability or the gray sheep users problem (users with unusual tastes). We can add these subjects to the above list, which make interesting new directions for research. Also, a more complete evaluation of our approach, based on other evaluation metrics like recommendation accuracy, diversity, coverage, novelty, serendipity, etc. would be interesting. For this, a prior stage of large context-aware data collection is needed. Indeed, for now, the available context-aware datasets are very small in comparison to real-word traditional recommendation datasets. And it is essential to test the evaluation metrics cited above, on large context-aware datasets to obtain reliable results.

Acronyms

2D 2-Dimention.

AG Aggregation.

ALS Alternating Least Squares.

ANRT Association Nationale Recherche Technologie.

API Application Program Interface.

CAMF Context-Aware Matrix Factorization.

CARS Context-Aware Recommender System.

CBPF Correlation-Based Pre-Filtering.

CBPoF Correlation-Based Post-Filtering.

CIFRE Convention Industrielle de Formation par la Recherche.

CN Concatenation.

CSLIM Contextual Sparse Linear Method.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

DCM Differential Context Modeling.

DCR Differential Context Relaxation.

DCW Differential Context Weighting.

DSPF Distributional Semantic Pre-Filtering.

EM Expectation-Maximization.

GMM Gaussian Mixture Models.

GPS Global Positioning System.

HAC Hierarchical Agglomerative Clustering.

HC Hierarchical Clustering.

IB Item-Based.

IDE Integrated Development Environment.

IoT Internet of Things.

LVF Las Vegas Filter.

MAE Mean Absolute Error.

MF Matrix Factorization.

OLAP Online Analytical Processing.

PCC Pearson Correlation Coefficient.

POI Point Of Interest.

PSO Particle Swarm Optimization.

RAM Random-Access Memory.

RMSE Root Mean Squared Error.

RS Recommender System.

SGD Stochastic Gradient Descent.

STS South Tyrol Suggests.

SVM Support Vector Machine.

TF Tensor Factorization.

UB User-Based.

References

[1] M. Abbas, M. U. Riaz, A. Rauf, M. T. Khan, and S. Khalid. Context-aware youtube recommender system. In *2017 International Conference on Information and Communication Technologies (ICICT)*, pages 161–164. IEEE, 2017.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[3] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[4] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23:103–145, 2005.

[5] 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. Knowl. Data Eng.*, 17(6):734–749, 2005.

[6] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[7] M. H. Aghdam. Context-aware recommender systems using hierarchical hidden markov model. *Physica A: Statistical Mechanics and its Applications*, 518:89–98, 2019.

[8] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 273–280, 2013.

[9] I. Akermi and M. B. Faiz. Une approche de recommandation proactive dans un environnement mobile. 2015.

[10] I. M. Al Jawarneh, P. Bellavista, A. Corradi, L. Foschini, R. Montanari, J. Berrocal, and J. M. Murillo. A pre-filtering approach for incorporating contextual information into deep learning based recommender systems. *IEEE Access*, 8:40485–40498, 2020.

[11] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proceedings of the 17th international conference on World Wide Web*, pages 199–208. ACM, 2008.

[12] L. R. H. Arigi, Z. A. Baizal, and A. Herdiani. Context-aware recommender system based on ontology for recommending tourist destinations at bandung. In *Journal of Physics: Conference Series*, volume 971. IOP Publishing, 2018.

[13] F. Azouaou and C. Desmoulins. Using and modeling context with ontology in e-learning: the case of teacher's personal annotation. In *Workshop on Applications of Semantic Web Technologies for e-Learning (SWEL@ AH'06), Dublin, Ireland*, 2006.

[14] Z. Bahramian, R. A. Abbaspour, and C. Claramunt. A context-aware tourism recommender system based on a spreading activation method. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2017.

[15] Z. Bahramian, R. Ali Abbaspour, and C. Claramunt. A cold start context-aware recommender system for tour planning using artificial neural network and case based reasoning. *Mobile Information Systems*, 2017, 2017.

[16] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[17] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS'09*, 2009.

[18] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger. Incarmusic: Context-aware music recommendations in a car. *E-Commerce and web technologies*, pages 89–100, 2011.

[19] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context-aware places of interest recommendations and explanations. In *Joint Proceedings of the Workshop on Decision Making and Recommendation Acceptance Issues in Recommender Systems (DEMRA 2011) and the 2nd Workshop on User Models for Motivational Systems: The Affective and the Rational Routes to Persuasion (UMMS 2011). CEUR Workshop Proceedings*, volume 740, pages 19–26, 2011.

[20] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16:507–526, 2012.

[21] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *RecSys*, pages 301–304. ACM, 2011.

[22] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126, 2010.

[23] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, 24, 2014.

[24] M. Bazire and P. Brézillon. Understanding context before using it. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 29–40. Springer, 2005.

[25] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, 9(2):75–79, 2007.

[26] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 43–52. IEEE, 2007.

[27] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[28] I. Benouaret. Un système de recommandation sensible au contexte pour la visite de musée. In *CORIA*, pages 515–524, 2015.

[29] D. Bernardes, M. Diaby, R. Fournier, F. FogelmanSoulié, and E. Viennet. A social formalism and survey for recommender systems. *ACM SIGKDD Explorations Newsletter*, 16(2):20–37, 2015.

[30] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[31] M. Braunhofer, V. Codina, and F. Ricci. Switching hybrid for cold-starting context-aware recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 349–352, 2014.

[32] M. Braunhofer, M. Elahi, and F. Ricci. User personality and the new user problem in a context-aware point of interest recommender system. In *Information and communication technologies in tourism 2015*, pages 537–549. Springer, 2015.

[33] M. Braunhofer, M. Elahi, F. Ricci, and T. Schievenin. Context-aware points of interest suggestion with dynamic weather data management. In *Information and communication technologies in tourism 2014*, pages 87–100. Springer, 2013.

[34] M. Braunhofer, I. Fernández-Tobías, F. Ricci, et al. Parsimonious and adaptive contextual information acquisition in recommender systems. In *IntRS@ RecSys*, pages 2–8, 2015.

[35] R. Burke. Knowledge-based recommender systems. In *Encyclopedia of library and information systems*, page 2000. Marcel Dekker, 2000.

[36] R. Capuano, H. Muccini, and F. Rossi. Catres: a context-aware recommender system for indoor and outdoor museums tours planning. In *KaRS@ CIKM*, pages 31–34, 2019.

[37] A. Chen. Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. In *LoCA*, volume 3479, pages 244–253. Springer, 2005.

[38] G. Chen and D. Kotz. A survey of context-aware mobile computing research. 2000.

[39] V. Codina. *Exploiting Distributional Semantics for Content-based and Context-aware Recommendation: PhD Thesis in Artificial Intelligence*. PhD thesis, V. Codina, 2014.

[40] V. Codina, F. Ricci, and L. Ceccaroni. Distributional semantic pre-filtering in context-aware recommender systems. *User Modeling and User-Adapted Interaction*, 26, 2016.

[41] M. M. Deza and E. Deza. Encyclopedia of distances. In *Encyclopedia of distances*, pages 1–583. Springer, 2009.

[42] T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74, 1993.

[43] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[44] F. Gedikli, D. Jannach, and M. Ge. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382, 2014.

[45] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of communication networks and distributed systems modeling and simulation conference*, volume 2004, pages 270–275. San Diego, CA, USA., 2004.

[46] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. Librec: A java library for recommender systems. In *UMAP Workshops*, 2015.

[47] K. Haruna, M. Akmar Ismail, S. Suhendroyono, D. Damiasih, A. C. Pierewan, H. Chiroma, and T. Herawan. Context-aware recommender system: A review of recent developmental process and future research direction. *Applied Sciences*, 7(12):1211, 2017.

[48] C. Hayes and P. Cunningham. Context boosting collaborative recommendations. *Knowledge-Based Systems*, 17:131–138, 2004.

[49] M. Hericko, I. Rozman, et al. Taxonomy of context-aware systems. *Elektrotehniski Vestnik*, 79(1/2):41, 2012.

[50] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[51] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[52] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86. ACM, 2010.

[53] L. Kaufmann. Clustering by means of medoids. In *Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987*, pages 405–416, 1987.

[54] R. Khoshkangini, M. S. Pini, and F. Rossi. A self-adaptive context-aware group recommender system. In *Conference of the Italian Association for Artificial Intelligence*, pages 250–265. Springer, 2016.

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

[56] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[57] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42, 2009.

[58] P. Korpipää and J. Mäntyjärvi. An ontology for mobile device sensor-based context awareness. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 451–458. Springer, 2003.

[59] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic. Database for contextual personalization. *Elektrotehniški vestnik*, 78, 2011.

[60] B. Y. Lim, A. K. Dey, and D. Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2119–2128, 2009.

[61] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[62] A. Livne, M. Unger, B. Shapira, and L. Rokach. Deep context-aware recommender system utilizing sequential latent context. *arXiv preprint arXiv:1909.03999*, 2019.

[63] A. Lommatzsch, B. Kille, and S. Albayrak. Incorporating context and trends in news recommender systems. In *Proceedings of the International Conference on Web Intelligence*, pages 1062–1068, 2017.

[64] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[65] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[66] M. Mazloom, B. Hendriks, and M. Worring. Multimodal context-aware recommender for post popularity prediction in social media. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 236–244, 2017.

[67] S. A. Mohamed, T. H. A. Soliman, and A. A. Sewisy. A context-aware recommender system for personalized places in mobile applications. *Int. J. Adv. Comput. Sci. Appl*, 7(3):442–448, 2016.

[68] A. Montes-García, J. M. Álvarez-Rodríguez, J. E. Labra-Gayo, and M. Martínez-Merino. Towards a journalist-based news recommendation system: The wesomender approach. *Expert Systems with Applications*, 40(17):6735–6741, 2013.

[69] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[70] E. Negre. *Information and Recommender Systems*. John Wiley & Sons, 2015.

[71] E. Negre, F. Ravat, and O. Teste. Olap queries context-aware recommender system. In *International Conference on Database and Expert Systems Applications*, pages 127–137. Springer, 2018.

[72] C. P. Nguyen. *Conception d'un système d'apprentissage et de travail pervasif et adaptatif fondé sur un modèle de scénario*. PhD thesis, 2010.

[73] A. Odić, M. Tkalčič, J. Tasič, and A. Košir. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. CEUR-WS. org, 2012.

[74] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25:74–90, 2013.

[75] F. Orciuoli and M. Parente. An ontology-driven context-aware recommender system for indoor shopping based on cellular automata. *Journal of Ambient Intelligence and Humanized Computing*, 8(6):937–955, 2017.

[76] R. Pagano, P. Cremonesi, M. Larson, B. Hidasi, D. Tikk, A. Karatzoglou, and M. Quadrana. The contextual turn: From context-aware to context-driven recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems, RecSys*, volume 16, pages 249–252, 2016.

[77] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE transactions on knowledge and data engineering*, 20(11):1535–1549, 2008.

[78] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, pages 265–268. ACM, 2009.

[79] M. Pichl, E. Zangerle, and G. Specht. Improving context-aware music recommender systems: beyond the pre-filtering approach. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 201–208, 2017.

[80] C. E. Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

[81] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644. ACM, 2011.

[82] D. Riboni and C. Bettini. Cosar: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, 2011.

[83] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[84] F. Ricci, L. Rokach, and B. Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.

[85] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[86] I. A. P. Santana and M. A. Domingues. A systematic review on context-aware recommender systems using deep learning and embeddings. *arXiv preprint arXiv:2007.04782*, 2020.

[87] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[88] C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471, 1987.

[89] P. Spirtes. Introduction to causal inference. *Journal of Machine Learning Research*, 11(5), 2010.

[90] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004.

[91] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.

[92] C. V. Sundermann, R. de Pádua, V. R. Tonon, M. A. Domingues, and S. O. Rezende. A context-aware recommender method based on text mining. In *EPIA Conference on Artificial Intelligence*, pages 385–396. Springer, 2019.

[93] C. V. Sundermann, M. A. Domingues, M. da Silva Conrado, and S. O. Rezende. Privileged contextual information for context-aware recommender systems. *Expert Systems with Applications*, 57:139–158, 2016.

[94] C. V. Sundermann, M. A. Domingues, R. A. Sinoara, R. M. Marcacini, and S. O. Rezende. Using opinion mining in context-aware recommender systems: A systematic review. *Information*, 10(2):42, 2019.

[95] L. Tamine-Lechani, M. Boughanem, and M. Daoud. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowledge and Information Systems*, 24(1):1–34, 2010.

[96] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pages 801–810. IEEE, 2007.

[97] S. Ujjin and P. J. Bentley. Particle swarm optimization recommender system. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 124–131. IEEE, 2003.

[98] Z. Vahidi Ferdousi, D. Colazzo, and E. Negre. Cbpf: leveraging context and content information for better recommendations. In *International Conference on Advanced Data Mining and Applications*, pages 381–391. Springer, 2018.

[99] Z. Vahidi Ferdousi, D. Colazzo, and E. Negre. Correlation-based pre-filtering for context-aware recommendation. In *Proceedings of the 16th IEEE International Conference on Pervasive Computing and Communications Workshops (CoMoRea)*. IEEE, 2018.

[100] Z. Vahidi Ferdousi, D. Colazzo, and E. Negre. Recommandations contextuelles à base de corrélations. 2018.

[101] Z. Vahidi Ferdousi, E. Negre, and D. Colazzo. Context factors in context-aware recommender systems. In *AISR 2017: Atelier interdisciplinaire sur les systèmes de recommandation*, 2017.

[102] B. Vargas-Govea, G. González-Serna, and R. Ponce-Medellın. Effects of relevant contextual features in the performance of a restaurant recommender system. *ACM RecSys*, 11(592):56, 2011.

[103] D. Véras, R. Prudêncio, and C. Ferraz. Cd-cars: Cross-domain context-aware recommender systems. *Expert Systems with Applications*, 135:388–409, 2019.

[104] N. M. Villegas, C. Sánchez, J. Díaz-Cely, and G. Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173–200, 2018.

[105] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 871–878. IEEE, 2007.

[106] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[107] P. Yu, L. Lin, and J. Wang. A novel framework to alleviate the sparsity problem in context-aware recommender systems. *New Review of Hypermedia and Multimedia*, 23(2):141–158, 2017.

[108] V. N. Zhao, M. Moh, and T.-S. Moh. Contextual-aware hybrid recommender system for mixed cold-start problems in privacy protection. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 400–405. IEEE, 2016.

[109] Y. Zheng, R. Burke, and B. Mobasher. Optimal feature selection for context-aware recommendation using differential relaxation. *Acm Recsys*, 12, 2012.

[110] Y. Zheng, R. Burke, and B. Mobasher. Splitting approaches for context-aware recommendation: An empirical study. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 274–279. ACM, 2014.

[111] Y. Zheng and A. A. Jose. Context-aware recommendations via sequential predictions. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 2525–2528, 2019.

[112] Y. Zheng, B. Mobasher, and R. Burke. Contexts recommendation using multi-label classification. In *Proceedings of the 13th IEEE/WIC/ACM International Conference on Web Intelligence (WI 2014)*. IEEE/WIC/ACM, 2014.

[113] Y. Zheng, B. Mobasher, and R. Burke. Cslim: Contextual slim recommendation algorithms. In *Proceedings of the 8th ACM RecSys*, pages 301–304. ACM, 2014.

[114] Y. Zheng, B. Mobasher, and R. Burke. Carskit: A java-based context-aware recommendation engine. In *Proceedings of the 15th IEEE International Conference on Data Mining Workshops*. IEEE, 2015.

[115] Y. Zheng, B. Mobasher, and R. Burke. Similarity-based context-aware recommendation. In *WISE*, pages 431–447. Springer, 2015.

[116] Y. Zheng, B. Mobasher, and R. Burke. User-oriented context suggestion. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 249–258, 2016.

[117] A. Zimmermann, A. Lorenz, and R. Oppermann. An operational definition of context. In *International and Interdisciplinary Conference on Modeling and Using Context*, pages 558–571. Springer, 2007.

RÉSUMÉ

Avec l'augmentation du volume de données produit par diverses sources, nous avons un besoin croissant de systèmes de recommandation, qui filtrent les données pour aider les utilisateurs à trouver l'information appropriée. Afin de satisfaire encore plus les besoins des utilisateurs et générer des recommandations plus pertinentes, un nouveau type de systèmes de recommandation, nommé système de recommandation contextuel (CARS), intègre les informations contextuelles des utilisateurs dans le processus de recommandation. Cependant, il n'existe toujours pas de définition unique du contexte. L'objectif de cette thèse est, dans un premier temps, d'identifier les facteurs de contexte pertinents pour les CARSs, afin d'améliorer les précédentes propositions de l'état de l'art, et pouvant être utilisés pour un large éventail d'applications. Ensuite, nous proposons une nouvelle représentation du contexte, ainsi qu'une approche pour intégrer ce type d'information dans un système de recommandation. Nous représentons le contexte en nous basant sur l'influence du contexte sur les scores donnés par les utilisateurs aux éléments, calculée à l'aide du Coefficient de Corrélation de Pearson. Ensuite nous filtrons les données à partir de ces représentations, afin de les intégrer dans le processus de recommandation. Nous présentons deux approches de recommendations contextuelles à base de pré-filtrage et post-filtrage. De plus, nous proposons une méthode pour générer des explications pour nos recommandations contextuelles. Par des expérimentations, nous démontrons que notre approche réduit la parcimonie, problématique bien connue des CARS, et peut également améliorer les performances de l'état de l'art.

MOTS CLÉS

Recommandation contextuelle, Filtrage à base de corrélation, Intégration d'information contextuelle.

ABSTRACT

With the rise in volume of data from various sources, we have an increasing need of recommender systems, which provide a data filtering to help users to find appropriate information. To satisfy even more users' needs and generate more relevant recommendations, a new kind of recommender systems called CARS integrates contextual information related to the users in their recommendation process. However there exists no unique definition for context. In this thesis we firstly identify relevant context factors for CARSs, to improve upon previous propositions, which can be used for a large spectrum of applications. Then we propose a new context representation and approach to integrate this kind of information into a recommender system. We make a relevant representation of the context, based on the influence of context on ratings, calculated using the Pearson Correlation Coefficient. We present a pre-filtering and a post-filtering context-aware recommender systems based on this representation. We propose a method to generate explanations for our context-aware recommendations. Also, we demonstrate that our approach can reduce the well-known sparsity problem of CARS and outperform state of the art approaches.

KEYWORDS

Context-aware recommendation, Correlation-based filtering, Contextual information integration.